NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

PHARMACY STOCK CONTROL PROGRAM

GRADUATION PROJECT
COM-400

Student : Özerk RAİF (20043190)

Supervisor : Ümit İLHAN

Nicosia-2007

NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

Pharmacy Stock Control Program

GRADUATION PROJECT
COM-400

Student : Özerk RAİF (20043190)

Supervisor : Ümit İLHAN

Nicosia-2007

ACKNOWLEDGMENT

First of all I would like to thank to my supervisor Mr. Ümit İlhan for his endless support, guidance and friendship in the course of the preparation of this work.

On the completion of this thesis, I would like to express my sincere thanks to Dr. Kaan Uyar for his invaluable advice and belief in me and my academic work. His ideas and tremendous support had a major influence on this thesis.

I gratefully dedicate this thesis to my parents that I am deeply indebted to for their love, support and encouragement for all my life.

Finally, I would like to thank the many faculty and staff who have enriched my experience at Near East University, specially Mr. Okan Dodangil and the vice president Prof. Dr Fakhreddin Mamedov..

ABSTRACT

Computer science has developed tremendously over the last decades. It is possible to state this in terms of both hardware and software. Programming is always providing the scientists a continuous systematic development in their studies and research. In this project it's been constructed a special program related to Pharmacy Automation. The pharmacy industry should not be regarded as an isolated and unrelated field from the other industries but it is within this framework that the history of pharmacy development should be examined. New concepts in pharmacy design have been developed more recently in an effort to meet the changing preferences and new characteristics.

The pharmacy automation program consists of many departments like, sell, customers, products and purchase. The program that been given in this thesis, resumes that the briefly in a quick time in order to have quick and economic services. On the other hand, the pharmacy development is suitable for researchers and students in computer science; the development of pharmacy automation programs is designed to help compute professionals who want to learn about this exciting field and to serve as a basic reference.

The aim of this project is to create and to develop a project in a scientific method to introduce the gab between scientific theoretical life and work normal life.

In this project, it's been constructed a pharmacy automation program for the availability of information is incrementally important in all over the world, how to make a cays process in order to have a quick research, data process, analysis process.

Finally, full file enclosed full details about the project.

TABLE OF CONTENTS

CHAPTER TREE

CHAPTER FOUR

LIST OF FIGURES

1. VISUAL BASIC PROGRAMMING

1.1 Introduction to Visual Basic

The "Visual" part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add rebuilt objects into place on screen. If you've ever used a drawing program such as Paint, you already have most of the skills necessary to create an effective user interface.

The "Basic" part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language.

Visual Basic is an Object-Oriented Programming (OOP) language and a Rapid Application Development (RAD) environment from Microsoft. Visual Basic provides tools for Internet programming, and helps developers quickly create and deploy enterprise client/server applications, most often to access both local and remote databases.

It's evolved from the earlier DOS version called BASIC (Beginners' All-Purpose Symbolic Instruction Code) in which programming is done in a text-only environment and the program is executed sequentially. BASIC has advanced through many versions since it was first created in 1964 at Dartmouth College. This initial version of BASIC allowed students to write programs to run the Time-Sharing System, one of the first time-share computer systems in the United States.

Visual Basic has diverged from BASIC into an Object-oriented Programming Language, and even further into a visual and action, or events, driven language. It offers a GUI (Graphical User Interface) to allow developers to choose and modify pre-selected sections of code

written in BASIC syntax. Utilizing a graphical environment, Visual Basic developers can select and edit program objects independently. Consequently, a fully functional VB Program is made up of many subprograms that can be executed independently or grouped together.

The Visual Basic programming language is not unique to Visual Basic. The Visual Basic system Edition included in Microsoft Excel, Microsoft Access, and many other Windows applications uses the same language. The Visual Basic Scripting Edition (VBScript) is a widely used scripting language and a subset of the Visual Basic language. The investment you make in learning Visual Basic will carry over to these other areas.

Whether your goal is to create a small utility for yourself or your work group, a large enterprise-wide system, or even distributed applications spanning the globe via the Internet, Visual Basic has the tools you need.

Data access features allow you to create databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.

ActiveX™ technologies allow you to use the functionality provided by other applications, such as Microsoft Word processor, Microsoft Excel spreadsheet, and other Windows applications. You can even automate applications and objects created using the Professional or Enterprise editions of Visual Basic.

Internet capabilities make it easy to provide access to documents and applications across the Internet or intranet from within your application, or to create Internet server applications.

Visual Basic is designed for simple, rapid application development, and can be used to prototype an application that will later be written in a more difficult but efficient language. Other object-oriented programming languages such as C++, Java, and Smalltalk, operate in text-only environments, and do not employ a GUI to build programs.

1.2 Brief History

In 1988, Alan Cooper, the 'father' of Visual Basic, produced a drag-and-drop shell prototype for the BASIC programming language. The shell prototype, named Tripod, included a widget control box and a small language engine. After showing it to Bill Gates, Microsoft negotiated to buy the concept and code-named it Ruby. Microsoft joined Ruby with their current BASIC

programming environment, QuickBasic, resulting in the first tool that allowed developers to create Windows applications quickly, easily, and visually (code named Thunder).

In 1991, Microsoft released Visual Basic 1.0. It was the first visual development tool from Microsoft, and was designed to compete with C, C++, Pascal, and any other well- known programming language at the time. However "when it came out, Visual Basic wasn't a success. It wasn't until Microsoft released VB 2.0 in 1993 that people really started to discover the power of the language, and when Microsoft released VB 3.0 it had become the fastest growing programming language on the market.

1.3 The Basics of a Programming Language

Traditional program languages are composed of commands (often called statements), operators, variables and data. Variables represent data and the statements and operators operate on the data to produce the require output.

1.4 Visual Basic is Windows Development Language

The VB is Windows development language, that's why you must be familiar with the Windows environment. Windows involves three key concepts:

1. Window

A window is a simply rectangular region with its own boundaries.

Examples of windows are:

An Explorer window in Windows 95.

A document window in word processor.

Dialog box that pop up window and reminds you of an appointment.

A command button.

Icons.

Text boxes.

Option boxes.

Menu bars.

The Microsoft Windows Operating system manages all of these many windows by assigning each one a unique id number. The system continually monitors each of these windows for signs of activity or events.

2. Events

An event is an action recognized by a form or control. Events can occur through user action (response) such as a mouse click or a key press using objects of window (through programmatic control), or even as a result of another window's action.

Event-driven applications execute Basic code in response to an event. Each form and control in VB has a predefined set of events. If one of these events occurs and there is a user code in the associated event procedure, VB invokes that code.

For example most objects recognize a Click event. If a user clicks a form (object), code in the form's Click event procedure is executed. If a user clicks a command button, code in the button's click event procedure is executed.

Each time an event occurs, it causes a message to be sent to the O.S. The system processes the message and broadcasts it to the other windows. Each window can take the appropriate action based on its own instructions from dealing with that particular message.

Fortunately, VB insulates you from having to deal with all of the low-level message handling. Many of the messages are handled automatically by VB.

This allows you to quickly create powerful applications without having to deal with unnecessary details.

- **Understanding the Event-Driven Model**

Programs in conventional (traditional or procedural) programming languages run from the top down. For older programming languages, execution starts from the first line and moves with the flow of the program to different parts as needed.

A VB program usually works completely different. The code doesn't follow a predefined path. It executes different code section in response to events.

The core of a VB program is a set of independent pieces of code that are activated by, and so respond to, only the events they have been told to recognize.

The programming code in VB that tells your program how to respond to events (event procedure). An event procedure is a body of code that is only executed in response to an external event.

Your code can also trigger events during execution. It is for this reason that it is important to understand the event-driven model and keep it in mind when designing your application in windows environment.

1.5 Developing an Application in Visual Basic

As you develop an application, you work with a project to manage all the different files that make up the application. A project consists of:
One project file that keeps track of all the components (.vbp).
One file for each form (.frm).
One binary data file for each form containing data for properties of controls on the form (.frx). These files are not editable and are automatically generated for any .frm file that contains binary properties, such as Picture or Icon.
Optionally, one file for each class module (.cls).
Optionally, one file for each standard module (.bas).
Optionally, one or more files containing ActiveX controls (.ocx).
Optionally, a single resource files (.res).

The project file is simply a list of all the files and objects associated with the project, as well as information on the environment options you set. This information is updated every time you save the project. All of the files and objects can be shared by other projects as well.

1.5.1 Building Applications with Visual Basic

There are essentially 3 basic phases of building a computer application:

1. The Design phase, which is analogous to an architect designing a building before it is built.
2. The programming phase, where sets of instructions in the form of functions and subroutines are written to carry out the events of the application.
3. The final step, which actually never ends, is the de- bugging phase.

The last two phases are an iterative procedure, where the programmer should be continuously evaluating potential errors that might arise, and writing code to handle obvious errors. All programs have bugs, but good programs have fewer bugs.

1.5.2 Design VB Applications

Here is a summary of the steps you take to design a VB application:

- Customize the windows that the user sees.
- Decide what events the controls on the window should recognize.
- Write the event procedures for those events.

1.5.3 Running Application

Here is what happens when the application is running:

- The application starts and a form is loaded and displayed
- The form (or a control on the form) receives an event. The event might be caused by the user(for example , a keystroke), by the system(for example , a timer event), or indirectly by your code(for example, a Load event when your code loads a form)
- If you have written an event procedure, VB executes the code.
- The application waits for the next event

1.6 New Tools in data access

ADO (ActiveX Data Objects)

All Editions This new data access technology features a simpler object model, better integration with other Microsoft and non-Microsoft technologies, a common interface for both local and remote data access, removable and disconnected record sets, a user-accessible data binding interface, and hierarchical record sets.

1.6.1 ADO Data Control

All Editions A new OLEDB-aware data source control that functions much like the intrinsic Data and Remote Data controls, in that it allows you to create a database application with minimum code.

Visual Database Tools Integration (Query Designer and Database Designer)

Enterprise Edition Visually create and modify database schemas and queries: Create SQL Server and Oracle database tables, drag and drop to create views, and automatically change column data types.

Many data access applications created with earlier versions of Visual Basic store and manage data using the Microsoft Jet database engine, the engine used by Microsoft Access. These applications use Microsoft Data Access Objects (DAO) to access and manipulate data.

When you have completed all the files for a project, you can convert the project into an executable file (.exe): From the File menu, choose the Make project.exe command.

Interacting with Data in a Microsoft Jet/Microsoft Access Database

Now we can use Microsoft ActiveX Data Objects (ADO) to easily manipulate data in a variety of database formats, including Microsoft Jet format. We may still be able to use DAO to work with your local Microsoft Jet databases, but for new applications you'll probably want to use ADO and the new data access features of Visual Basic.

1.6.2 DataGrid Control

All Editions An OLEDB-aware version of DBGrid, the control allows you to quickly build an application to view and edit recordsets. It also supports the new ADO Data control

Working with Projects.

2. MICROSOFT ACCESS

2.1 Introduction to Microsoft Access

Access is one of the well-known implementations of the relational data model on the PC platform. It is considered as part of an integrated set of tools for creating and managing databases on the PC Windows platform. The database applications for Access may range from personal applications, such as maintaining an inventory of your personal audio and video collection, to small business applications, such as maintaining business-specific customer information. With compliance to the Microsoft Open Database Connectivity (ODBC) standard and the prevalence of today's client-server architectures, PC relational databases may be used as a front-end to databases stored on non-PC platforms. For example, an end user can specify ad hoc queries graphically in Access over an Oracle database stored on a UNIX server.

Access provides a database engine and a graphical user interface (GUI) for data definition and manipulation, with the power of SQL. It also provides a programming language called Access Basic. Users can quickly develop forms and reports for input/output operations against the database through the use of Wizards, which are interactive programs that guide the user through a series of questions in a dialog mode. The definition of the forms and reports is interactively accomplished when the user designs the layout and links the different fields on the form or report to items in the database. Access 97 (the latest release of Access at the time of this writing) also provides the database developer with hyperlinks as a native data type, extending the functionality of the database with the ability to share information on the Internet.

Access is an RDBMS that has several components. One component is the underlying database engine, called the Microsoft Jet engine which is responsible for managing the data. Another component is the user interface, which calls the engine to provide data services, such as storage and retrieval of data. The engine stores all the application data (tables, indexes, forms, reports, macros, and modules) in a single Microsoft database file (.mdb file). The engine also provides advanced capabilities, such as heterogeneous data access through ODBC, data validation, concurrency control using locks, and query optimization.

Access works like a complete application development environment, with the internal engine serving to provide the user with RDBMS capabilities. The Access user interface provides Wizards and Builders to aid the user in designing a database application. Builders are interactive programs that help the user build syntactically correct expressions. The programming model used by Access is event-driven. The user builds a sequence of simple operations, called macros, to be performed in response to actions that occur during the use of the database application. While some applications can be written in their entirety using macros, others may require the extended capabilities of Access Basic, the programming language provided by Access.

There are different ways in which an application with multiple components that includes Access can be integrated. A component (in Microsoft terminology) is an application or development tool that makes its objects available to other applications. Using automation in Visual Basic, it is possible to work with objects from other components to construct a seamless integrated application. Using the Object Linking and Embedding (OLE) technology, a user can include documents created in another component on a report or form within Access. Automation and OLE are distinct technologies, which are a part of the Component Object Model (COM), a standard proposed by Microsoft.

2.2 Data Definition of Access Databases

Although Access provides a programmatic approach to data definition through Access SQL, its dialect of SQL, the Access GUI provides a graphical approach to defining tables and relationships among them. A table can be created directly in a design view or it can be created interactively under the guidance of a table wizard. Table definition contains not only the structure of the table but also the formatting of the field layout and masks for field inputs, validation rules, captions, default values, indexing, and so on. The data types for fields include text, number, date/time, currency, Yes/no (boolean), hyperlink, and AutoNumber, which automatically generates sequential numbers for new records. Access also provides the capability to import data from external tables and to link to external tables.

Field Properties window for displaying the properties of the Fields. The format property provides for a default display format. The input mask provides automatic formatting characters for display during data input in order to validate the input data. For example, the input mask for SSN displays the hyphen positions and indicates that the other characters are

digits. The caption property specifies the name to be used on forms and reports for this field. A blank caption specifies the default, which is the field name itself. A default value can be specified if appropriate for a particular field. Field validation includes the specification of validation rules and validation text—the latter displayed when a validation rule is violated. Other field properties include specifying whether the field is required—that is, NULL is not allowed—and whether textual fields allow zero length strings. Another field property includes the index specification, which allows for three possibilities: (1) no index, (2) an index with duplicates, or (3) an index without duplicates. In the case of primary key, the field is indexed with no duplicates allowed.

In addition to the Field Properties window, Access also provides a Table Properties window. This is used to specify table validation rules, which are integrity constraints across multiple columns of a table or across tables.

2.3 Defining Relationships and Referential Integrity Constraints

Access allows interactive definition of relationships between tables—which can specify referential integrity constraints-via the Relationships window. To define a relationship, the user first adds the two tables involved to the window display and then selects the primary key of one table and drags it to where it appears as a foreign key in the other table. This action pops up another window that prompts the user for further information regarding the establishment of the relationship, the user checks the "Enforce Referential Integrity" box if Access is to automatically enforce the referential integrity specified by the relationship. The user may also specify the automatic cascading of updates to related fields and deletions of related records by selecting the appropriate boxes. The "Relationship Type" is automatically determined by Access based on the definition of the related fields. If only one of the related fields is a primary key or has a unique index, then Access creates a one-to-many relationship, indicating that an instance (value) of the primary key can appear many times as an instance of the foreign key in the related table. This is the case in our example because DNUMBER is the primary key of DEPARTMENT and DNO is not the primary key of EMPLOYEE nor does it have a unique index defined on it. If both fields are either keys or have unique indexes, then Access creates a one-to-one relationship.

Although specifying a relationship is the mechanism used to specify referential integrity between tables, the user need not choose the option to enforce referential integrity because relationships are also used to specify implicit join conditions for queries. For example, if no

relationship is pre-specified during the graphical design of a query, then a default join of the related fields is performed if related tables are selected for that query, regardless of whether referential integrity is enforced or not. Access chooses an inner join as the default join type but the user may choose a right or left outer join by clicking on the "Join Type" box and selecting the appropriate join type.

2.4 Data Manipulation in Access

The data manipulation operations of the relational model are categorized into retrieval queries and updates (insert, delete, and modify operations). Access provides for query definition either graphically through a QBE interface or programmatically through Access SQL. The user has the ability to design a graphical query and then switch to the SQL view to examine the SQL query generated by Access. Access provides for update operations through forms that are built by the application programmer, by direct manipulation of the table data in Datasheet view, or through the Access Basic programming language.

