

NEAR EAST UNIVERSITY



Faculty of Engineering

Department of Computer Engineering

HOSPITAL MANAGEMENT SYSTEM

Graduation Project

COM – 400

Student: Hayan Saleh (20033242)

Supervisor: Assoc. Prof. Dr. Adil Amirjanov

Nicosia - 2007



TABLE OF CONTENTS

AKNOWLEDGMENT	i
ABSTRACT	ii
INTRODUCTION	iii
1. Visual Basic	1
Introduction	1
1.1 Components used in project	1
1.1.1 Forms	1
1.1.2 Controls	1
1.2 Project Scope	2
1.2.1 Block Scope	2
1.2.2 Variable scope	2
1.2.2.1 What is variable?	2
1.2.2.2 Storing Information in Variables	3
1.2.2.3 Declaring Variables	3
1.3 Procedure Scope	3
1.4 Events and Events Handling	4
1.5 Error Handlers	4
1.6 The Controls used in this project	8
1.6.1 Command Button	8
1.6.2 Text Box	8
1.6.3 Option Button	9
1.6.4 Image	9
1.6.5 Shape	9
1.6.6 Combo Box	10
1.6.7 Microsoft Flex Grid	10
1.6.8 Label	10

2.DATABASE	11
Introduction	11
2.1 What is database?	11
2.2 Database Management System	11
2.3 Database Design	12
2.4 Microsoft Access	13
Introduction	13
2.4.1 Project Database	13
2.4.1.1 Tables	13
2.4.1.2 Relationships	15
2.4.1.3 Relationships used to connect between tables	15
2.4.1.4 Referential Integrity	16
2.5 SQL (Structured Query Language)	16
2.6 SQL statements used in the project	17
 3.ODBC	18
Introduction	18
3.1 Register a database using ODBC in Windows XP	19
3.2 DAO (Data Access Object)	22
3.2.1 Add DAO references in Visual Basic 6	22
3.2.2 Connect to hospital database using DAO	23
3.2.3 Execute SQL statements in DAO	23
3.2.4 Using VB functions in DAO	23
 4. HOSPITAL MANAGEMENT SYSTEM	25
Overview	25
4.1 Login Form	25
4.2 Main Form	26

4.3 Doctors and Services	27
4.3.1 Doctor Details	27
4.3.1.1 Add Doctor	28
4.3.2 Services	29
4.3.2.1 Add Service	29
4.4 Patients	30
4.4.1 Out Patients	31
4.4.1.1 Out Patient Details	31
4.4.1.1.1 Add Patient	31
4.4.1.1.2 Display Patients	33
4.4.1.2 Doctor Appointments	33
4.4.1.2.1 Add Doctor Appointment	33
4.4.1.2.2 Cancel Doctor Appointment	34
4.4.1.3 Service Appointments	36
4.4.1.3.1 Add Service Appointment	36
4.4.1.3.2 Cancel Service Appointment	37
4.4.1.4 Patient Prescription	38
4.4.1.4.1 Add Prescription	38
4.4.1.5 Bill Payments	39
4.4.1.5 .1 View Doctor Payments	39
4.4.1.5 .2 View Service Payments	40
4.4.2.1 Admission	42
4.4.2.1.1 Add In Patient	42
4.2.2.2 Payments	43
4.2.2.2.1 Add Payment	43
4.2.2.2.2 View All Payments	44
4.5 Management	47
4.5.1 Hospital	48
4.5.1.1 Doctor Management	48
4.5.1.1.1 Doctor Schedule	48

4.5.1.2 Services Management	48
4.5.1.2.1 Services Schedule	48
4.5.1.3Rooms	49
✓ 4.5.1.3.1 Add Room	49
4.5.2 Employee Management	50
4.5.2.1 Add Employee	50
CONCLUSION	51
REFERENCES	52

ACKNOWLEDGMENT

First of all I am happy to complete the task which I had given with blessing of God and also I am grateful to all the people in my life who have, supported me, advised me, taught me and who have always encouraged me to follow my dreams and ambitions. My dearest parents, my brothers, my friends and my tutors. They have taught me that no dream is unachievable.

I wish to thank my advisor, Assoc. Prof. Dr. Adil Amirjanov, for intellectual support, encouragement, and enthusiasm, which made this project possible, and his patience for correcting both my stylistic and scientific errors.

My sincerest thanks must go to my friends Juma Al-khatib, Issam Yasen, Salam Abd-Alamir, Maher Nezha, Omar Aziz and all friends who shared their suggestions and evaluations throughout the completion of my project. The comments from these friends enabled me to present this project successfully.

Finally, I wish by this project to be useful for all students, especially Computer Engineering to support our improvement.

And above, I thank God for giving me stamina and courage to achieve my objectives.

To all of them, my love and respect

ABSTRACT

This project is a hospital management system.

The following techniques are used in the project are Microsoft Visual Basic 6.0, also Microsoft Access for creating Data Base and its tables also with some simple SQL Queries to manage the database.

The aim of this project is to help the user to manage the hospital like adding, editing, deleting and searching the doctors, employees and payments.

Daily there are more than a hundred patients went to the hospitals , some patients pays in cash and some pays by using credit cards so this payments operation must be managed and controlled daily. In the other hand the hospital must hire employees and doctors with full information about them in which they can help and treat the patients and also the hospital have to put some services which are needed in treatment process.

This program helps the user to manage the whole information in general and save it in database so they can use it later when it's needed.

INTRODUCTION

This project is talking about Hospital Management System using visual basic programming language, Visual Basic is a visual programming environment for developing Windows and Web applications. Visual Basic makes it possible to develop complicated applications very quickly. The programmer designs windows graphically, drags program elements from the Visual Basic Toolbox and writes basic code for each element. Visual Basic is event-driven which means that procedures are called automatically when the end user chooses menu items, clicks the mouse and moves objects on the screen.

Using Microsoft Access the project's database created and all the tables inside it are related to each others, the database is an organized collection of data. A database management system (DBMS).It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

The project includes 4 chapters:

Chapter one: describes the visual basic programming language, components used to design the forms, how to declare the variables and how to use the procedures.

Chapter two: talks about MS Access, the tables used in the database, the relationships that connects each table with other and SQL statements used in the project.

Chapter three: describes the connection between data base and visual basic and how they are working together in the same program.

Chapter four: represents the program using a block diagram and snapshots, and describes how the user can use the program without any difficulty.

CHAPTER 1

VISUAL BASIC

Introduction

Visual Basic is a visual programming environment for developing Windows (also Web now) applications. Visual Basic makes it possible to develop complicated applications very quickly. The programmer designs windows graphically, drags program elements from the Visual Basic Toolbox and writes basic code for each element. Visual Basic is event-driven which means that procedures are called automatically when the end user chooses menu items, clicks the mouse and moves objects on the screen [1].

1.1 Components used in project

1.1.1 Forms

Forms are the basic building blocks for your UI. Each form in your program represents a window that appears to users. When working in the Visual Basic IDE (integrated development environment), a form is the *designer* that you use to design your UI, much the same as you would use Windows Paint to draw a picture [2].

1.1.2 Controls

Controls are used in the designer to create the look of your UI. A control is an object that has a predefined appearance and behavior. For example, a *Command* control looks and behaves like a push button—when a user clicks it, its appearance changes to show that it has been pressed [2].

Each control in Visual Basic has its own purpose. For example, a *TextBox* control is used for entering text, while a *PictureBox* control is used for displaying pictures. There are more than fifty different controls included in Visual Basic; you can also create your own controls known as user controls [2].

1.2 Project Scope

The scope of a declared element is the set of all code that can refer to it without qualifying its name or making it available through an Imports Statement. An element can have scope at one of the following levels:

Block scope	Available only within the code block in which it is declared
Variable scope	Available only in the general part
Procedure scope	Available to all code within the procedure in which it is declared

Table 1.1:Scopes in VB6

1.2.1 Block Scope

A block is a set of statements enclosed within initiating and terminating declaration statements, such as the following:

- **Do and Loop**
- **For [Each] and Next**
- **If and End If**
- **Select and End Select**
- **Try and End Try**
- **While and wend**
- **With and End With**

If you declare a variable within a block, you can use it only within that block. In the following example, the scope of the integer variable cube is the block between **If** and **End If**, and you can no longer refer to cube when execution passes out of the block [2].

```
If n < 1291 Then
    Dim cube as Integer
    cube = n ^ 3
End If
```

1.2.2 Variable scope

1.2.2.1 What is variable?

Variables are an important concept in computer programming. A variable is a letter or name that can store a value. When you create computer programs, you can use variables to store numbers, such as the height of a building, or words, such as a person's name. Simply put, you

can use variables to represent any kind of information your program needs [2].

1.2.2.2 Storing Information in Variables

There are three steps to using a variable:

1. **Declare the variable.** Tell the program the name and kind of variable you want to use.
2. **Assign the variable.** Give the variable a value to hold.
3. **Use the variable.** Retrieve the value held in the value and use it in your program [2].

1.2.2.3 Declaring Variables

When you declare a variable, you have to decide what to call it and what *data type* to assign to it [2].

You declare a variable using the **Dim** and **As** keywords, as shown below.

```
Dim aNumber As Integer
```

This line of code tells the program that you want to use a variable named aNumber, and that you want it to be a variable that stores whole numbers (the Integer data type).

Because aNumber is an Integer, it can store only whole numbers. If you had wanted to store 42.5, for example, you would have used the Double data type. And if you wanted to store a word, you'd use a data type called a String. One other data type worth mentioning at this point is Boolean, which can store a True or False value [2].

1.3 Procedure Scope

As with variables, you can restrict the scope of procedures, and you do that with the Private, Public, Friend, and Static keywords. The Private and Public keywords are the main keywords here; using them, you can specify if a subroutine or function is private to the module or form in which it is declared or public (that is global) to all forms and modules. You use these keywords before the Sub or Function keywords like this:

```

Private sub calculate ()

    Dim number1 as integer

    Dim number2 as integer

    Number1 = number1+number2

End sub

```

You can also declare procedures as friend procedures with the Friend keyword.

Friend procedures are usually used in class modules (they are not available in standard modules, although you can declare them in forms) to declare that the procedure is available outside the class, but not outside the current project [2].

1.4 Events and Events Handling

In Visual Basic 6.0, events are tied to specific objects and have their own event-handling code. For example, on a form with a button and a menu, each has its own **Click** event; you have to write code in the event handler for each, even if they both perform exactly the same function [1].

```

Private Sub HelpButton_Click ()
    HelpButton.Caption = "Help me!"
End Sub

Private Sub HelpMenu_Click()
    HelpMenu.Caption = "Help me!"
End Sub

```

1.5 Error Handlers

Enables an error-handling routine and specifies the location of the routine within a procedure; can also be used to disable an error-handling routine.

Without an On Error statement, any run-time error that occurs is fatal: an error message is displayed, and execution stops.