Retrieval operations are easily specified graphically in the Access QBE interface. in QBE and SQL. To establish a join that had not been prespecified the user selects the join attribute from one table and drags it over to the join attribute in the other table. To include an attribute in the query, the user drags it from the top window to the bottom window. For attributes to be displayed in the query result, the user checks the "Show" box. To specify a selection condition on an attribute, the user can type an expression directly in the "Criteria" grid or use the aid of an Expression Builder. To see the equivalent query in Access SQL, the user switches from the QBE Design View to the SQL View.

Update operations on the database are typically guided by the use of forms that incorporate the business rules of the application. There is also a Datasheet view of a table that the sophisticated end user can use to insert, delete, or modify data directly by choosing "open table" from a database window. These updates are subject to the constraints specified through the data definition process, including data types, input masks, field and table validation rules, and relationships.

2.5 Designing of the Databases

A database is only useful if it is designed to meet the specific needs of its users. Good database design requires careful planning to determine the fields, tables, and relationships needed to satisfy the data input and output requirements. The following guidelines help to insure that the database will be able to produce the needed results:

- Identify all of the fields needed to produce the required information. Consider the type of information to be stored in the database and the type of reports that must be generated from the data. Plan fields that will produce this information.

- Group related fields into tables. Look for logical grouping of field information. For example, all information pertaining to students might be placed in one table. All information pertaining to counselors might be placed in a second table.

- Determine each table's primary key field. Look for a field that uniquely identifies each record. Such fields include social security numbers, identification codes, part numbers, or product serial numbers. It might be necessary to assign a unique number to each record or to allow Access to assign one automatically.

- Include a common field in related tables. The common field is used to connect one table logically with another table. For example, each student record might include a counselor code that matches the counselor code listed for each counselor in the counselor table.

- Avoid redundancy. Data redundancy occurs when the data is stored in more than one place in the database. With the exception of the common field(s) to connect tables, redundancy wastes storage space and can increase the likelihood that data will be entered inconsistently.

- Determine the properties of each field. Field properties include field name, field type, maximum number of characters, field description, and validity

Microsoft Access database mainly consists of: Database File, Table, Record, Field, Field

value, Data-type. Here is the Hierarchy that Microsoft Access uses in breaking down a database

Database File: This is your main file that encompasses the entire database and that is saved to your hard-drive or floppy disk, it's A collection of related tables. (Access is a relational database).

Table: A table is a collection of data about a specific topic (A collection of records.). There can be multiple tables in a database.

Field: Fields are the different categories within a Table. Tables usually contain multiple fields, it's a single characteristic or attribute of a person, place, object, event or idea (a column).

Field Value: The specific value, or content, of a field.

Primary Key - A field, or a collection of fields, whose values uniquely identify each record (unique identifier).

Common Field: A field that appears in both tables. The common field is used to connect tables.

Record: A set of field values that describe a person, place, object, event, or idea (a row).

Datatypes: Datatypes are the properties of each field. A field only has 1 datatype.

2.6 Naming Fields

Each field on a database must have a name (this is also true for anything in the computer). The name held by a field allows you, the database developer, and the operating system, to refer to that particular field.

It is best to choose a field name that describes the purpose of the field so that it is easy to remember. In addition, the following rules apply to naming fields:

- Must not exceed 255 characters. You should limit the name of a variable to 30 characters

- A name can contain letters, numbers, spaces, and special characters except for a period, exclamation mark, accent mark, and square brackets
- A name cannot begin with a space, Must begin with a letter (a-z or A-Z)
- A name must be unique within a table, but it can be used again in another table.

Experienced users of databases capitalize the first letter of each word in a field name, avoid using long field names, use standard abbreviations, and avoid using spaces in field names.

2.7 Assigning Field Data Types and Defining Properties

After specifying a name of the field, you can decide what type of data can be entered into that field. The data type determines the field values that can be entered in the field. Access provides the following data types:

Text: Allows field values containing letters, digits, spaces and special characters. Field size: 0 - 255 characters.

Memo: Allows field values containing letters, digits, spaces and special characters that make up long comments. Field size: 1-64,000 characters.

Number: Allows positive and negative numbers as field values. Field size: 1-15 digits.

Date/Time: Allows field values containing dates and times to December 31, 9999. Field size: 8 bytes.

Currency: Allows field values similar to number data type using the currency format. Field size: 15 digits on the left side of decimal and 4 digits on the right side.

AutoNumber: Integers controlled by Access. Access automatically inserts a value field and numbers records as they are entered. Field size: 9 digits.

Yes/No: Limits values to yes and no, on and off, or true and false. Field size: 1 character.

Hyperlink: Consists of a hyperlink address. Field size: 1 gigabyte maximum.

Lookup Wizard: Creates a field that lets you look up a field value in another table or in a predefined list of values. Field size: Same as the primary key field used to perform the lookup.

Each data type allows for a set of properties that help to insure that the data is entered accurately. Such properties include making fields required, selecting default values, entering captions, and specifying data validation rules and text.

2.8 Specifying a Primary Key

A primary key uniquely identifies each record in the table. Access does not allow for duplicate values in the primary key field. Once a primary key field is selected, every record must have a value in the primary key field. Access stores the records on the disk in the order they are entered but displays them in order by the field values of the primary key. In addition, Access responds faster to requests for specific records based on the primary key.

2.9 Adding Records

Records are added to tables by using the table datasheet or by creating a form. A table datasheet provides a simple way to add records. A table datasheet displays records in rows and columns. Each row is a separate record in the table, and each field is a separate column. When a table contains many fields, it is useful to create a form to maintain the records. While forms can be customized, Access provides a wizard that automatically creates a form for data entry.

2.10 Retrieving and Reporting Information

The process of retrieving information from a database is known as querying. Access provides powerful query capabilities that allow the user to display selected fields and records from a table, sort records, perform calculations, find and display information from two or more tables, and generate professionally designed reports.

3. THE TOOLS USED IN THE PROGRAM

3.1 The ADO Data Control

The ADO Data control uses Microsoft ActiveX Data Objects (ADO) to quickly create connections between data-bound controls and data providers. Data-bound controls are any controls that feature a DataSource property. Data providers can be any source written to the OLE DB specification. You can also easily create your own data provider using Visual Basic's class module.

Although you can use the ActiveX Data Objects directly in your applications, the ADO Data control has the advantage of being a graphic control (with Back and Forward buttons) and an easy-to-use interface that allows you to create database applications with a minimum of code.

Several of the controls found in Visual Basic's Toolbox can be data-bound, including the CheckBox, ComboBox, Image, Label, ListBox, PictureBox, and TextBox controls. Additionally, Visual Basic includes several data-bound ActiveX controls such as the DataGrid, DataCombo, Chart, and DataList controls. You can also create your own data-bound ActiveX controls, or purchase controls from other vendors.

Previous versions of Visual Basic featured the intrinsic Data control and the Remote Data control (RDC) for data access. Both controls are still included with Visual Basic for backward compatibility. However, because of the flexibility of ADO, it's recommended that new database applications be created using the ADO Data Control

3.1.1 Possible Uses

- Connect to a local or remote database.
- Open a specified database table or define a set of records based on a Structured Query Language (SQL) query or stored procedure or view of the tables in that database.

- Pass data field values to data-bound controls, where you can display or change the values.

- Add new records or update a database based on any changes you make to data displayed in the bound controls.

To create a client, or front-end database application, add the ADO Data control to your forms just as you would any other Visual Basic control. You can have as many ADO Data controls on your form as you need. Be aware, however, that the control is a comparatively "expensive" method of creating connections, using at least two connections for the first control, and one more for each subsequent control.

3.2 The DataGrid Control

Displays and enables data manipulation of a series of rows and columns representing records and fields from a Recordset object.

Syntax

DataGrid

Remarks

The data-aware DataGrid control appears similar to the Grid control; however, you can set the DataGrid control's DataSource property to a Data control so that the control is automatically filled and its column headers set automatically from a Data control's Recordset object. The DataGrid control is really a fixed collection of columns, each with an indeterminate number of rows.

Each cell of a DataGrid control can hold text values, but not linked or embedded objects. You can specify the current cell in code, or the user can change it at run time using the mouse or the arrow keys. Cells can be edited interactively, by typing into the cell, or programmatically. Cells can be selected individually or by row.

If a cell's text is too long to be displayed in the cell, the text wraps to the next line within the same cell. To display the wrapped text, you must increase the cell's Column object's Width property and/or the DataGrid control's RowHeight property. At design time, you can change

the column width interactively by resizing the column or by changing the column's width in the Column object's property page.

Use the DataGrid control's Columns collection's Count Property and the Recordset object's RecordCount property to determine the number of columns and rows in the control. A DataGrid control can have as many rows as the system resources can support and up to 32767 columns.

When you select a cell, the ColIndex property is set, thus selecting one of the Column objects in the DataGrid object's Columns collection. The Text and Value properties of the Column object reference the contents of the current cell. The data in the current row can be accessed using the Bookmark property, which provides access to the underlying Recordset object's record. Each column of the DataGrid control has its own font, border, word wrap, and other attributes that can be set without regard to other columns. At design time, you can set the column width and row height and establish columns that are not visible to the user. You can also prevent users from changing the formatting at run time.

Note If you set any of the DataGrid column properties at design time, you will need to set all of them in order to maintain the current settings.

Note If you use the Move method to position the DataGrid control, you may need to use the Refresh method to force it to repaint.

The DataGrid control functions similarly to the DBGrid control except that it doesn't support an unbound mode.

3.3 The Combo Box Control

A combo box control combines the features of a text box and a list box. This control allows the user to select an item either by typing text into the combo box, or by selecting it from the list.

3.3.1 When to Use a Combo Box Instead of a List Box

Generally, a combo box is appropriate when there is a list of suggested choices, and a list box is appropriate when you want to limit input to what is on the list. A combo box contains an edit field, so choices not on the list can be typed in this field.

In addition, combo boxes save space on a form. Because the full list is not displayed until the user clicks the down arrow (except for Style 1, which is always dropped down), a combo box can easily fit in a small space where a list box would not fit.

3.3.2 Drop-down Combo Box

The user can either enter text directly (as in a text box) or click the detached arrow at the right of the combo box to open a list of choices. Selecting one of the choices inserts it into the text portion at the top of the combo box. The user also can open the list by pressing ALT+ DOWN ARROW when the control has the focus.

3.3.3 Accessing List Items with the List Property

The List property provides access to all items in the list. This property contains an array in which each item in the list is an element of the array. Each item is represented in string form. To refer to an item in the list, use this syntax:

box.List(index)

The box argument is a reference to a combo box, and index is the position of the item.

3.4 The Timer Control

Timer controls respond to the passage of time. They are independent of the user, and you can program them to take actions at regular intervals. A typical response is checking the system clock to see if it is time to perform some task. Timers also are useful for other kinds of background processing.

Each timer control has an Interval property that specifies the number of milliseconds that pass between one timer events to the next. Unless it is disabled, a timer continues to receive an event (appropriately named the Timer event) at roughly equal intervals of time.

The Interval property has a few limitations to consider when you're programming a timer control:

If your application or another application is making heavy demands on the system — such as long loops, intensive calculations, or drive, network, or port access your application may not get timer events as often as the Interval property specifies.

The interval can be between 0 and 64,767, inclusive, which means that even the longest interval can't be much longer than one minute (about 64.8 seconds).

The interval is not guaranteed to elapse exactly on time. To ensure accuracy, the timer should check the system clock when it needs to, rather than try to keep track of accumulated time internally.

The system generates 18 clock ticks per second so even though the Interval property is measured in milliseconds, the true precision of an interval is no more than one-eighteenth of a second.

Every timer control must be associated with a form. Therefore, to create a timer application, you must create at least one form (though you don't have to make the form visible if you don't need it for any other purpose).

Note The word "timer" is used in several ways in Visual Basic, each closely related to the workings of the timer control. In addition to the control name and control type, "timer" is used in the Timer event and the Timer function.

3.4.1 Placing a Timer Control on a Form

Placing a timer control on a form is like drawing any other control: Click the timer button in the toolbox and drag it onto a form.

The timer appears on the form at design time only so you can select it, view its properties, and write an event procedure for it. At run time, a timer is invisible and its position and size are irrelevant.

3.4.2 Initializing a Timer Control

A timer control has two key properties.

Property	Setting
Enabled	If you want the timer to start working as soon as the form loads, set it to True. Otherwise, leave this property set to False. You might choose to have an outside event (such as a click of a command button) start operation of the timer.
Interval	Number of milliseconds between timer events.

Note that the Enabled property for the timer is different from the Enabled property for other objects. With most objects, the Enabled property determines whether the object can respond to an event caused by the user. With the Timer control, setting Enabled to False suspends timer operation.

Remember that the Timer event is periodic. The Interval property doesn't determine "how long" as much as it determines "how often." The length of the interval should depend on how much precision you want. Because there is some built-in potential for error, make the interval one-half the desired amount of precision.

Note The more often a timer event is generated, the more processor time is used in responding to the event. This can slow down overall performance. Don't set a particularly small interval unless you need it.

3.5 Multiple-Document Interface (MDI) Applications

The multiple-document interface (MDI) allows you to create an application that maintains multiple forms within a single container form. Applications such as Microsoft Excel and Microsoft Word for Windows have multiple-document interfaces.

An MDI application allows the user to display multiple documents at the same time, with each document displayed in its own window. Documents or child windows are contained in a parent window, which provides a workspace for all the child windows in the application. For example, Microsoft Excel allows you to create and display multiple-document windows of different types. Each individual window is confined to the area of the Excel parent window. When you minimize Excel, all of the document windows are minimized as well; only the parent window's icon appears in the task bar.

A child form is an ordinary form that has its MDIChild property set to true. Your application can include many MDI child forms of similar or different types.

At run time, child forms are displayed within the workspace of the MDI parent form (the area inside the form's borders and below the title and menu bars). When a child form is minimized, its icon appears within the workspace of the MDI form instead of on the taskbar.

The application can also include standard, non-MDI forms that are not contained in the MDI form. A typical use of a standard form in an MDI application is to display a modal dialog box.

An MDI form is similar to an ordinary form with one restriction. Its not allowed to place a control directly on a MDI form unless that control has an Align property (such as a picture box control) or has no visible interface (such as a timer control).

In my program the customers and products forms are MDI Childs for the Main form.

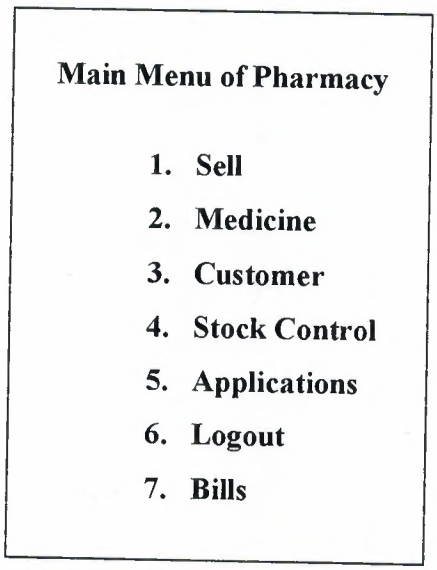
4. PROGRAM DESIGN PROCESS

4.1 Login Screen Form

Here the user enters the username and password to login to the program. If the information provided by the user is correct the main form will be opened else it will not be displayed. There is an option for the user to change the login information. When the change password button is pressed a form will be displayed for the user to change the password.

4.2 Main Menu Form

The aim of the main menu is to use the program easily, faster and use all the process screens or necessary program at the same time.



The aim of the main menu is to use the program easily, faster and use all the process screens or necessary program at the same time.