Whenever possible, we suggest you use structured exception handling in your code, rather than resorting to unstructured exception handling and the On Error statement [2].

```
Private sub calculate()  
    ^  
    Dim a as integer  
  
    Dim b as integer  
  
    b=a/0  
  
End sub
```

Here is an example on error handling in VB6, in this example I wrote the code above and generate an error; a message appears till that there is an overflow as shown below

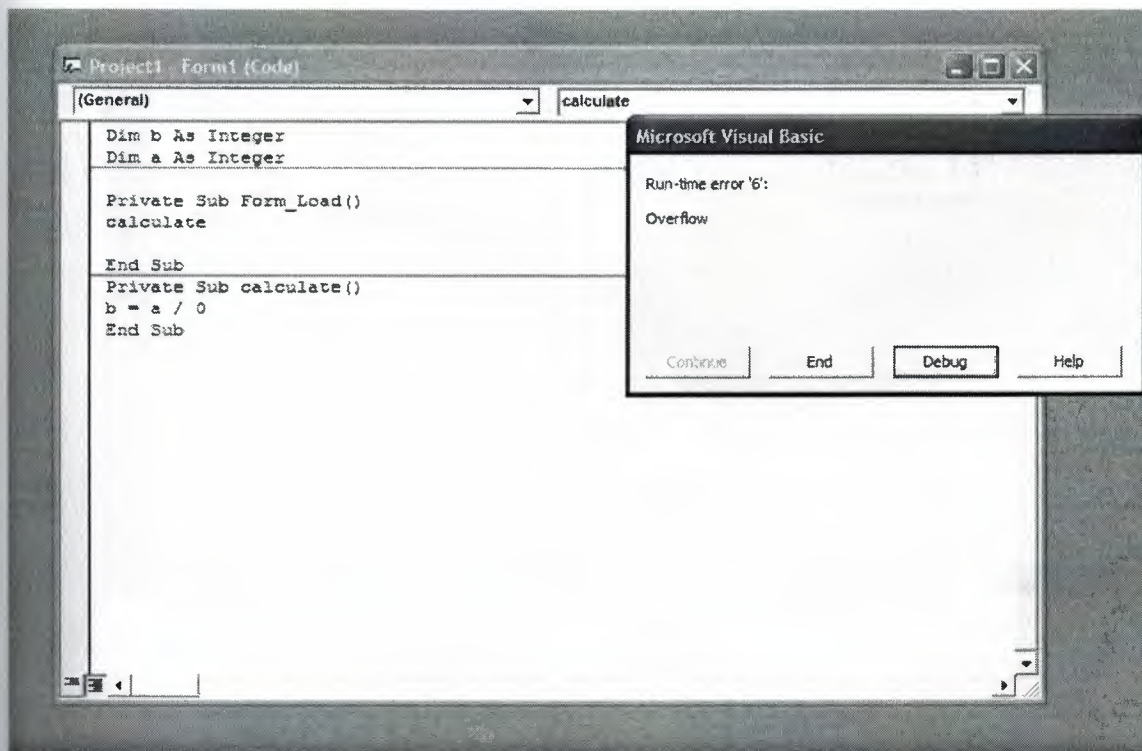


Figure 1.1: Error Message

As shown above the program will stop if End button pressed. To avoid this problem, VB can handle this kind of errors by using On Error keyword and Goto keyword as shown below in the code:

Private sub calculate ()

Dim a as integer

Dim b as integer

On error goto handler

b=a/0

handler:

msgbox err. description

End sub

The error will be handled as shown below

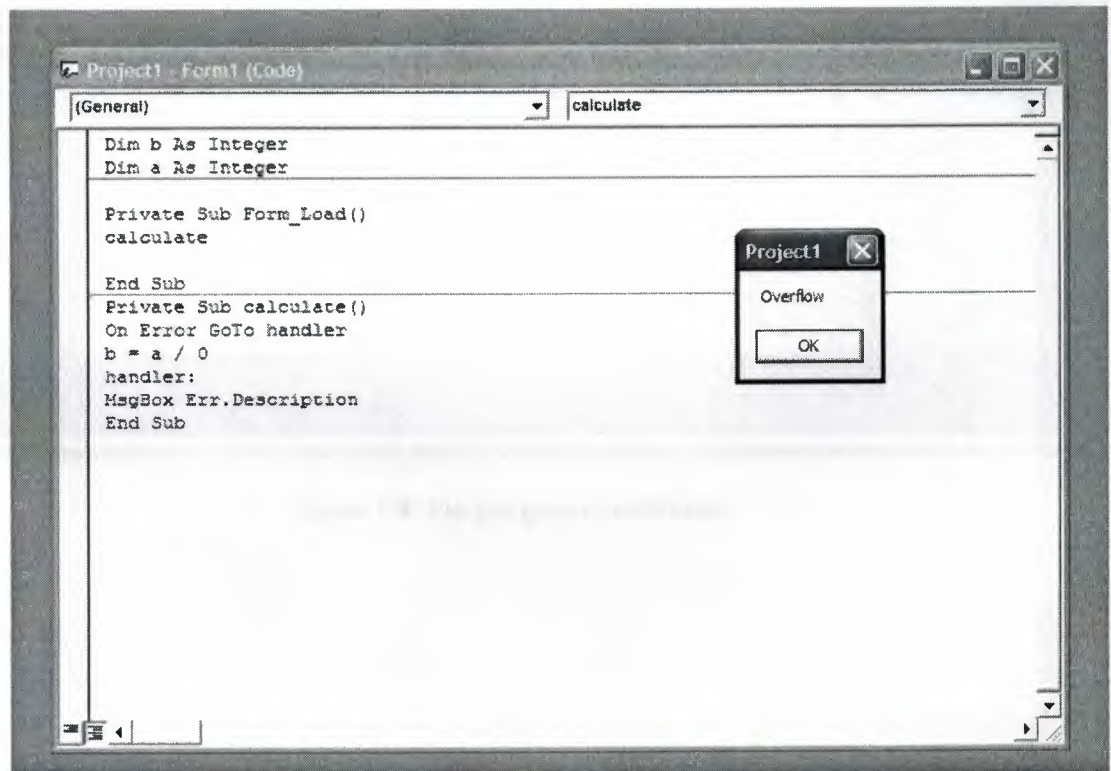


Figure1.2: The Error Handler handles the error

If OK button pressed the program is going to continue in its work as shown below

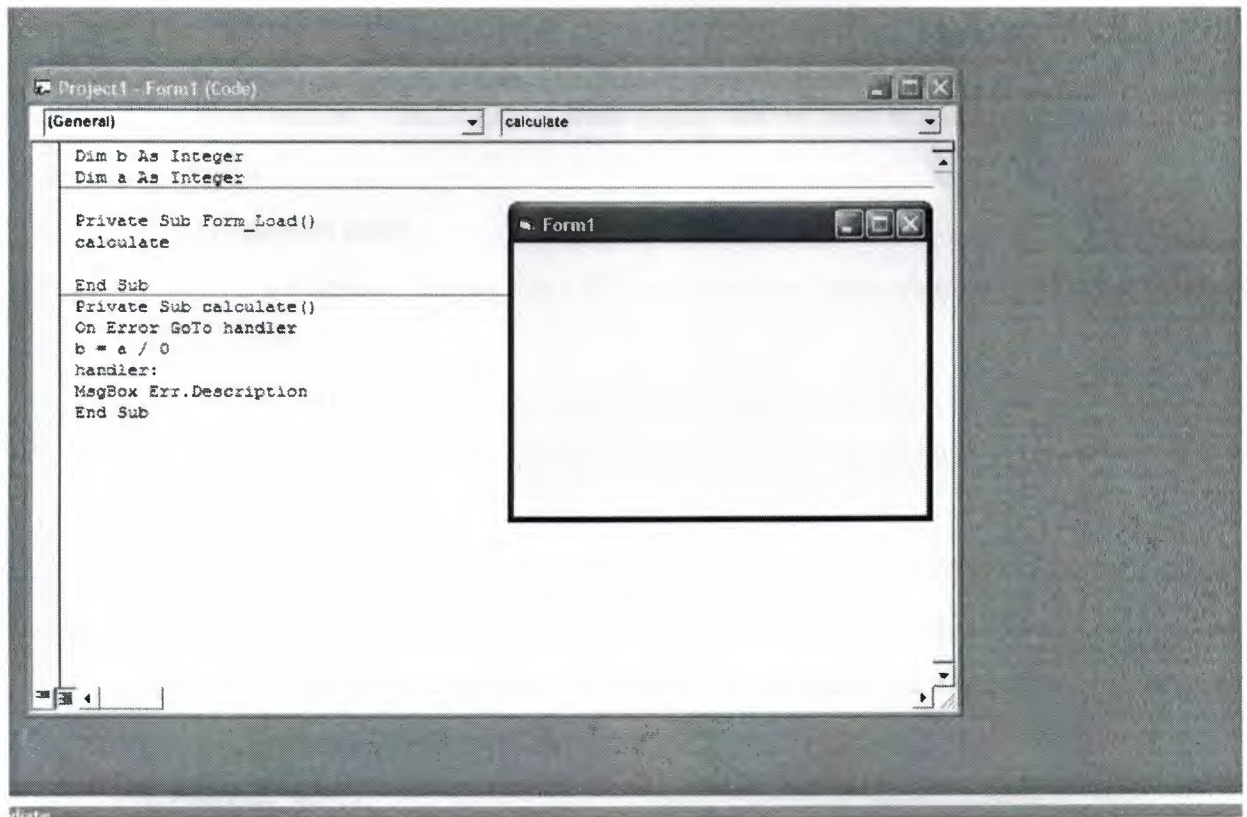


Figure 1.3: The program continued

1.6 The Controls used in this project

1.6.1 Command Button

- **Description:** Looks like a push button and is used to begin, interrupt, or end a process.
- **Properties used:**
 - **Caption:** Returns/sets the text displayed in an object's title bar or below an object's icon
 - **Name:** Returns the name used in code to identify an object.
 - **Visible:** Returns/sets a value that determines whether an object is visible or hidden
 - **Style:** Returns/sets the appearance of the control, whether standard (standard Windows style) or graphical (with a custom picture).
 - **Picture:** Returns/sets a graphic to be displayed in a Command Button, control, if Style is set to 1.
- **Events used:**
 - **Click ():** Occurs when the user presses and then releases a mouse button over an object.

1.6.2 Text Box

- **Description:** Displays information entered at design time by the user, or in code at run time.
- **Properties used:**
 - **Name:** Returns the name used in code to identify an object.
 - **Visible:** Returns/sets a value that determines whether an object is visible or hidden
 - **Text:** Returns/sets the text contained in the control.
 - **Multiline:** Returns/sets a value that determines whether a control can accept multiple lines of text.

- **Events used:**

- **Change():** Occurs when the contents of a control have changed.

1.6.3 Option Button

- **Description:** Displays an option that can be turned on or off.

- **Properties used:**

- **Caption:** Returns/sets the text displayed in an object's title bar or below an object's icon.
 - **Name:** Returns the name used in code to identify an object.
 - **Value:** Returns/sets the value of an object.

- **Events used:**

- **Click():** Occurs when the user presses and then releases a mouse button over an object.

1.6.4 Image

- **Description:** Displays a graphic.

- **Properties used:**

- **Name:** Returns the name used in code to identify an object.

- **Events used:**

- None.

1.6.5 Shape

- **Description:** A graphical control displayed as a rectangle, square, oval, circle or rounded rectangle or square.

- **Properties used:**

- **Shape:** Returns/sets a value indicating the appearance of a control.

- **Events used:**

- None.

1.6.6 Combo Box

- **Description:** Displays a list of items from which the user can select one or more.
- **Properties used:**
 - **Name:** Returns the name used in code to identify an object.
 - **Text:** Returns/sets the text contained in the control.
 - **Style:** Returns/sets a value that determines the type of control and the behavior of its list box portion.
- **Events used:**
 - **Click():** Occurs when the user presses and then releases a mouse button over an object.