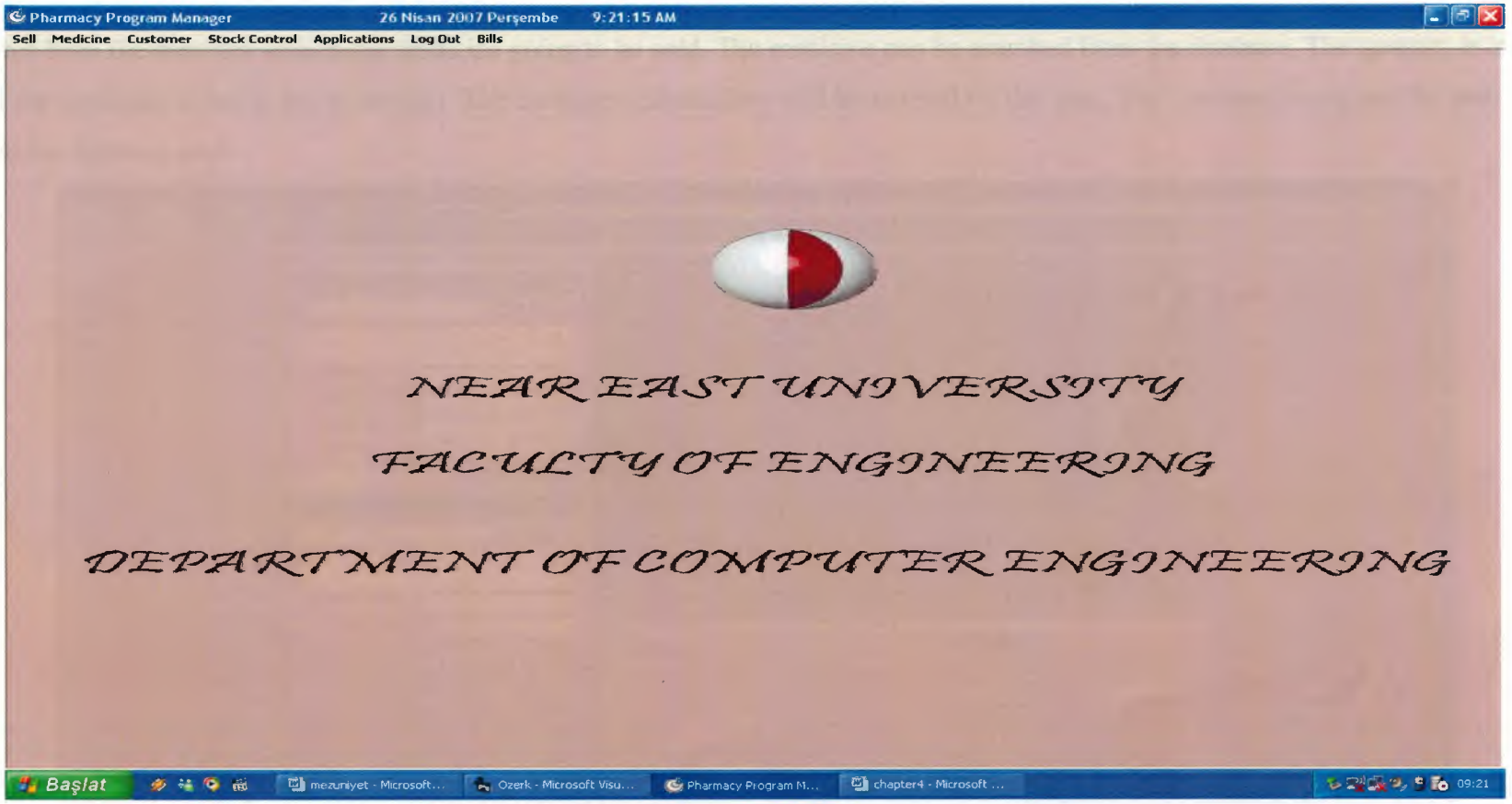


Figure 4.1: Main menu screenshot

4.2.1 Sell Form

In this form the user first selects the medicine going to be sold. The medicine can be searched from the database. The quantity is input and the medicine is being put to the cart. The customer information will be entered by the user. The customer name can be searched from the database also.

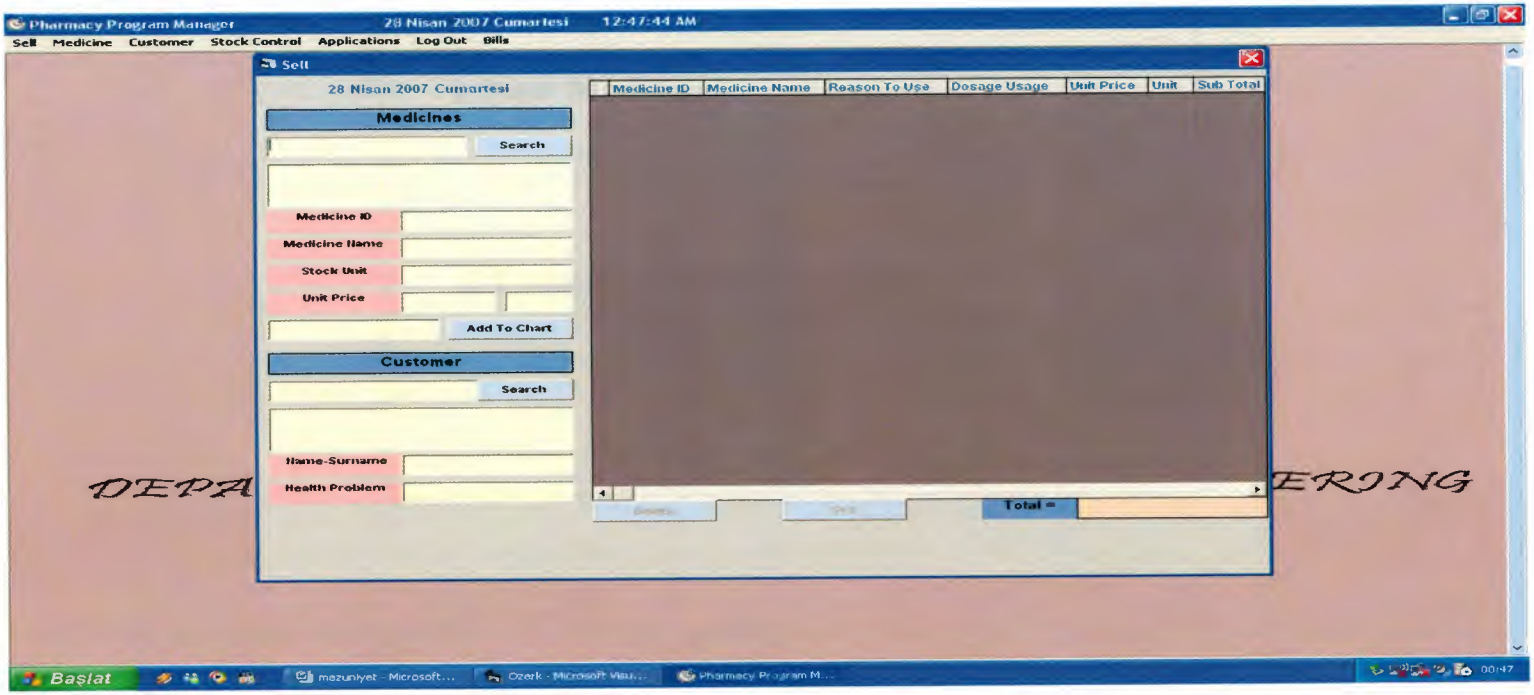
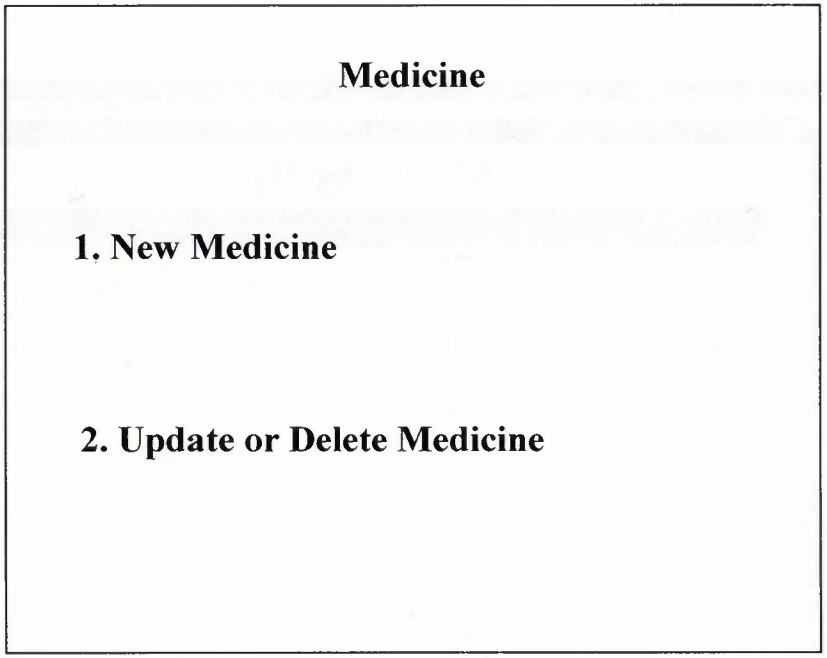


Figure 4.2: Sell form screenshot

4.2.2 Medicine Form



4.2.2.1 New Medicine Form

In this form of the program a new medicine can be entered to the pharmacy database. All the fields of the form should be filled. The expiration date is important for the program to remind you the medicines that is going to be expired.

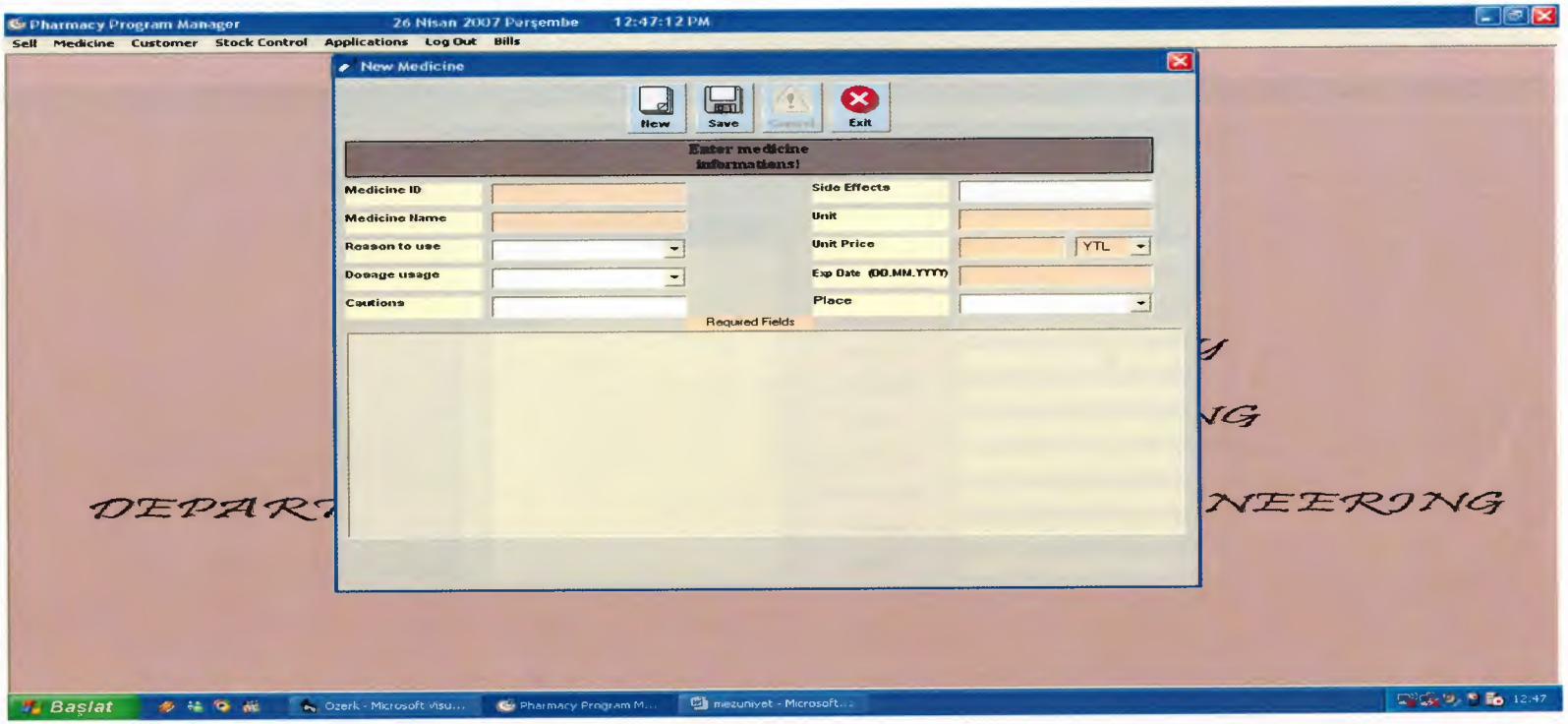


Figure 4.3: New medicine form screenshot

4.2.2.2 Update or Delete Medicine Form

In this form the medicines in the database are listed. The user selects the medicine from the list and the information belonging to that medicine is shown in the fields of the form.

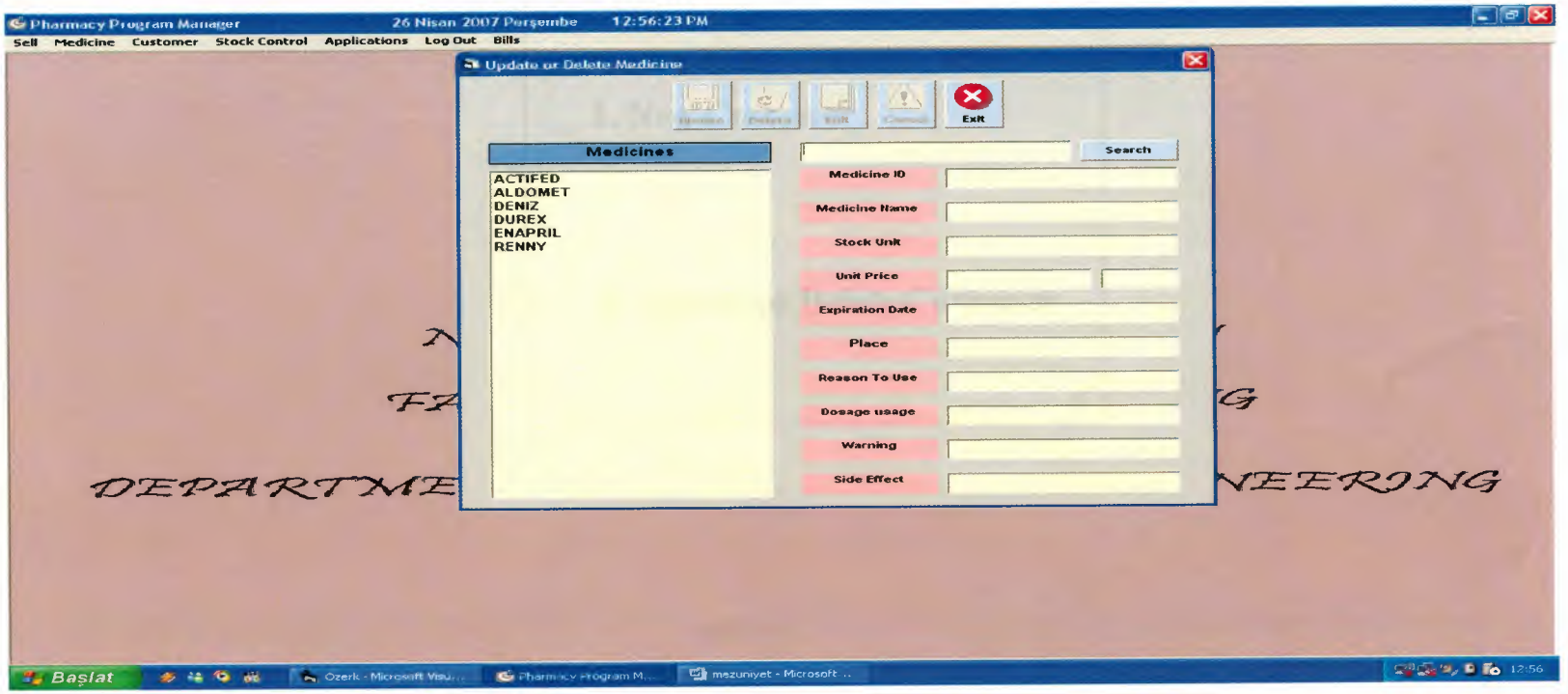
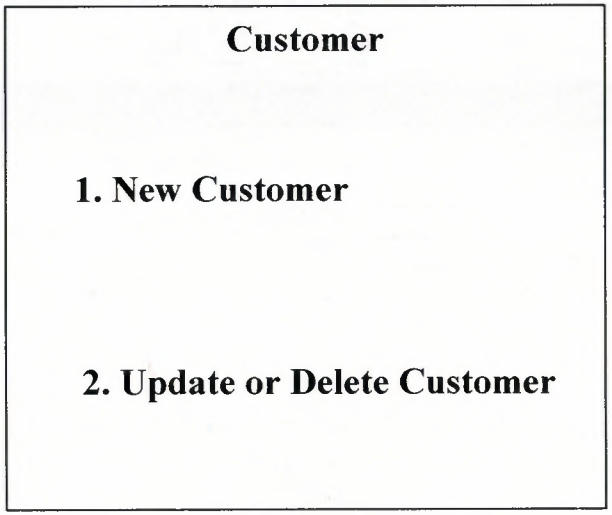


Figure 4.4: Update medicine form screenshot

4.2.3 Customer Form



4.2.3.1 New Customer Form

In this part of the program a new customer can be added to the database. All the fields should be filled.