1.6.7 Microsoft Flex Grid

- **Description:** Displays data in table.
- **Properties used:**
 - **Name:** Returns the name used in code to identify an object.
 - **Appearance:** Returns/sets whether a control should be painted with 3-D effects.
- **Events used:**
 - **None.**

1.6.8 Label

- **Description:** Displays data in table.
- **Properties used:**
 - **Name:** Returns the name used in code to identify an object.
 - **Caption:** Returns/sets the text displayed in an object's title bar or below an object's icon.
- **Events used:**
 - **None.**

CHAPTER 2

DATABASE

Introduction

The database is an organized collection of data. A database management system (DBMS) such as Access, FileMaker Pro, Oracle or SQL Server provides you with the software tools you need to organize that data in a flexible manner. It includes facilities to add, modify or delete data from the database, ask questions (or queries) about the data stored in the database and produce reports summarizing selected contents.

2.1 What is database?

The database one or more, large structured sets of persistent data. Usually associated with software to update and query the data. A simple database might be a single file containing many records, each of which contains the same set of fields, where each field is a certain fixed width. A database is one component of a database management system [3].

2.2 Database Management System

The DBMS is in charge of access, security, storage and a host of other functions for the database system. It does this through a selection of computer programs. This allows it to manage the large, structured sets of persistent data, which make up the database, and provide access to the data for multiple, concurrent users whilst maintaining the integrity of the data [3].

The DBMS provides security facilities in a variety of forms, both to prevent unauthorized access and to prevent authorized users from accessing data at the same time as each other or overwriting information, which they should not. As a first line of security to prevent unauthorized users from accessing the system, it uses user names and passwords to identify operators, programs and individual machines and sets of privileges assigned to them. These privileges can include the ability to read, write and update data in the database [3].

The DBMS controls who is able to read and write data through the use locks. Locks are used for read and write to specific rows of data in relational

systems, or objects in object oriented ones, which are currently being used. By doing this only one user at a time can alter data [3].

DBMS's typically run on special hardware, for example servers which have been specially designed to only run databases and often only with specific database systems in mind. This allows the hardware designers to increase the number and speed of its network connections, use multiple processors and discs to speed up searched for information, increase the amount of memory and cache, as well as a host of other features not found on standard hardware. This also allows the programmers to take advantage of special features in the hardware to get the best possible performance from the system [3].

2.3 Database Design

Database Design is the process of taking the requirements for the database, i.e. what information is to be stored, who can access it, how many people may be accessing it simultaneously etc. and designing a system which will achieve this [3].

This initial stage is carried out by either a database design specialist, or the Database Administrators and Software Analysts. At the end of it, a schema is produced which defines what data is stored, how it relates to other data, and who can access, add and modify data. This schema is broken down into several sub-schemas which define individual tables and relationships between the tables and the data contained within [3].

2.4 Microsoft Access

Introduction

Microsoft Office Access, previously known as Microsoft Access, is a relational database management system from Microsoft which combines the relational Microsoft Jet Database Engine with a graphical user interface.

Access can use data stored in Access/Jet, Microsoft SQL Server, Oracle, or any ODBC-compliant data container. Skilled software developers and data architects use it to develop application software. Relatively unskilled programmers and non-programmer "power users" can use it to build simple applications. It supports some object-oriented (OO) techniques but falls short of being a fully OO development tool [3].

2.4.1 Project Database

The project database contains 16 table each table hold a different data from the other. The tables have a relationship between each other and the relationship differ from one table to other according to the data held in table.

2.4.1.1 Tables

As mentioned above there 16 table in the database. Here is a figure about all tables in my database.

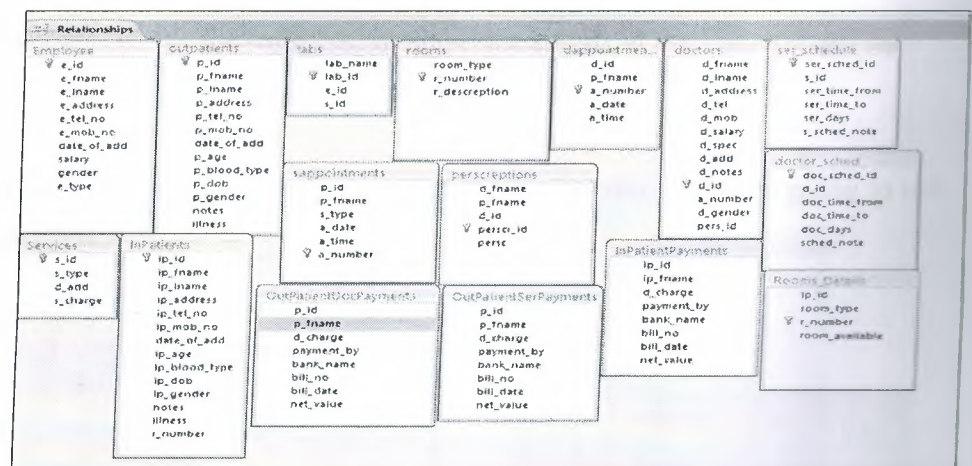


Figure 4: Tables used in database

1. **Employee** : holds the information about the employee like full name, address, id, telephone number, salary, date of add and type.
2. **Outpatient**: the table holds all information about the patients that comes to the hospital and don't stay in it.
3. **Inpatient**: the table holds all information about the patients that comes to the hospital and stay in it.
4. **Doctors**: the table holds all information about the doctors and the patients related to the doctor, the prescriptions that the doctor wrote.
5. **Services**: holds the services available in the hospital and how much each service cost.
6. **OutPatientPayments**: holds the information about how much the patient paid and by which way he/she paid, and for whom he/she paid.
7. **InPatientPayments**: holds the information about how much the patient paid and by which way he/she paid.
8. **OutPatientDocPayments**: holds the information about how much the patient paid and by which way he/she paid to the doctor.
9. **OutPatientSerPayments**: holds the information about how much the patient paid and by which way he/she paid for the service.
10. **Ser_Sched**: holds the information about in which days and which time the service is available.
11. **Doc_Sched**: holds the information about in which days and which time the doctor is available
12. **Prescriptions**: holds the information about the prescriptions that wrote by the doctor.
13. **Rooms**: Holds the information about the rooms in the hospital and the availability of the rooms.

That's all about the tables that used in the database, next iam going to talk about the relationships that used to connect each table by another.

2.4.1.2 Relationships

In this section iam going to talk about the relationships between the tables. Here is a figure shows the relationships that I establish.

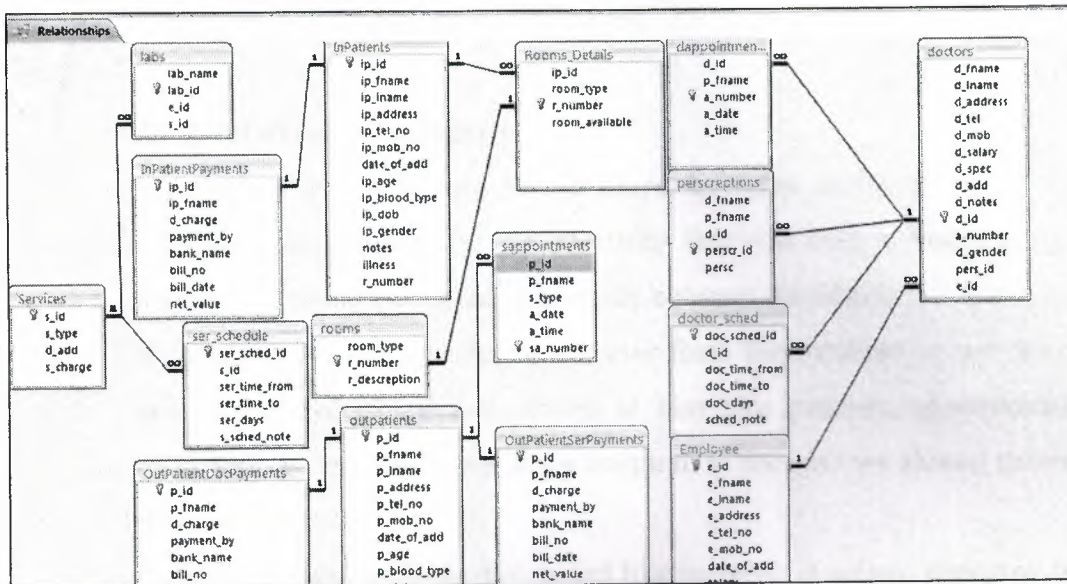


Figure 2.5: Relationships between tables

2.4.1.3 Relationships used to connect between tables

1. One to Many relationships (1 to ∞)

The one to many relationships is used to relate one record in table with many records in another. Examples are one doctor to many patients or one patient to many appointments [3].

2. Many to One relationship (∞ to 1)

The many to one relationship (often called the lookup table relationship) tells access that many records in the table are related to a single record in another table.

Normally, many to one relationship are related to a single record in another table. Example on this relationship is many visitors can visit one patient [3].

3. One to One relationship (1 to 1)

The one to one relationship, though rarely used in database systems, can be a very useful way to link two tables together.

An example on one to one relationship in this database like a link between Rooms and RoomsDetail [3].

2.4.1.4 Referential Integrity

In addition to specifying relationships between tables in an access database, you can also setup some rules that will help in maintaining a degree of accuracy, referential integrity between the tables.

Example if you want to delete a doctor from the database so we should also delete everything that related to him like patients, appointments, prescriptions and his record in the hospital, in the case we should enforce deleting integrity.

And if we want to update his record like increase in salary, changing the address and changing telephone number so we should enforce updating integrity [3].

2.5 SQL (Structured Query Language)

SQL is a transform-oriented and non-procedure language designed to use relations to transform inputs into required outputs.

SQL has two major components

- A data definition language (DDL) for defining the database structure [4].
- A data manipulation language (DML) for retrieving and updating data [4].

SQL contains only these definitional and manipulative commands. It does not contain flow control commands. There are no if...then...else, goto, do....while and other commands to provide a flow control, due to this lack of computational completeness, SQL can be used in two ways

- Use SQL interactively by entering the statements at a terminal.
- Embed SQL statement in a procedural language.

2.6 SQL statements used in the project

- **SELECT**

The purpose of the select statement is to retrieve and display data from one or more database tables.

It is an extremely powerful command capable of performing the equivalent of the relational algebra.

Format of the statement is

SELECT [DISTINCT/] FROM table-name(s) WHERE[for condition]*

- **INSERT**

Insert statement is used to add new entries to an existing database table.

Format of the statement is

INSERT INTO [table-name] [(column list)] VALUES (data-value-list)

- **UPDATE**

The existing database table entry can be modified using UPDATE statement. UPDATE statement will not change the contents of the primary key.

Format of the statement is

*UPDATE [table-name] SET (column1=value1, column2=value2...
ColumnN=valueN) WHERE [for condition]*

- **DELETE**

The delete statement allows rows to be deleted from the specified table.

Format of the statement is

DELETE FROM [Table-name] WHERE [Search condition]

CHAPTER 3

OPEN DATABASE CONNECTIVITY (ODBC)

Introduction

The ODBC specification offers a procedural API for using SQL queries to access data. An implementation of ODBC will contain one or more applications, a core ODBC library, and one or more "database drivers". The core library, independent of the applications and DBMS systems, acts as an "interpreter" between the applications and the database drivers, whereas the database drivers contain the DBMS-specific details. Thus a programmer can write applications that use standard types and features without concern for the specifics of each DBMS that the applications may encounter. Likewise, database driver implementers need only know how to attach to the core library. This makes ODBC modular [5].

To write ODBC code that exploits DBMS-specific features requires more advanced programming. An application must use introspection, calling ODBC metadata functions that return information about supported features, available types, syntax, limits, isolation levels, driver capabilities and more. Even when programmers use adaptive techniques, however, ODBC may not provide some advanced DBMS features. The ODBC 3.x API operates well with traditional SQL applications such as OLTP, but it has not evolved to support richer types introduced by SQL:1999 and SQL:2003[5].