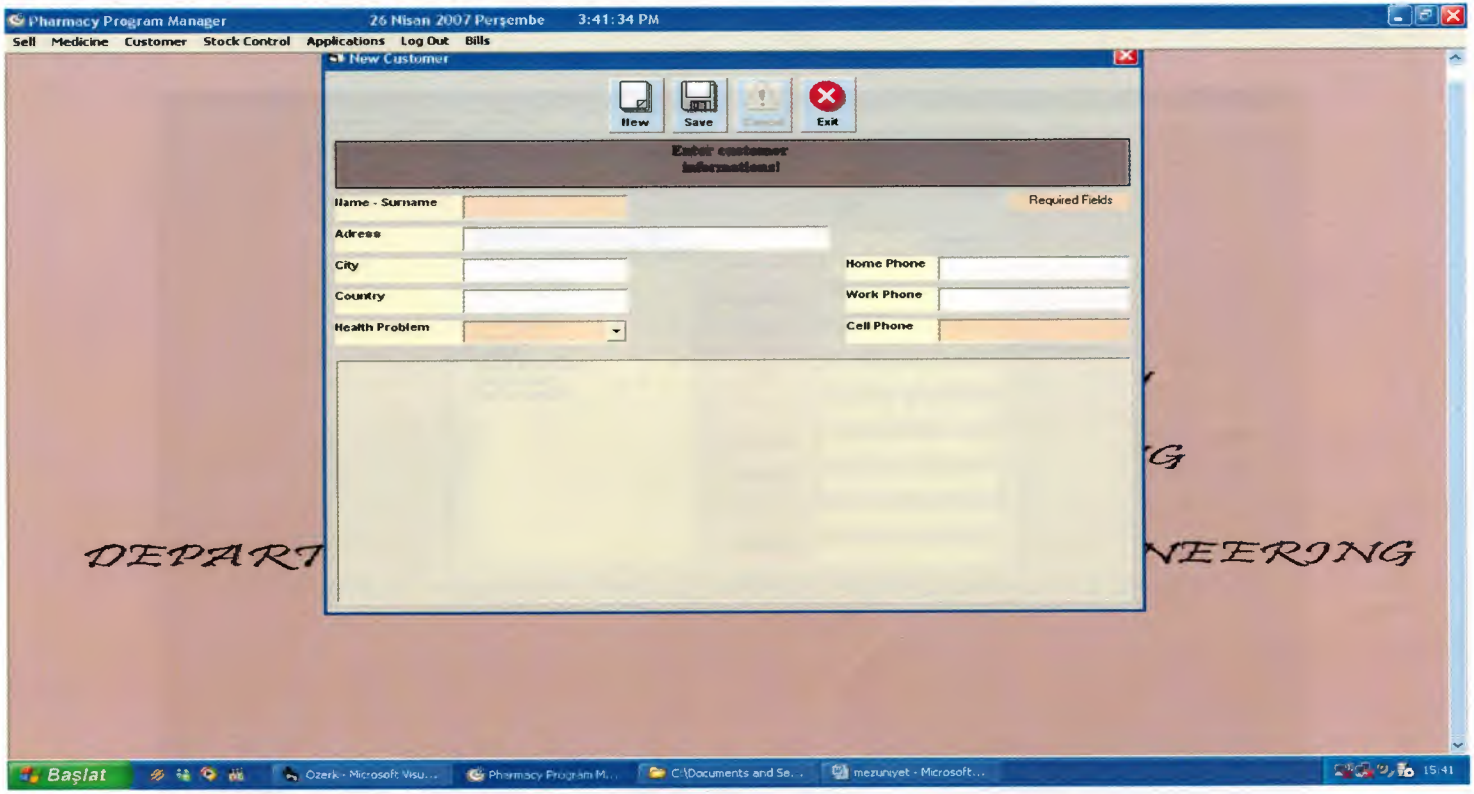


Figure 4.5: New customer form screenshot

4.2.3.2 Update Or Delete Customer Form

In this form a list of customers in the database are listed in a listbox. By double clicking on a customer the information of the customer is taken from the database and it is shown on the form fields. Then necessary update is done.

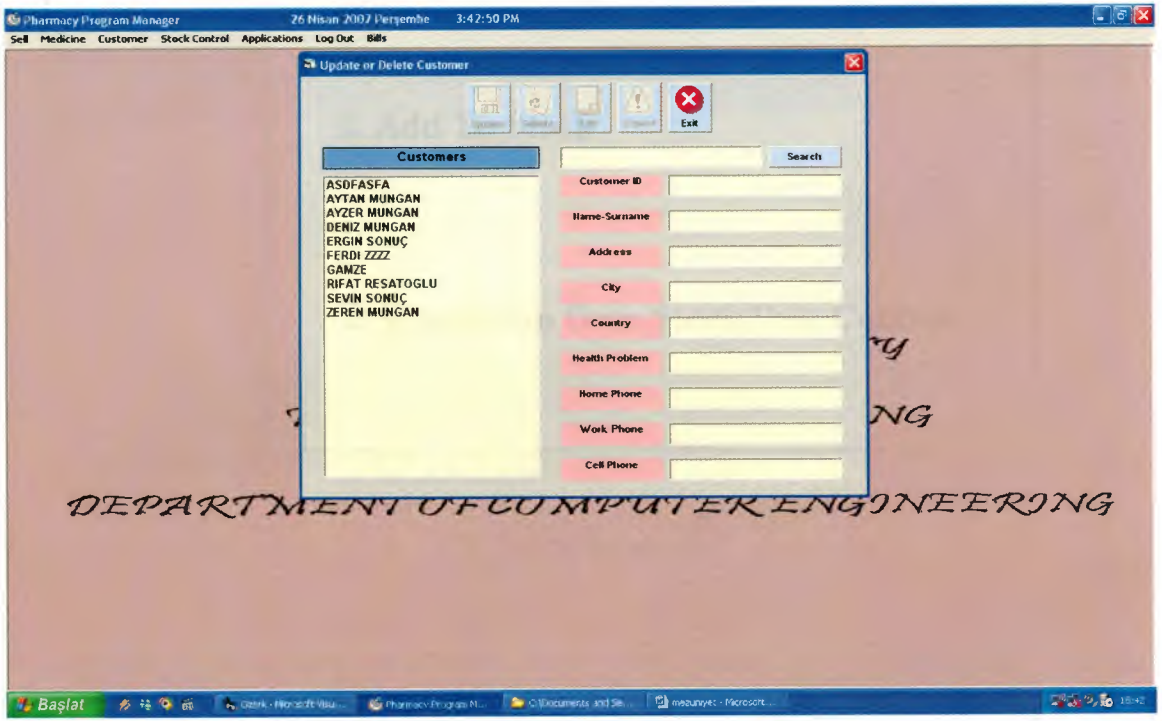
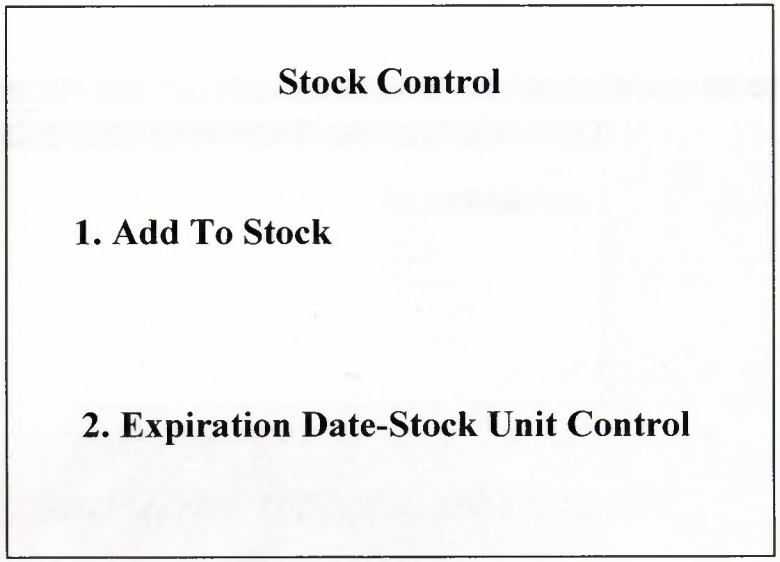


Figure 4.6: Update or delete customer form screenshot

4.2.4 Stock Control Form



4.2.4.1 Add To Stock Form

In this section of the form the incoming medicine can be added to the stock.

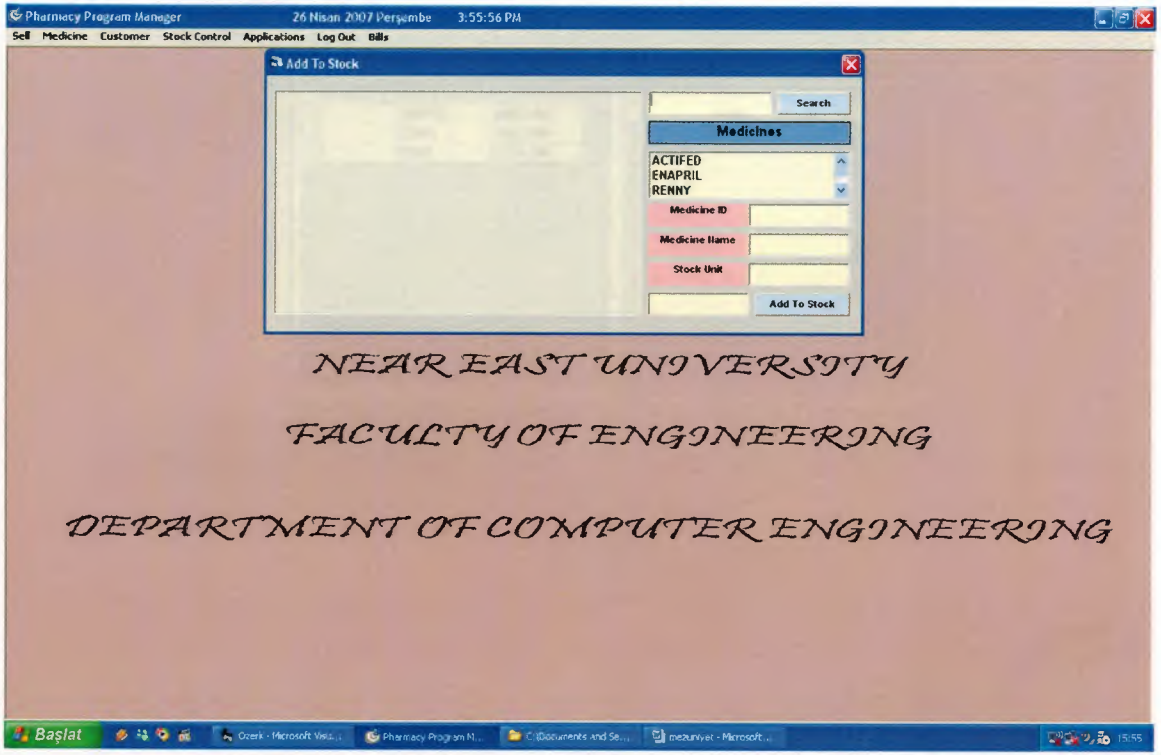


Figure 4.7: New medicine form screenshot

4.2.4.2 Expiration Date – Stock Control Form

In this form of the program the medicine which has critical stock number and the medicines nearly expired are reported.

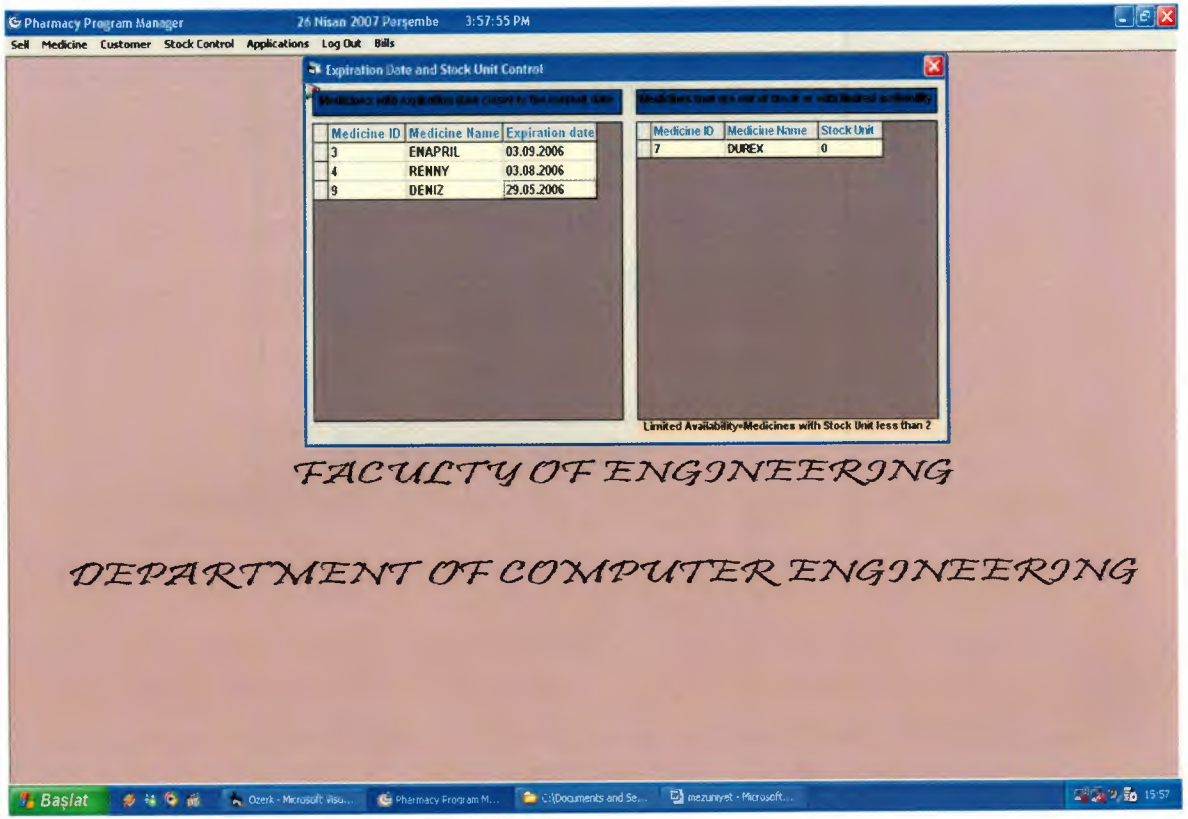
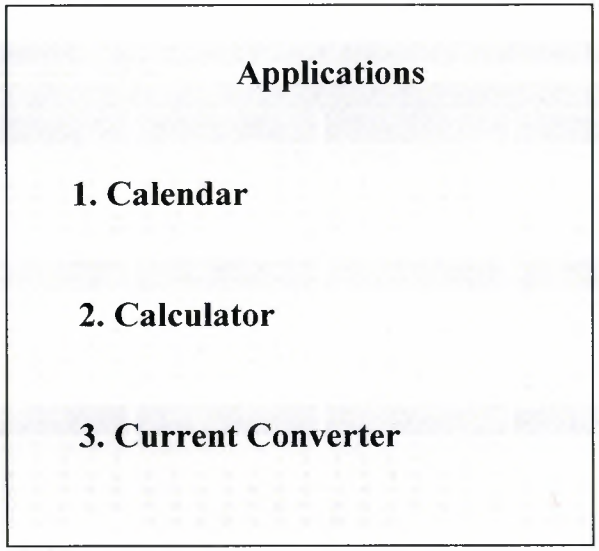


Figure 4.8: Expiration Date and Stock unit control form screenshot

4.2.5 Applications Form



4.2.5.1 Calendar Form

In this section of the program some windows programs can be launched such as calendar.

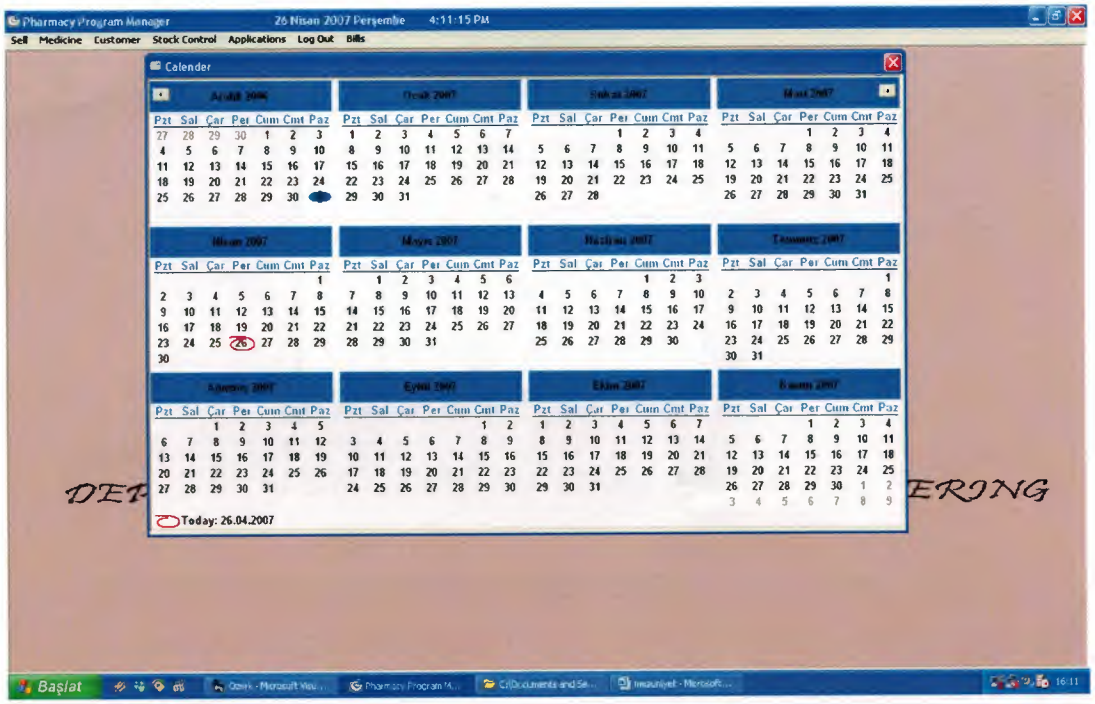


Figure 4.9: Calendar component shown on the form.

4.2.5.2 Calculator Form

In this section of the program some windows programs can be launched such as calculator.

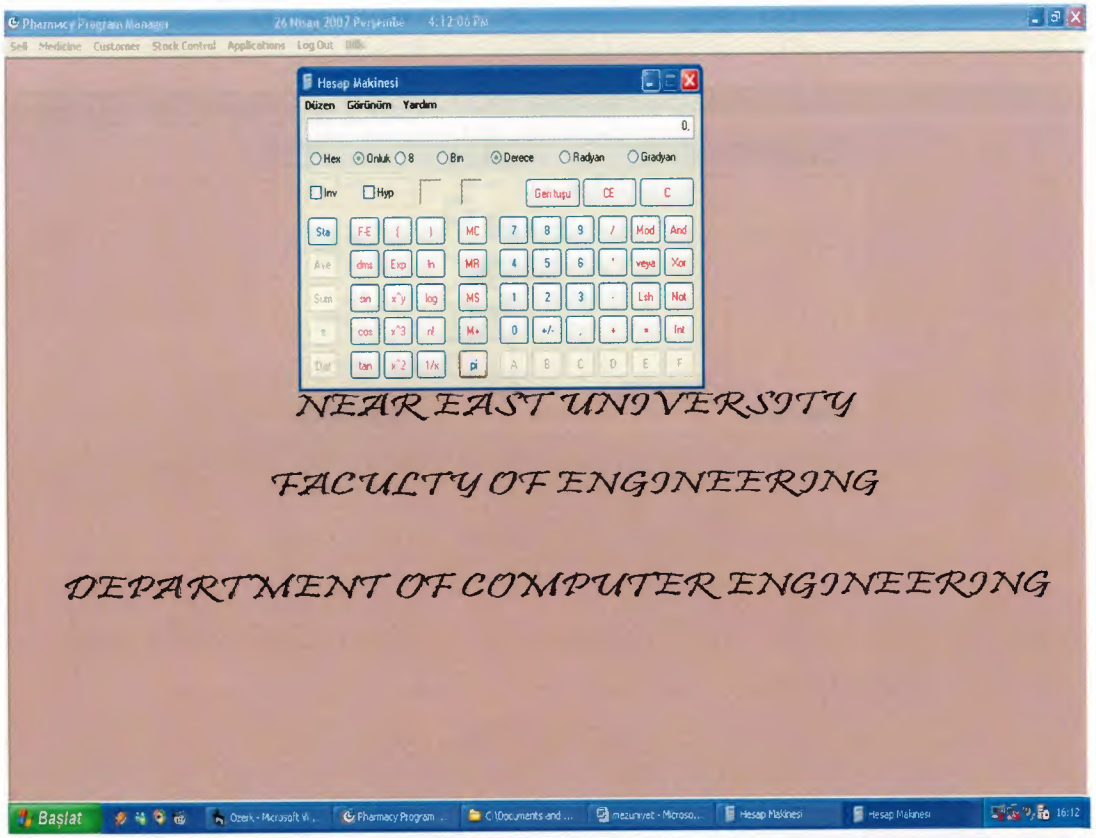


Figure 4.10: Calculator component shown on the form.

4.2.5.3 Currency Converter Form

In this section of the program some windows programs can be launched such as calendar.

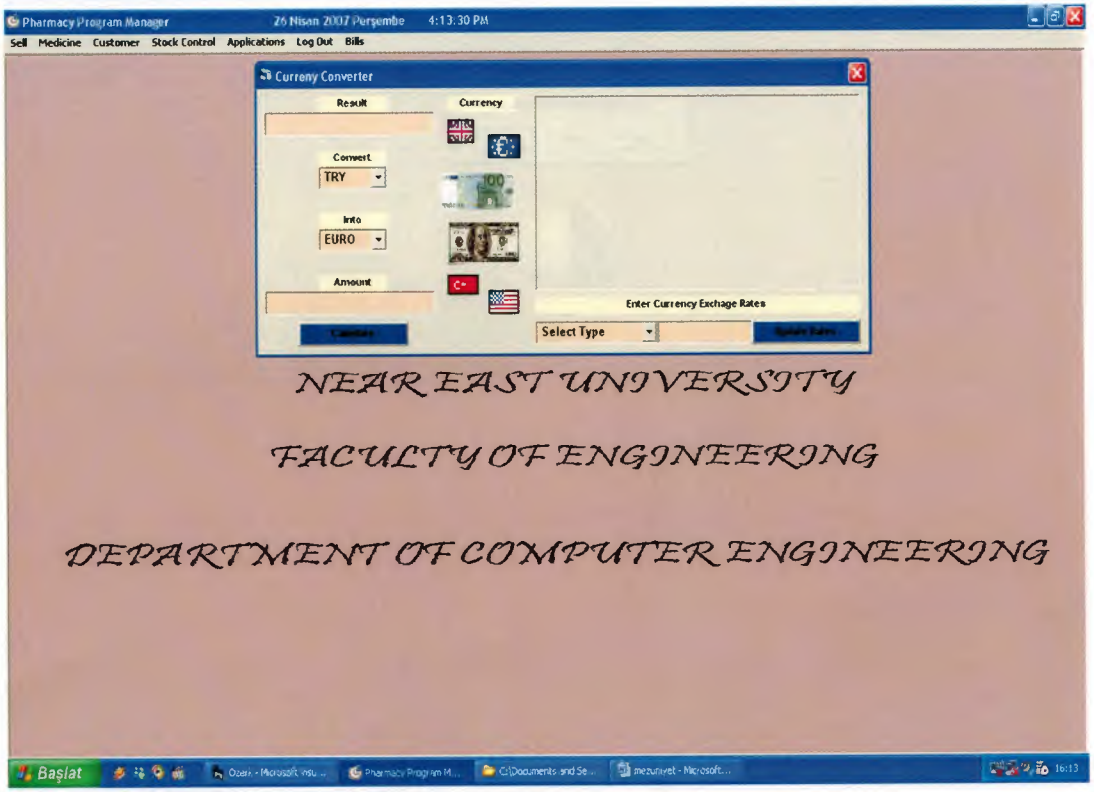


Figure 4.11: Currency converter form screenshot.

4.2.6 Log Out Form

You can exit from the program by clicking this menu item.

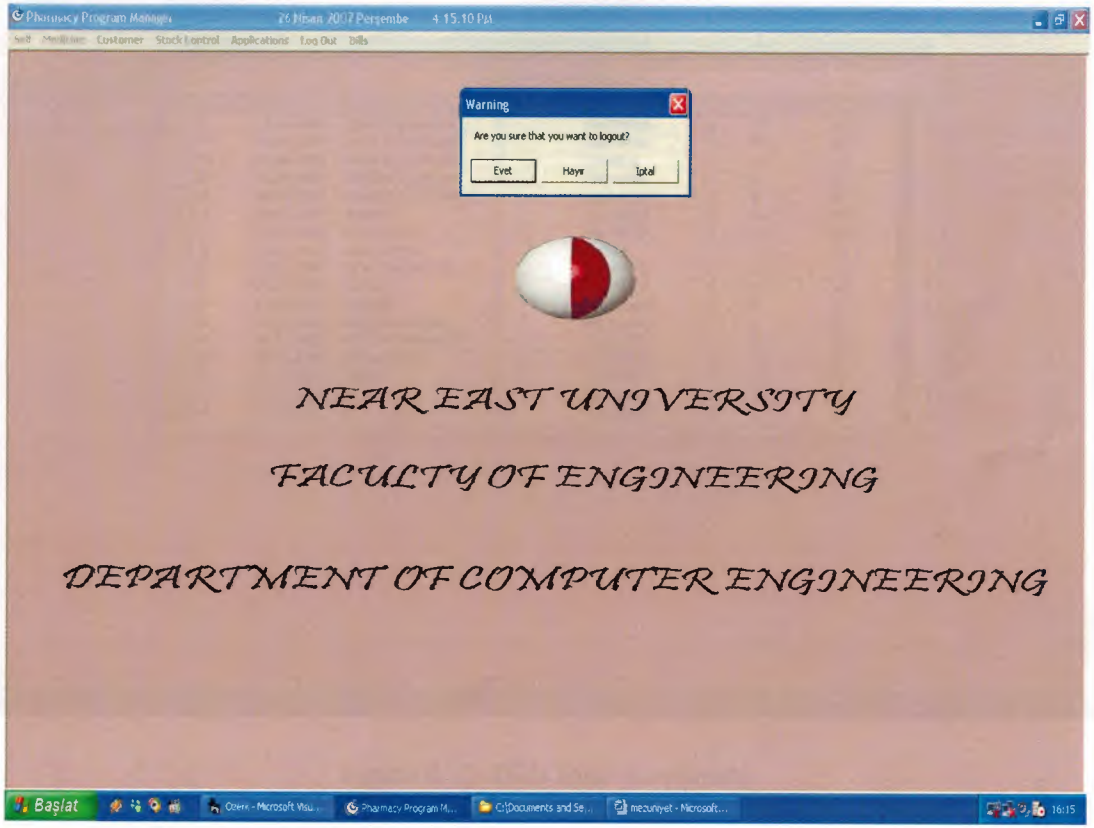


Figure 4.12: Logout.

4.2.7 Bills Form

In this section of the program we can see all customer who is selling from pharmacy.

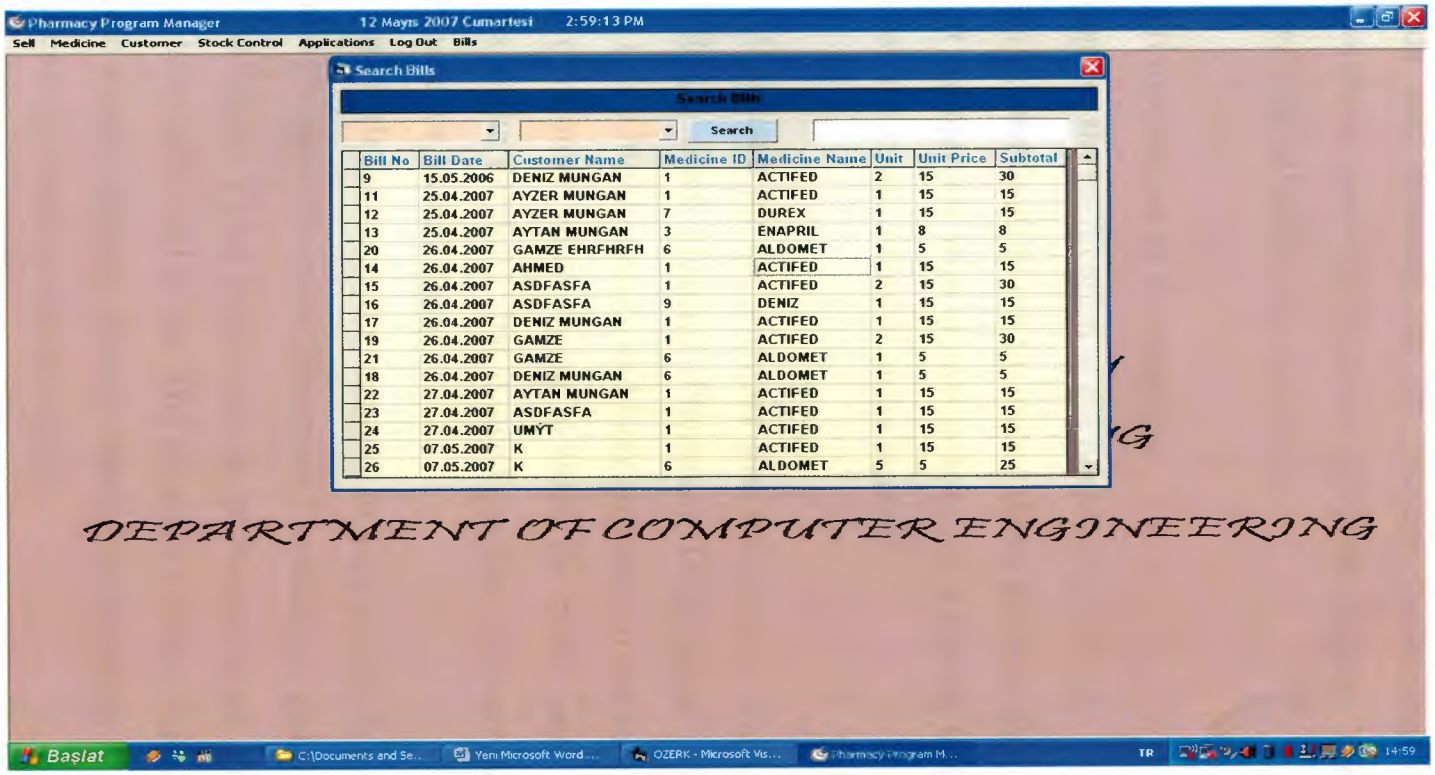


Figure 4.13: Bills form screenshot.

CONCLUSION

Particular needs and demands of Dr. Burhan Nalbantoglu State Hospital Pharmacy Department made this software program posible. The features of the program that the hospital needs are as follows:

- To sell medication to the patients,
- To register the medication at the hospital data base,
- To see the medication whose expiry dates are over due,
- To list the patients who use medication on a regular basis,
- To keep the stock quantity under control.

Visual Basic 6.0 is used for the software that will answer the needs of the Dr. Burhan Nalbantoglu State Hospital Pharmacy Department. This program is chosen because it is a highly secure and easy software to use in terms of both the user and the programmer when trying to fulfil the needs and demands of the hospital. Access is the database that is used for this software as it is the most applicable database for Visual Basic.

This software holds all the necessary aspects to support the heavy work load of Burhan Nalbantoglu State Hospital Pharmacy Department and it is being tested in the same department. There have been no problems faced with the program coping with the hospital's load so far. The program requires further development to handle remote access perhaps through internet to facilitate remote access by doctors and patients for better communications.

REFERENCES

1- H.M, P.J.Deitel and T.R.Nieto,Visual Basic 6:0 How To Program ,Prentice Hall,Inc .Upper Saddle River .New Jursey,1999

2- İhsan Karagulle and Zeydin Pala Microsoft Visual Basic 6:0 Pro Türkmen Printing House ,İstanbul ,2001

3- A research for finding Visual Basic code ,Finded November 01.2005 From the World Wide Web (www.vbturk.com).