3.1 Register a database using ODBC in Windows XP

I registered the project database in ODBC to make a connection between the forms in the project and the database.

1. Creating and storing the database

This is my database in path *d:\university stuff\graduation project (vb)*

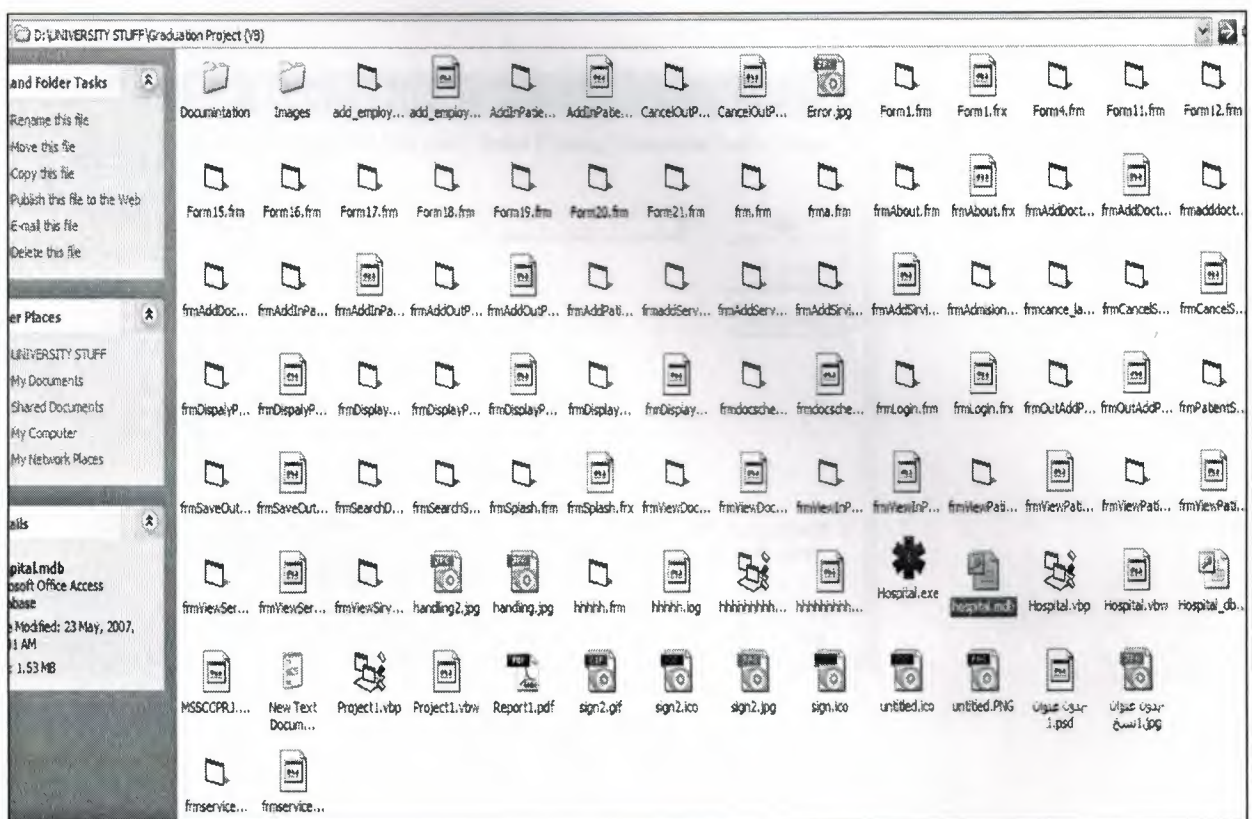


Figure 3.1: database location

2. Where to find the data source dialogue

Go to start → control panel → administrative tools → ODBC

3. ODBC

This data source dialogue for adding a database to ODBC

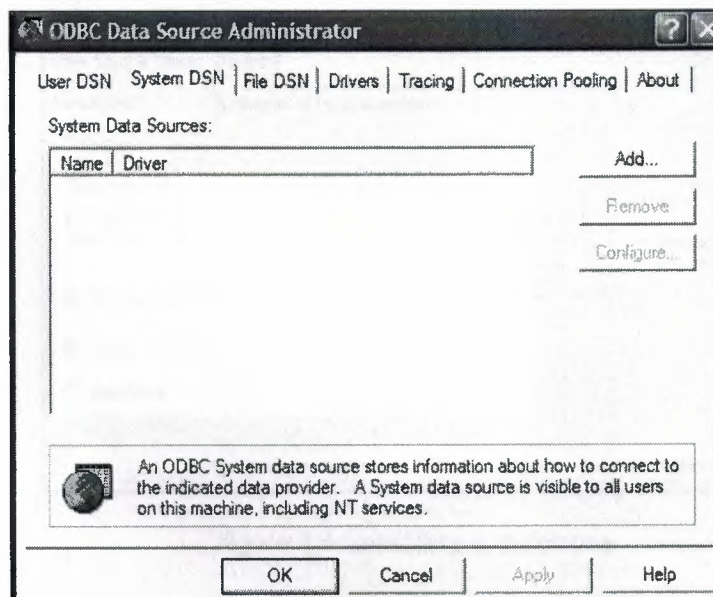


Figure 3.2: ODBC

4. Create new data source

This window specified which kind of database you want to create

Here I create a access database because the extension of database in .mdb

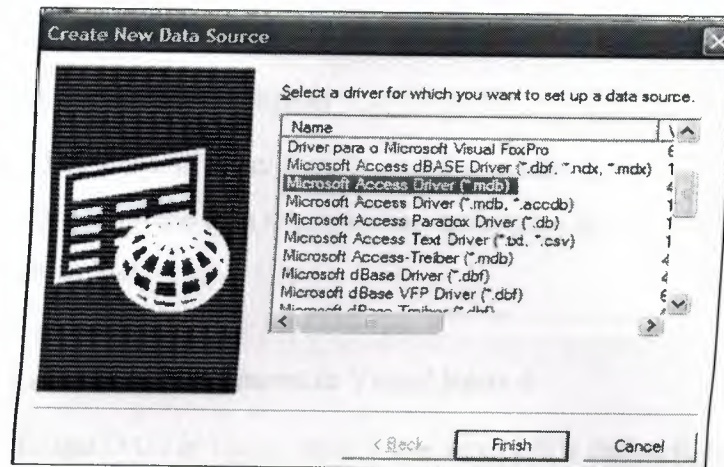


Figure3.3: Create new data source

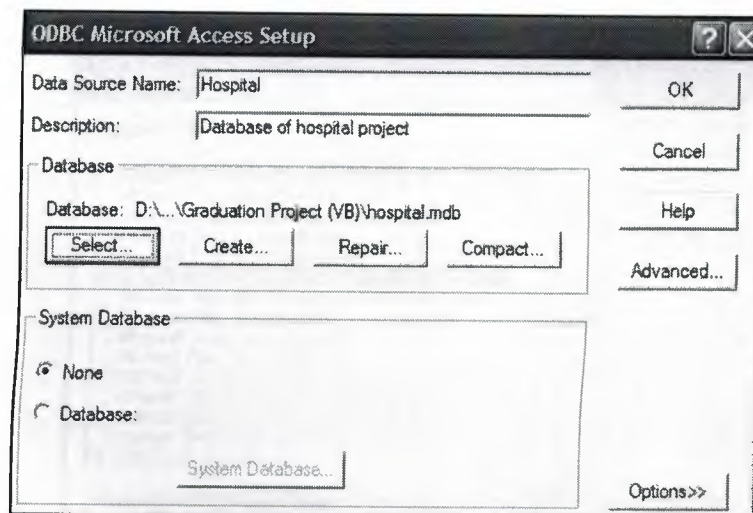


Figure 3.4: selecting a database

3.2 DAO (Data Access Object)

The Data Access Object (DAO) model provides you with an object-oriented interface to all the data manipulation techniques inherent in Microsoft Jet Database files.

3.2.1 Add DAO references in Visual Basic 6

To use DAO in visual basic 6 you should first define it, here is the method to define it.

Start→VB6→Project→References→DAO 3.6

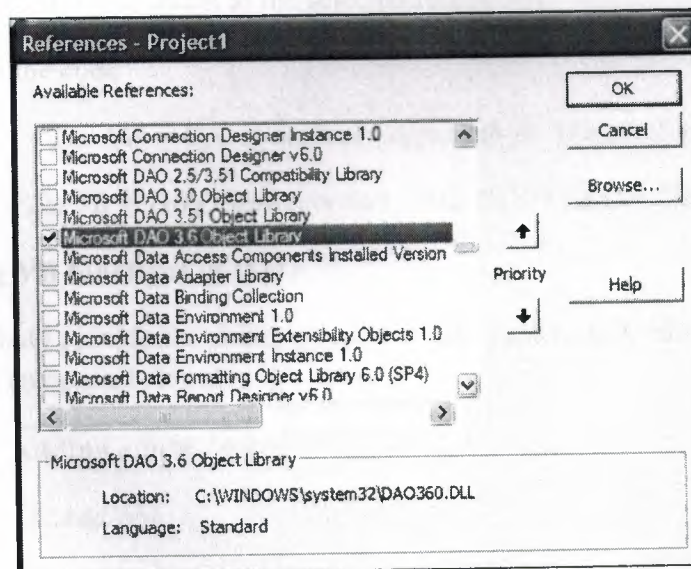


Figure 3.5: References window where you can define DAO object library in VB6

3.2.2 Connect to hospital database using DAO

To connect to the database I used one of the properties that are included in DAO, OpenDatabase is a property in DAO that allow you to connect to the database and open it. First I declared a variable for example DB,TB then I used the following code

```
Set DB = OpenDatabase (App.Path & "\hospital.mdb")
```

```