4- A guide research for writing program .Retrieved October 10.2005 from the World Wide Web (www.programlama.com)

5- A Guide For writing about Visual Basic Description ,Retrieved Decenber 05.2005 from the World Wide web (www.vbtutor.net /lesson.html)

6- Lefkoşa Dr Burhan Nalbantoğlu Devlet Hastanesi Pharmacy Department.

APPENDIX

Login Form

```
Private Sub Combo1_Click()
Text2.Enabled = True
Text2.SetFocus
End Sub
Private Sub Combo1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text2.Enabled = True
Text2.SetFocus
End If
End Sub
Private Sub Command1_Click()
Dim response As Integer
Data1.Refresh
k = 0
Do While Not Data1.Recordset.EOF
If Data1.Recordset("userid") = Combo1.Text And Data1.Recordset("password") = Text2.Text Then
Form1.Hide
MDI.Show
k = 1
End If
Data1.Recordset.MoveNext
Loop
```

```
If k <> 1 Then
response = MsgBox("Wrong Information", vbNo, "Warning!")
If response = vbOK Then
Combo1.Text = ""
Text2.Text = ""
Command1.Enabled = False
Combo1.SetFocus
End If
End If
End Sub
Private Sub Command3_Click()
End
End Sub
Private Sub Command4_Click()
Form1.Hide
Form2.Show
End Sub
Private Sub Form_Load()
Combo1.Text = ""
Text2.Text = ""
Text2.Enabled = False
Data1.Refresh
Do While Not Data1.Recordset.EOF
Combo1.AddItem (Data1.Recordset("userid"))
Data1.Recordset.MoveNext
Loop
```

```
Command1.Enabled = False
Call ShowCurrTime
End Sub
Private Sub Label3_Click()
End Sub
Private Sub Text2_Change()
Command1.Enabled = True
End Sub
Private Sub text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call Command1_Click
End If
End Sub
Private Sub ShowCurrTime()
Text3.Text = Format$(Date, "dddddd") + "       " + Format$(Now, "h:mm:ss AM/PM")
End Sub
Private Sub Timer1_Timer()
ShowCurrTime
   'Update time display Call ShowCurrTime
End Sub
```

Change Pasword

```
Private Sub Combo1_Click()
Text4.Enabled = True
Text5.Enabled = True
```

```
Text4.SetFocus

End Sub

Private Sub Command4_Click()

Dim response As Integer

Dim respon As Integer

response = MsgBox("Are you sure that you want to change your password?",
vbYesNoCancel, "Warning")

If response = vbYes Then

c = 0

Data1.Refresh

Do While Not Data1.Recordset.EOF

If Data1.Recordset("userid") = Combo1.Text And Data1.Recordset("password") = Text4.Text
Then

c = 1

Data1.Recordset.Edit

Data1.Recordset("password") = Text5.Text

Data1.Recordset.Update

response = MsgBox("Your password is saved succesfully", vbNo, "Congragulations!")

If response = vbOK Then

Combo1.Text = ""

Text4.Text = ""

Text5.Text = ""

Text4.Enabled = False

Text5.Enabled = False

End If

End If
```

```
Data1.Recordset.MoveNext
Loop
If c <> 1 Then
respon = MsgBox("Wrong Information", vbNo, "Warning!")
Combo1.Text = ""
Text4.Text = ""
Text5.Text = ""
Text4.Enabled = False
Text5.Enabled = False
End If
End If
If response = vbNo Then
Combo1.Text = ""
Text4.Text = ""
Text5.Text = ""
Text4.Enabled = False
Text5.Enabled = False
End If
End Sub
Private Sub Command5_Click()
Form2.Hide
Form1.Show
End Sub
Private Sub Command6_Click()
Combo1.Text = ""
```

```
Text4.Text = ""
Text5.Text = ""
Command4.Enabled = False
End Sub
Private Sub Form_Load()
Data1.Refresh
Do While Not Data1.Recordset.EOF
Combo1.AddItem (Data1.Recordset("userid"))
Data1.Recordset.MoveNext
Loop
Text4.Enabled = False
Text5.Enabled = False
Command4.Enabled = False
End Sub
Private Sub Label4_Click()
End Sub
Private Sub text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text5.SetFocus
End If
End Sub
Private Sub Text5_Change()
Command4.Enabled = True
End Sub
Private Sub text5_KeyPress(KeyAscii As Integer)
```

```
If KeyAscii = 13 Then

Command4.Enabled = True

Call Command4_Click

End If

End Sub
```

Sell Form

```
Private Sub Text6_Change()

End Sub

Private Sub Text8_Change()

'customer

If Text8.Text = "" Then

List2.Clear

End If

If Text8.Text <> "" Then

If Asc(Left(Text8.Text, 1)) > 96 And Asc(Left(Text8.Text, 1)) < 123 Or
Asc(Left(Text8.Text, 1)) > 64 And Asc(Left(Text8.Text, 1)) < 133 Then

searchtextcus

Else

searchnumbercus

End If

End If

End Sub

Private Sub Text9_Change()
```

```
'medicine

If Text9.Text = "" Then

List1.Clear

End If

If Text9.Text <> "" Then

If Asc(Left(Text9.Text, 1)) > 96 And Asc(Left(Text9.Text, 1)) < 123 Or
Asc(Left(Text9.Text, 1)) > 64 And Asc(Left(Text9.Text, 1)) < 133 Then

searchtext

Else

searchnumber

End If

End If

End Sub

Function searchtext()

'medicine

Data1.Refresh

List1.Clear

j = Len(Text9.Text)

Do While Not Data1.Recordset.EOF

If UCase(Text9.Text) = UCase(Mid(Data1.Recordset("Name"), 1, j)) Then

List1.AddItem UCase(Data1.Recordset("Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")

End If

Data1.Recordset.MoveNext

Loop

End Function
```

```
Function searchnumber()
'medicine
Data1.Refresh
List1.Clear
Do While Not Data1.Recordset.EOF
If Text9.Text = Data1.Recordset("MedicineID") Then
List1.AddItem Data1.Recordset("MedicineID") + "    " + UCase(Data1.Recordset("Name"))
List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")
End If
Data1.Recordset.MoveNext
Loop
End Function
Function searchtextcus()
'customer
Data2.Refresh
List2.Clear
j = Len(Text8.Text)
Do While Not Data2.Recordset.EOF
If UCase(Text8.Text) = UCase(Mid(Data2.Recordset("Customer Name"), 1, j)) Then
List2.AddItem UCase(Data2.Recordset("Customer Name"))
List2.ItemData(List2.ListCount - 1) = Data2.Recordset("Customer ID")
End If
Data2.Recordset.MoveNext
Loop
End Function
```

```
Function searchnumbercus()

'customer

Data2.Refresh

List2.Clear

Do While Not Data2.Recordset.EOF

If Text8.Text = Data2.Recordset("Customer ID") Then

List2.AddItem Data2.Recordset("Customer ID") & "     " &
UCase(Data2.Recordset("Customer Name"))

List2.ItemData(List2.ListCount - 1) = Data2.Recordset("Customer ID")

End If

Data2.Recordset.MoveNext

Loop

End Function

Private Sub ShowCurrTime()

Text6.Text = Format$(Date, "dddddd")

End Sub

Function calculate()

Text12.Text = 0

For i = 1 To grid.Rows - 1

Text12.Text = CDbl(grid.TextMatrix(i, 7)) + Val(Text12.Text)

Next i

End Function

Function back()

Data4.Refresh

Do While Not Data4.Recordset.EOF

If Data4.Recordset("MedicineID") = grid.TextMatrix(grid.RowSel, 1) Then
```

```
Data4.Recordset.Edit

Data4.Recordset("StockUnit") = Data4.Recordset("StockUnit") +
Val(grid.TextMatrix(grid.RowSel, 6))

Data4.Recordset.Update

Data4.Refresh

Exit Do

End If

Data4.Recordset.MoveNext

Loop

End Function
```

New Medicine

```
Private Sub Command10_Click()

Dim response As Integer

response = MsgBox("Are you sure that you want to cancel?", vbYesNoCancel, "Warning")

If response = vbYes Then

Data1.Refresh

Data2.Refresh

Data3.Refresh

Data4.Refresh

Text1.Text = ""

Text2.Text = ""

Combo1.Text = ""

Combo2.Text = ""

Text3.Text = ""
```

```
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Combo3.Text = ""
End If
End Sub
Private Sub Command6_Click()
Command10.Enabled = False
Data1.Refresh
Data2.Refresh
Data3.Refresh
Data4.Refresh
Text1.Text = ""
Text2.Text = ""
Combo1.Text = ""
Combo2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Combo3.Text = ""
Text1.SetFocus
End Sub
```

```
Private Sub Command8_Click()
Dim response As Integer
Dim mydate, x As Date
mydate = Date
x = mydate
For i = 1 To Len(Text1.Text)
If Asc(Mid(Text1.Text, i, 1)) < 48 Or Asc(Mid(Text1.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the MedicineID correctly!", vbNo, "Attention!")
Text1.Text = ""
Text1.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text3.Text)
If Asc(Mid(Text3.Text, i, 1)) < 48 Or Asc(Mid(Text3.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the stock unit correctly!", vbNo, "Attention!")
Text3.Text = ""
Text3.SetFocus
Exit Sub
End If

Next i
For i = 1 To Len(Text4.Text)
If Asc(Mid(Text4.Text, i, 1)) < 44 Or Asc(Mid(Text4.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the unit price correctly!", vbNo, "Attention!")
```

```
Text4.Text = ""
Text4.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text4.Text)
   If Asc(Mid(Text4.Text, i, 1)) > 44 And Asc(Mid(Text4.Text, i, 1)) < 48 Then
   response = MsgBox("Please Enter the unit price correctly!", vbNo, "Attention!")
   Text4.Text = ""
   Text4.SetFocus
   Exit Sub
   End If
Next i
For i = 1 To Len(Text5.Text)
If Asc(Mid(Text5.Text, i, 1)) < 44 Or Asc(Mid(Text5.Text, i, 1)) > 57 Then
response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")
Text5.Text = ""
Text5.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text5.Text)
If Asc(Mid(Text5.Text, i, 1)) = 44 Or Asc(Mid(Text5.Text, i, 1)) = 45 Then
response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")
Text5.Text = ""
```

```
Text5.SetFocus

Exit Sub

End If

Next i

If Mid(Text5.Text, 3, 1) <> "." Or Mid(Text5.Text, 6, 1) <> "." Or Len(Text5.Text) <> 10
Then

response = MsgBox("Please Enter the date in DD.MM.YYYY format", vbNo, "Attention!")

Text5.Text = ""

Text5.SetFocus

Exit Sub

End If

If CDbl(Left(Text5.Text, 2)) > 31 Or CDbl(Left(Text5.Text, 2)) <= 0 Or
CDbl(Mid(Text5.Text, 4, 2)) > 12 Or CDbl(Mid(Text5.Text, 4, 2)) <= 0 Then

response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")

Text5.Text = ""

Text5.SetFocus

Exit Sub

End If

If CDate(Text5.Text) <= x Or CDate(Text5.Text) < x + 30 Then

response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")

Text5.Text = ""

Text5.SetFocus

Exit Sub

End If

c = 0

Data1.Refresh
```

```
If Text1.Text = "" Or Text2.Text = "" Or Text3.Text = "" Or Text4.Text = "" Or Text5.Text =
"" Or Combo4.Text = "" Then

response = MsgBox("Please fill all required fields!", vbNo, "Attention!")

Exit Sub

End If

response = MsgBox("Are you sure that you want to save?", vbYesNoCancel, "Warning")

If response = vbYes Then

Do While Not Data1.Recordset.EOF

If Text1.Text = Data1.Recordset("Medicine ID") Then

c = 1

response = MsgBox("Medicine already exists in database.If you want to update this item, goto
main menu and click 'update or delete medicine.'", vbNo, "Warning")

If response = vbOK Then

Call Command6_Click

End If

End If

Data1.Recordset.MoveNext

Loop

If c = 0 Then

Data1.Recordset.AddNew

Data1.Recordset("Medicine ID") = Text1.Text

Data1.Recordset("Medicine Name") = UCase(Text2.Text)

Data1.Recordset("Reason To Use") = UCase(Combo1.Text)

Data1.Recordset("Dosage") = UCase(Combo2.Text)

Data1.Recordset("Stock Unit") = Val(Text3.Text)

Data1.Recordset("Unit Price") = CDbl(Text4.Text)
```

```
Data1.Recordset("Currency") = Combo4.Text
Data1.Recordset("Expiration Date") = CDate(Text5.Text)
Data1.Recordset("Place") = UCase(Combo3.Text)
Data1.Recordset("Warning") = UCase(Text6.Text)
Data1.Recordset("Side Effects") = UCase(Text7.Text)
Data1.Recordset.Update
Data1.Refresh
response = MsgBox("Item is saved succesfully", vbNo, "Congragulations!")
If response = vbOK Then
Call Command6_Click
End If
End If
End If
End Sub
Private Sub Command9_Click()
Dim response As Integer
response = MsgBox("Are you sure that you want to exit?", vbYesNoCancel, "Warning")
If response = vbYes Then
Unload Form3
End If
End Sub
Private Sub Form_Load()
Data1.Refresh
Data2.Refresh
Data3.Refresh
```

```
Data4.Refresh
Command10.Enabled = False
Combo4.AddItem ("YTL")
Combo4.AddItem ("USD")
Combo4.AddItem ("GBP")
Combo4.AddItem ("EURO")
Combo4.Text = "YTL"
Do While Not Data2.Recordset.EOF
Combo1.AddItem (Data2.Recordset("reasontouse"))
Data2.Recordset.MoveNext
Loop
Do While Not Data3.Recordset.EOF
Combo2.AddItem (Data3.Recordset("dosage"))
Data3.Recordset.MoveNext
Loop
Do While Not Data4.Recordset.EOF
Combo3.AddItem (Data4.Recordset("place"))
Data4.Recordset.MoveNext
Loop
End Sub
Private Sub Text1_Change()
Command10.Enabled = True
End Sub
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
```

```
Text2.SetFocus
End If
End Sub
Private Sub text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text2.Text = UCase(Text2.Text)
Combo1.SetFocus
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text3.Text = UCase(Text3.Text)
Text4.SetFocus
End If
End Sub
Private Sub text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text5.SetFocus
End If
End Sub
Private Sub text6_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text6.Text = UCase(Text6.Text)
Text7.SetFocus
End If
```

```
End Sub
Private Sub text7_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text7.Text = UCase(Text7.Text)
Text3.SetFocus
End If
End Sub
Private Sub text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Combo3.SetFocus
End If
End Sub
Private Sub Combo1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Combo2.SetFocus
Combo1.Text = UCase(Combo1.Text)
End If
End Sub
Private Sub Combo2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Combo2.Text = UCase(Combo2.Text)
Text6.SetFocus
End If
End Sub
Private Sub Combo3_KeyPress(KeyAscii As Integer)
```

```
If KeyAscii = 13 Then
Combo3.Text = UCase(Combo3.Text)
Call Command8_Click
End If
End Sub
Private Sub Combo4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text5.SetFocus
End If
End Sub
```

Update Or Delete Medicine

```
Private Sub Command3_Click()
Command5.Enabled = True
Text2.Locked = False
Text3.Locked = False
Text4.Locked = False
Text5.Locked = False
Text6.Locked = False
Text7.Locked = False
Text8.Locked = False
Text11.Locked = False
Text12.Locked = False
Text13.Locked = False
```

```
Text2.SetFocus
End Sub
Private Sub Command5_Click()
Dim response As Integer
For i = 1 To Len(Text5.Text)
If Asc(Mid(Text5.Text, i, 1)) < 48 Or Asc(Mid(Text5.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the stock unit value correctly!", vbNo, "Attention!")
Text5.Text = ""
Text5.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text6.Text)
If Asc(Mid(Text6.Text, i, 1)) < 44 Or Asc(Mid(Text6.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the unit price correctly!", vbNo, "Attention!")
Text6.Text = ""
Text6.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text6.Text)
If Asc(Mid(Text6.Text, i, 1)) > 44 And Asc(Mid(Text6.Text, i, 1)) < 48 Then
response = MsgBox("Please Enter the unit price correctly!", vbNo, "Attention!")
Text6.Text = ""
Text6.SetFocus
```

```
Exit Sub
End If
Next i
For i = 1 To Len(Text7.Text)
If Asc(Mid(Text7.Text, i, 1)) < 44 Or Asc(Mid(Text7.Text, i, 1)) > 57 Then
response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")
Text7.Text = ""
Text7.SetFocus
Exit Sub
End If
Next i
For i = 1 To Len(Text7.Text)
If Asc(Mid(Text7.Text, i, 1)) = 44 Or Asc(Mid(Text7.Text, i, 1)) = 45 Then
response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")
Text7.Text = ""
Text7.SetFocus
Exit Sub
End If
Next i
If Mid(Text7.Text, 3, 1) <> "." Or Mid(Text7.Text, 6, 1) <> "." Or Len(Text7.Text) <> 10
Then
response = MsgBox("Please Enter the date in DD.MM.YYYY format", vbNo, "Attention!")
Text7.Text = ""
Text7.SetFocus
Exit Sub
End If
```

```
If CDbl(Val(Left(Text7.Text, 2))) > 31 Or CDbl(Val(Left(Text7.Text, 2))) <= 0 Or
CDbl(Val(Mid(Text7.Text, 4, 2))) > 12 Or CDbl(Val(Mid(Text7.Text, 4, 2))) <= 0 Then

response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")

Text7.Text = ""

Text7.SetFocus

Exit Sub

End If

Dim mydate, x As Date

mydate = Date

x = mydate

If CDate(Text7.Text) <= x Or CDate(Text7.Text) < x + 30 Then

response = MsgBox("Expiration date is not acceptable", vbNo, "Attention!")

Text7.Text = ""

Text7.SetFocus

Exit Sub

End If

response = MsgBox("Are you sure that you want to save?", vbYesNoCancel, "Warning")

c = 0

If response = vbYes Then

Data1.Refresh

Do While Not Data1.Recordset.EOF

If Data1.Recordset("MedicineID") = Val(Text1.Text) Then

Data1.Recordset.Edit

Data1.Recordset("Name") = UCase(Text2.Text)

Data1.Recordset("ReasonToUse") = UCase(Text3.Text)

Data1.Recordset("Dosage") = UCase(Text4.Text)
```

```
Data1.Recordset("StockUnit") = Val(Text5.Text)

Data1.Recordset("UnitPrice") = CDbl(Text6.Text)

Data1.Recordset("ExpDate") = CDate(Text7.Text)

Data1.Recordset("Place") = UCase(Text8.Text)

Data1.Recordset("Currency") = UCase(Text11.Text)

Data1.Recordset("Warning") = UCase(Text12.Text)

Data1.Recordset("SideEffects") = UCase(Text13.Text)

Data1.Recordset.Update

Data1.Refresh

c = 1

Exit Do

End If

Data1.Recordset.MoveNext

Loop

End If

If c = 1 Then

   response = MsgBox("Selected Medicine is updated succesfully", vbNo,
"Congragulations!")

      If response = vbOK Then

      List1.Clear

      Call Form_Load

      End If

End If

Text9.SetFocus

End Sub

Private Sub Command6_Click()
```

```
Dim response As Integer
If Text9.Text = "" Then
response = MsgBox("Search field is empty!", vbNo, "Attention!")
Text9.SetFocus
Exit Sub
End If
Call Text9_Change
End Sub
Private Sub Command1_Click()
Dim response As Integer
response = MsgBox("Are you sure that you want to cancel?", vbYesNoCancel, "Warning")
If response = vbYes Then
Text9.SetFocus
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text11.Text = ""
Text12.Text = ""
Text13.Text = ""
```

```
End If

End Sub

Private Sub Command2_Click()

Dim response As Integer

response = MsgBox("Are you sure that you want to exit?", vbYesNoCancel, "Warning")

If response = vbYes Then

Unload Form6

End If

End Sub

Private Sub Command4_Click()

Dim response As Integer

response = MsgBox("Are you sure that you want to delete this record?", vbYesNoCancel,
"Warning")

If response = vbYes Then

Data1.Refresh

Do While Not Data1.Recordset.EOF

If Data1.Recordset("MedicineID") = Text1.Text Then

Data1.Recordset.Delete

End If

Data1.Recordset.MoveNext

Loop

List1.Clear

Call Form_Load

Data1.Refresh

response = MsgBox("Selected Medicine is deleted succesfully", vbNo, "Congragulations!")

End If
```

```
Text9.SetFocus
End Sub
Private Sub Form_Load()
Command1.Enabled = False
Command4.Enabled = False
Command5.Enabled = False
Command3.Enabled = False
Data1.Refresh
Do While Not Data1.Recordset.EOF
List1.AddItem UCase(Data1.Recordset("Name"))
List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")
Data1.Recordset.MoveNext
Loop
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text12.Text = ""
Text13.Text = ""
End Sub
```

```
Private Sub List1_dblClick()
Dim sql As String
Command4.Enabled = True
Command1.Enabled = True
Command3.Enabled = True
Data1.Refresh
Do While Not Data1.Recordset.EOF
If List1.ItemData(List1.ListIndex) = Data1.Recordset("MedicineID") Then
Text1.Text = UCase(Data1.Recordset("MedicineID"))
Text2.Text = UCase(Data1.Recordset("Name"))
Text3.Text = UCase(Data1.Recordset("ReasonToUse"))
Text4.Text = UCase(Data1.Recordset("Dosage"))
Text5.Text = UCase((Data1.Recordset("StockUnit")))
Text6.Text = UCase(Data1.Recordset("UnitPrice"))
Text7.Text = UCase(Data1.Recordset("ExpDate"))
Text8.Text = UCase(Data1.Recordset("Place"))
Text11.Text = UCase(Data1.Recordset("Currency"))
Text12.Text = Data1.Recordset("Warning")
Text13.Text = Data1.Recordset("SideEffects")
End If
Data1.Recordset.MoveNext
Loop
Text1.Locked = True
Text2.Locked = True
Text3.Locked = True
```

```
Text4.Locked = True
Text5.Locked = True
Text6.Locked = True
Text7.Locked = True
Text8.Locked = True
Text11.Locked = True
Text12.Locked = True
Text13.Locked = True
Text9.SetFocus
End Sub
Private Sub List1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call List1_dblClick
End If
End Sub
Private Sub text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text2.Text = UCase(Text2.Text)
Text5.SetFocus
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text3.Text = UCase(Text3.Text)
Text4.SetFocus
```

```
End If
End Sub
Private Sub text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text4.Text = UCase(Text4.Text)
Text12.SetFocus
End If
End Sub
Private Sub text12_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text12.Text = UCase(Text12.Text)
Text13.SetFocus
End If
End Sub
Private Sub text13_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text13.Text = UCase(Text13.Text)
Text2.SetFocus
End If
End Sub
Private Sub text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text5.Text = UCase(Text5.Text)
Text6.SetFocus
End If
```

```
End Sub
Private Sub text6_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text6.Text = UCase(Text6.Text)
Text11.SetFocus
End If
End Sub
Private Sub text7_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text7.Text = UCase(Text7.Text)
Text8.SetFocus
End If
End Sub
Private Sub text8_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text8.Text = UCase(Text8.Text)
Text3.SetFocus
End If
End Sub
Private Sub Text11_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text11.Text = UCase(Text11.Text)
Text7.SetFocus
End If
End Sub
```

```
Private Sub Text9_Change()

If Text9.Text = "" Then

List1.Clear

End If

If Text9.Text <> "" Then

If Asc(Left(Text9.Text, 1)) > 96 And Asc(Left(Text9.Text, 1)) < 123 Or
Asc(Left(Text9.Text, 1)) > 64 And Asc(Left(Text9.Text, 1)) < 133 Then

searchtext

Else

searchnumber

End If

End If

End Sub

Function searchtext()

Data1.Refresh

List1.Clear

j = Len(Text9.Text)

Do While Not Data1.Recordset.EOF

If UCase(Text9.Text) = UCase(Mid(Data1.Recordset("Name"), 1, j)) Then

List1.AddItem UCase(Data1.Recordset("Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")

End If

Data1.Recordset.MoveNext

Loop

End Function

Function searchnumber()
```

```
Data1.Refresh
List1.Clear
Do While Not Data1.Recordset.EOF
If Text9.Text = Data1.Recordset("MedicineID") Then
List1.AddItem Data1.Recordset("MedicineID") + "   " + UCase(Data1.Recordset("Name"))
List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")
End If
Data1.Recordset.MoveNext
Loop
End Function
Private Sub text9_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call Command6_Click
End If End Sub
```

New customer

```
Private Sub Command1_Click()
Dim response As Integer
response = MsgBox("Are you sure that you want to cancel?", vbYesNoCancel, "Warning")
If response = vbYes Then
Data1.Refresh
Data2.Refresh
Text2.Text = ""
Combo1.Text = ""
Text3.Text = ""
```

```
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
End If
End Sub
Private Sub Command2_Click()
Dim response As Integer
response = MsgBox("Are you sure that you want to exit?", vbYesNoCancel, "Warning")
If response = vbYes Then
Unload Form4
End If
End Sub
Private Sub Command3_Click()
Data1.Refresh
Dim response As Integer
If Text2.Text = "" Or Combo1.Text = "" Or Text8.Text = "" Then
response = MsgBox("Please fill all required fields!", vbNo, "Attention!")
Exit Sub
End If
response = MsgBox("Are you sure that you want to save?", vbYesNoCancel, "Warning")
If response = vbYes Then
Data1.Recordset.AddNew
Data1.Recordset("Customer Name") = UCase(Text2.Text)
```

```
Data1.Recordset("Address") = UCase(Text3.Text)
Data1.Recordset("City") = UCase(Text4.Text)
Data1.Recordset("Country") = UCase(Text5.Text)
Data1.Recordset("Home Phone") = UCase(Text6.Text)
Data1.Recordset("Work Phone") = UCase(Text7.Text)
Data1.Recordset("Cell Phone") = UCase(Text8.Text)
Data1.Recordset("Health Problem") = UCase(Combo1.Text)
Data1.Recordset.Update
Data1.Refresh
response = MsgBox("New Customer is saved succesfully", vbNo, "Congragulations!")
If response = vbOK Then
Call Command5_Click
End If
End If
End Sub
Private Sub Command5_Click()
Data1.Refresh
Data2.Refresh
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
```

```
Combo1.Text = ""
Text2.SetFocus
Command1.Enabled = False
End Sub
Private Sub Text2_Change()
Command1.Enabled = True
End Sub
Private Sub Form_Load()
Data1.Refresh
Data2.Refresh
Command1.Enabled = False
Do While Not Data2.Recordset.EOF
Combo1.AddItem (Data2.Recordset("healthproblem"))
Data2.Recordset.MoveNext
Loop
End Sub
Private Sub text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text2.Text = UCase(Text2.Text)
Text3.SetFocus
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text3.Text = UCase(Text3.Text)
```

```
Text4.SetFocus
End If
End Sub
Private Sub text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text4.Text = UCase(Text4.Text)
Text5.SetFocus
End If
End Sub
Private Sub text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text5.Text = UCase(Text5.Text)
Combo1.SetFocus
End If
End Sub
Private Sub Combo1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Combo1.Text = UCase(Combo1.Text)
Text6.SetFocus
End If
End Sub
Private Sub text6_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text6.Text = UCase(Text6.Text)
Text7.SetFocus
```

```
End If

End Sub

Private Sub text7_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then

Text7.Text = UCase(Text7.Text)

Text8.SetFocus

End If

End Sub

Private Sub text8_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then

Text8.Text = UCase(Text8.Text)

Call Command3_Click

End If

End Sub
```

Update Or Delete Customer

```
Private Sub Command1_Click()

Dim response As Integer

response = MsgBox("Are you sure that you want to cancel?", vbYesNoCancel, "Warning")

If response = vbYes Then
```

```
Text9.SetFocus
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
End If
End Sub
Private Sub Command2_Click()
Dim response As Integer
response = MsgBox("Are you sure that you want to exit?", vbYesNoCancel, "Warning")
If response = vbYes Then
Unload Form7
End If
End Sub
Private Sub Command3_Click()
Command5.Enabled = True
Text2.Locked = False
Text3.Locked = False
Text4.Locked = False
Text5.Locked = False
```

```
Text6.Locked = False

Text7.Locked = False

Text8.Locked = False

Text9.Locked = False

Text2.SetFocus

End Sub

Private Sub Command4_Click()

Dim response As Integer

response = MsgBox("Are you sure that you want to delete this record?", vbYesNoCancel,
"Warning")

If response = vbYes Then

Data1.Refresh

Do While Not Data1.Recordset.EOF

If Data1.Recordset("Customer ID") = Text1.Text Then

Data1.Recordset.Delete

End If

Data1.Recordset.MoveNext

Loop

List1.Clear

Call Form_Load

Data1.Refresh

response = MsgBox("Selected Customer is deleted succesfully", vbNo, "Congragulations!")

End If

Text10.SetFocus

End Sub

Private Sub Command5_Click()
```

```
Dim response As Integer
response = MsgBox("Are you sure that you want to save?", vbYesNoCancel, "Warning")
c = 0
If response = vbYes Then
Data1.Refresh
Do While Not Data1.Recordset.EOF
If Data1.Recordset("Customer ID") = Val(Text1.Text) Then
Data1.Recordset.Edit
Data1.Recordset("Customer Name") = UCase(Text2.Text)
Data1.Recordset("Address") = UCase(Text3.Text)
Data1.Recordset("City") = UCase(Text4.Text)
Data1.Recordset("Country") = UCase(Text5.Text)
Data1.Recordset("Health Problem") = UCase(Text6.Text)
Data1.Recordset("Home Phone") = UCase(Text7.Text)
Data1.Recordset("Work Phone") = UCase(Text8.Text)
Data1.Recordset("Cell Phone") = UCase(Text9.Text)
Data1.Recordset.Update
Data1.Refresh
c = 1
Exit Do
End If
Data1.Recordset.MoveNext
Loop
End If
If c = 1 Then
```

```
    response = MsgBox("Selected Customer is updated succesfully", vbNo,
"Congragulations!")

        If response = vbOK Then

        List1.Clear

        Call Form_Load

        End If

End If

Text10.SetFocus

End Sub

Private Sub Command6_Click()

Dim response As Integer

If Text10.Text = "" Then

response = MsgBox("Search field is empty!", vbNo, "Attention!")

Text10.SetFocus

Exit Sub

End If

Call Text10_Change

End Sub

Private Sub Text10_Change()

If Text10.Text = "" Then

List1.Clear

End If

If Text10.Text <> "" Then

If Asc(Left(Text9.Text, 1)) > 96 And Asc(Left(Text9.Text, 1)) < 123 Or
Asc(Left(Text9.Text, 1)) > 64 And Asc(Left(Text9.Text, 1)) < 133 Then

searchtext
```

```
Else
searchnumber
End If
End If
End Sub
Private Sub text2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text2.Text = UCase(Text2.Text)
Text3.SetFocus
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text3.Text = UCase(Text3.Text)
Text4.SetFocus
End If
End Sub
Private Sub text4_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text4.Text = UCase(Text4.Text)
Text5.SetFocus
End If
End Sub
Private Sub text5_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
```

```
Text5.Text = UCase(Text5.Text)
Text6.SetFocus
End If
End Sub
Private Sub text6_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text6.Text = UCase(Text6.Text)
Text7.SetFocus
End If
End Sub
Private Sub text7_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text7.Text = UCase(Text7.Text)
Text8.SetFocus
End If
End Sub
Private Sub text8_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text8.Text = UCase(Text8.Text)
Text9.SetFocus
End If
End Sub
Private Sub text9_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text9.Text = UCase(Text9.Text)
```

```
Text10.SetFocus
End If
End Sub
Private Sub Form_Load()
Command1.Enabled = False
Command4.Enabled = False
Command5.Enabled = False
Command3.Enabled = False
Data1.Refresh
Do While Not Data1.Recordset.EOF
List1.AddItem UCase(Data1.Recordset("Customer Name"))
List1.ItemData(List1.ListCount - 1) = Data1.Recordset("Customer ID")
Data1.Recordset.MoveNext
Loop
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
End Sub
```

```
Private Sub List1_dblClick()
Command4.Enabled = True
Command1.Enabled = True
Command3.Enabled = True
Data1.Refresh
Do While Not Data1.Recordset.EOF
If List1.ItemData(List1.ListIndex) = Data1.Recordset("Customer ID") Then
Text1.Text = UCase(Data1.Recordset("Customer ID"))
Text2.Text = UCase(Data1.Recordset("Customer Name"))
Text3.Text = UCase(Data1.Recordset("Address"))
Text4.Text = UCase(Data1.Recordset("City"))
Text5.Text = UCase(Data1.Recordset("Country"))
Text6.Text = UCase(Data1.Recordset("Health Problem"))
Text7.Text = UCase(Data1.Recordset("Home Phone"))
Text8.Text = UCase(Data1.Recordset("Work Phone"))
Text9.Text = UCase(Data1.Recordset("Cell Phone"))
End If
Data1.Recordset.MoveNext
Loop
Text1.Locked = True
Text2.Locked = True
Text3.Locked = True
Text4.Locked = True
Text5.Locked = True
Text6.Locked = True
```

```
Text7.Locked = True

Text8.Locked = True

Text9.Locked = True

Text10.SetFocus

End Sub

Function searchtext()

Data1.Refresh

List1.Clear

j = Len(Text10.Text)

Do While Not Data1.Recordset.EOF

If UCase(Text10.Text) = UCase(Mid(Data1.Recordset("Customer Name"), 1, j)) Then

List1.AddItem UCase(Data1.Recordset("Customer Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("Customer ID")

End If

Data1.Recordset.MoveNext

Loop

End Function

Function searchnumber()

Data1.Refresh

List1.Clear

Do While Not Data1.Recordset.EOF

If Text10.Text = Data1.Recordset("Customer ID") Then

List1.AddItem Data1.Recordset("Customer ID") & "     " &
UCase(Data1.Recordset("Customer Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("Customer ID")

End If
```

```
Data1.Recordset.MoveNext

Loop

End Function
```

Expiration And Date Stock Unit Control

```
Dim db As Database

Private Sub Form_Load()

'Expiration Date

Dim strSQL As String

Set db = OpenDatabase("pharmacy.mdb")

strSQL = "Select MedicineID as [Medicine ID], Name as [Medicine Name],ExpDate as
[Expiration Date] from medicine where datediff('d',now(),Expdate)< 30"

Set rb = db.OpenRecordset(strSQL)

grid2.Cols = 4

grid2.TextMatrix(0, 1) = "Medicine ID"

grid2.TextMatrix(0, 2) = "Medicine Name"

grid2.TextMatrix(0, 3) = "Expiration date"

grid2.Rows = 1

While Not rb.EOF

grid2.Rows = grid2.Rows + 1

grid2.TextMatrix(grid2.Rows - 1, 3) = rb.Fields("Expiration Date")

grid2.TextMatrix(grid2.Rows - 1, 2) = rb.Fields("Medicine Name")
```

```
grid2.TextMatrix(grid2.Rows - 1, 1) = rb.Fields("Medicine ID")

rb.MoveNext

Wend

grid2.Sort = 1

Call resize2

rb.Close

db.Close

'Stock Unit

sql2 = "Select MedicineID as [Medicine ID], Name as [Medicine Name],StockUnit as [Stock
Unit] from medicine where StockUnit<2 order by Name asc"

Data2.RecordSource = sql2

Data2.Refresh

grid.Sort = 1

Call resize

End Sub

Function resize()

Font.Name = grid.Font.Name

  Font.Size = grid.Font.Size

  For c = 0 To grid.Cols - 1

    max_len = 0

    For r = 0 To grid.Rows - 1

      new_len = TextWidth(grid.TextMatrix(r, c))

      If max_len < new_len Then max_len = new_len

    Next r

    grid.ColWidth(c) = max_len + 240

    grid.ColAlignment(c) = flexAlignLeftCenter
```

```
    Next c
    grid.AllowUserResizing = flexResizeBoth
End Function
Function resize2()
Font.Name = grid2.Font.Name
    Font.Size = grid2.Font.Size
    For c = 0 To grid2.Cols - 1
        max_len = 0
        For r = 0 To grid2.Rows - 1
            new_len = TextWidth(grid2.TextMatrix(r, c))
            If max_len < new_len Then max_len = new_len
        Next r
        grid2.ColWidth(c) = max_len + 240
        grid2.ColAlignment(c) = flexAlignLeftCenter
    Next c
    grid2.AllowUserResizing = flexResizeBoth
End Function
Private Sub grid2_Click()
End Sub
```

Add To Stock

```
Private Sub Command6_Click()
Dim response As Integer
If Text9.Text = "" Then
```

```
response = MsgBox("Search field is empty!", vbNo, "Attention!")
Text9.SetFocus
Exit Sub
End If
Call Text9_Change
End Sub
Private Sub Command7_Click()
Dim response As Integer
If Text11.Text = "" Then
response = MsgBox("Please Enter the number of medicine correctly!", vbNo, "Attention!")
Text11.Text = ""
Text11.SetFocus
End If
For i = 1 To Len(Text11.Text)
If Asc(Mid(Text11.Text, i, 1)) < 48 Or Asc(Mid(Text11.Text, i, 1)) > 57 Then
response = MsgBox("Please Enter the number of medicine correctly!", vbNo, "Attention!")
Text11.Text = ""
Text11.SetFocus
Exit Sub
End If
Next i
Call add
End Sub
Private Sub DBGrid1_Click()
End Sub
```

```
Private Sub Form_Load()
Data1.Refresh
Do While Not Data1.Recordset.EOF
List1.AddItem UCase(Data1.Recordset("Name"))
List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")
Data1.Recordset.MoveNext
Loop
Text1.Text = ""
Text2.Text = ""
Text5.Text = ""
Text9.Text = ""
Text11.Text = ""
Text11.Locked = True
End Sub
Private Sub List1_dblClick()
Data1.Refresh
Do While Not Data1.Recordset.EOF
If List1.ItemData(List1.ListIndex) = Data1.Recordset("MedicineID") Then
Text1.Text = UCase(Data1.Recordset("MedicineID"))
Text2.Text = UCase(Data1.Recordset("Name"))
Text5.Text = (Data1.Recordset("StockUnit"))
End If
Data1.Recordset.MoveNext
Loop
Text1.Locked = True
```

```
Text2.Locked = True

Text5.Locked = True

Text11.Locked = False

Text11.SetFocus

End Sub

Private Sub Text9_Change()

If Text9.Text = "" Then

List1.Clear

End If

If Text9.Text <> "" Then

If Asc(Left(Text9.Text, 1)) > 96 And Asc(Left(Text9.Text, 1)) < 123 Or
Asc(Left(Text9.Text, 1)) > 64 And Asc(Left(Text9.Text, 1)) < 133 Then

searchtext

Else

searchnumber

End If

End If

End Sub

Function searchtext()

Data1.Refresh

List1.Clear

j = Len(Text9.Text)

Do While Not Data1.Recordset.EOF

If UCase(Text9.Text) = UCase(Mid(Data1.Recordset("Name"), 1, j)) Then

List1.AddItem UCase(Data1.Recordset("Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")
```

```
End If

Data1.Recordset.MoveNext

Loop

End Function

Function searchnumber()

Data1.Refresh

List1.Clear

Do While Not Data1.Recordset.EOF

If Text9.Text = Data1.Recordset("MedicineID") Then

List1.AddItem Data1.Recordset("MedicineID") + "  " + UCase(Data1.Recordset("Name"))

List1.ItemData(List1.ListCount - 1) = Data1.Recordset("MedicineID")

End If

Data1.Recordset.MoveNext

Loop

End Function

Function add()

Dim response As Integer

response = MsgBox("Are you sure that you want to change the stock unit of this record?",
vbYesNoCancel, "Warning")

If response = vbYes Then

Data1.Refresh

c = 0

Do While Not Data1.Recordset.EOF

If Val(Text1.Text) = Data1.Recordset("MedicineID") Then

Data1.Recordset.Edit

Data1.Recordset("StockUnit") = Val(Val(Text11.Text) + Val(Data1.Recordset("StockUnit")))
```

```
Data1.Recordset.Update

Data1.Refresh

Data2.Refresh

c = 1

Exit Do

End If

Data1.Recordset.MoveNext

Loop

If c = 1 Then

   response = MsgBox("Selected medicine's stock unit is updated succesfully", vbNo,
"Congragulations!")

      If response = vbOK Then

      List1.Clear

      Call Form_Load

      End If

End If

End If

End Function
```

Celender

```
Private Sub MonthView1_DateClick(ByVal DateClicked As Date)

End Sub
```

Curreny Converter

```
Private Sub Combo3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text1.SetFocus
End If
End Sub
Private Sub Combo1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Combo2.SetFocus
End If
End Sub
Private Sub Combo2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Text3.SetFocus
End If
End Sub
Private Sub Command1_Click()
Dim response As Integer
flag = 0
j = 1
'control
If Combo3.ListIndex < 0 Then
response = MsgBox("Select Type first!", vbNo, "Attention!")
```

```
Exit Sub
End If
If Text1.Text = "" Then
response = MsgBox("Please Enter the exchange rate!", vbNo, "Attention!")
    Text1.Text = ""
    Text1.SetFocus
    flag = 1
    Exit Sub
End If
For i = 1 To Len(Text1.Text)
If Asc(Mid(Text1.Text, i, 1)) < 44 Or Asc(Mid(Text1.Text, i, 1)) > 57 Then
    response = MsgBox("Please Enter the exchange rate correctly!", vbNo, "Attention!")
    Text1.Text = ""
    Text1.SetFocus
    flag = 1
    Exit Sub
    End If
Next i
For i = 1 To Len(Text1.Text)
    If Asc(Mid(Text1.Text, i, 1)) > 44 And Asc(Mid(Text1.Text, i, 1)) < 48 Then
    response = MsgBox("Please Enter the exchange rate correctly!", vbNo, "Attention!")
    Text1.Text = ""
    Text1.SetFocus
    flag = 1
    Exit Sub
```

```
    End If
  Next i
  'control
  If flag <> 1 Then
  response = MsgBox("Are you sure that you want to update selected type exchange rate?",
  vbYesNoCancel, "Warning")
    If response = vbYes Then
    d = 0
    Data2.Refresh
    Do While Not Data2.Recordset.EOF
      If Data2.Recordset("CurrencyID") = Combo3.ItemData(Combo3.ListIndex) Then
        Data2.Recordset.Edit
        Data2.Recordset("Rate") = CDbl(Text1.Text)
        Data2.Recordset.Update
        Data2.Refresh
        d = 1
        Exit Do
      End If
    Data2.Recordset.MoveNext
Loop
    End If
    End If
If d = 1 Then
    response = MsgBox("Selected type's exchange rate is updated succesfully", vbNo,
"Congragulations!")
      If response = vbOK Then
```

```
        Combo3.Clear
        Call Form_Load
        End If
    End If
    End Sub
    Private Sub Command8_Click()
    c = 0
    Dim response As Integer
    If Text3.Text = "" Then
    response = MsgBox("Please Enter the amount value !", vbNo, "Attention!")
      Text3.Text = ""
      Text3.SetFocus
      c = 1
      Exit Sub
    End If
    For i = 1 To Len(Text3.Text)
    If Asc(Mid(Text3.Text, i, 1)) < 44 Or Asc(Mid(Text3.Text, i, 1)) > 57 Then
    response = MsgBox("Please Enter the amount number correctly!", vbNo, "Attention!")
    Text3.Text = ""
    Text3.SetFocus
    c = 1
    Exit Sub
    End If
    Next i
    For i = 1 To Len(Text3.Text)
```

```
    If Asc(Mid(Text3.Text, i, 1)) > 44 And Asc(Mid(Text3.Text, i, 1)) < 48 Then
    response = MsgBox("Please Enter the amount number correctly!", vbNo, "Attention!")
    Text3.Text = ""
    Text3.SetFocus
    c = 1
    Exit Sub
    End If
  Next i
  If c = 0 Then
  Dim res As Double
  Text4.Enabled = True
  Data1.Refresh
  Do While Not Data1.Recordset.EOF
    If Data1.Recordset("Type") = Combo1.Text + "/" + Combo2.Text Then
      res = CDbl(Text3.Text) * CDbl(Data1.Recordset("Rate"))
      Text4.Text = res
      Exit Sub
    End If
    Data1.Recordset.MoveNext
    Loop
  Text4.Locked = True
  End If
  End Sub
  Private Sub Form_Load()
  Combo1.Clear
```

```
Combo2.Clear
Combo3.Clear
Data1.Refresh
Data2.Refresh
Do While Not Data2.Recordset.EOF
   Combo3.AddItem (Data2.Recordset("Type"))
   Combo3.ItemData(Combo3.ListCount - 1) = Data2.Recordset("CurrencyID")
Data2.Recordset.MoveNext
Loop
Text1.Text = ""
Text4.Text = ""
Text3.Text = ""
Text4.Enabled = False
Combo1.AddItem ("GBP")
Combo1.AddItem ("EURO")
Combo1.AddItem ("USD")
Combo1.AddItem ("TRY")
Combo2.AddItem ("GBP")
Combo2.AddItem ("EURO")
Combo2.AddItem ("USD")
Combo2.AddItem ("TRY")
Combo1.Text = "TRY"
Combo2.Text = "EURO"
Combo3.Text = "Select Type"
End Sub
```

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call Command1_Click
End If
End Sub
Private Sub Text3_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
Call Command8_Click
End If
End Sub
```

Bills

```
Function resize2()
Font.Name = grid2.Font.Name
  Font.Size = grid2.Font.Size
  For c = 0 To grid2.Cols - 1
    max_len = 0
    For r = 0 To grid2.Rows - 1
      new_len = TextWidth(grid2.TextMatrix(r, c))
      If max_len < new_len Then max_len = new_len
    Next r
    grid2.ColWidth(c) = max_len + 240
    grid2.ColAlignment(c) = flexAlignLeftCenter
  Next c
```

```
    grid2.AllowUserResizing = flexResizeBoth

End Function

Private Sub Command6_Click()

Dim strSQL As String

Set db = OpenDatabase("pharmacy.mdb")

btarih = Month(Combo1.Text) & "/" & Day(Combo1.Text) & "/" & Year(Combo1.Text)

bitarih = Month(Combo2.Text) & "/" & Day(Combo2.Text) & "/" & Year(Combo2.Text)

strSQL = "select Bill as [Bill No],bdate as [Bill Date],cname as[Customer Name],mid as
[Medicine ID],mname as [Medicine Name],munit as [Unit],munitprice as [Unit
Price],msubtotal as [Subtotal] from sales where bdate BETWEEN #" & btarih & "# AND #"
& bitarih & "# order by bdate asc"

Text1.Text = strSQL

'Exit Sub

'strSQL = "select Bill as [Bill No],bdate as [Bill Date],cname as[Customer Name],mid as
[Medicine ID],mname as [Medicine Name],munit as [Unit],munitprice as [Unit
Price],msubtotal as [Subtotal] from sales where datediff('d',now(), bdate)< 30 order by bdate
asc "



Set rb = db.OpenRecordset(strSQL)

grid2.Cols = 9

grid2.TextMatrix(0, 1) = "Bill No"

grid2.TextMatrix(0, 2) = "Bill Date"

grid2.TextMatrix(0, 3) = "Customer Name"

grid2.TextMatrix(0, 4) = "Medicine ID"

grid2.TextMatrix(0, 5) = "Medicine Name"

grid2.TextMatrix(0, 6) = "Unit"

grid2.TextMatrix(0, 7) = "Unit Price"
```

```
grid2.TextMatrix(0, 8) = "Subtotal"
grid2.Rows = 1
While Not rb.EOF
grid2.Rows = grid2.Rows + 1
grid2.TextMatrix(grid2.Rows - 1, 1) = rb.Fields("Bill No")
grid2.TextMatrix(grid2.Rows - 1, 2) = rb.Fields("Bill Date")
grid2.TextMatrix(grid2.Rows - 1, 3) = rb.Fields("Customer Name")
grid2.TextMatrix(grid2.Rows - 1, 4) = rb.Fields("Medicine ID")
grid2.TextMatrix(grid2.Rows - 1, 5) = rb.Fields("Medicine Name")
grid2.TextMatrix(grid2.Rows - 1, 6) = rb.Fields("Unit")
grid2.TextMatrix(grid2.Rows - 1, 7) = rb.Fields("Unit Price")
grid2.TextMatrix(grid2.Rows - 1, 8) = rb.Fields("Subtotal")
rb.MoveNext
Wend
Call resize2
rb.Close
db.Close
End Sub
Private Sub Form_Load()
Data5.Refresh
Call resize2
grid2.MergeCol(0) = True
grid2.MergeCol(1) = True
grid2.MergeCol(2) = True
grid2.MergeCol(3) = True
```

```
grid2.MergeCol(4) = True
grid2.MergeCol(5) = True
grid2.MergeCol(6) = True
grid2.MergeCol(7) = True
End Sub

Private Sub grid2_Click()
End Sub
```