Set TB = DB.OpenRecordset ("table name")
```

3.2.3 Execute SQL statements in DAO

DAO has the ability to use SQL statements,the Execute object allow the user to apply SQL statement to the selected record set.

Here is the code

```
Set DB = OpenDatabase (App.Path & "\hospital.mdb")
```

```
Set TB = DB.OpenRecordset ("SELECT * FROM Doctors")
```

3.2.4 Using VB functions in DAO

Also DAO provide the ability to use VB functions to add, edit, delete and update an opened record.

- **Adding a new record**

```
TB.AddNew
```

```
TB.Fields ("field name") = text1.text
```

```
TB.Fields ("field name") = text2.text
```

```
TB.Fields ("field name") = text3.text
```

```
.
```

```
.
```

```
.
```

```
TB.Fields ("field name") = textN.text
```

- **Delete a record**

If TB.nomatch = 0 then

TB.Delete

End If

- **Update a record**

TB.Update

- **Edit a record**

TB.Edit

TB.Fields ("field name") = text1.text

TB.Fields ("field name") = text2.text

TB.Fields ("field name") = text3.text

.

.

.

TB.Fields ("field name") = textN.text

- **Close a record and database**

TB.Close

DB.Close

CHAPTER 4

HOSPITAL MANAGEMENT SYSTEM

Overview

In this chapter iam going to talk about the project, the project contains 20 forms each form connects to the database.

The project in general has a friendly GUI (Graphical User Interface), and ease to use.

4.1 Login Form

The login form connects to the database for security purpose, the form checks if the user name and password are matched to the one in the database, if it matches then the user will go to the main form if not the user will receive a warning message tell him to enter the password and user name.

Here is a snap shot for the login form and the code to enter to the main program

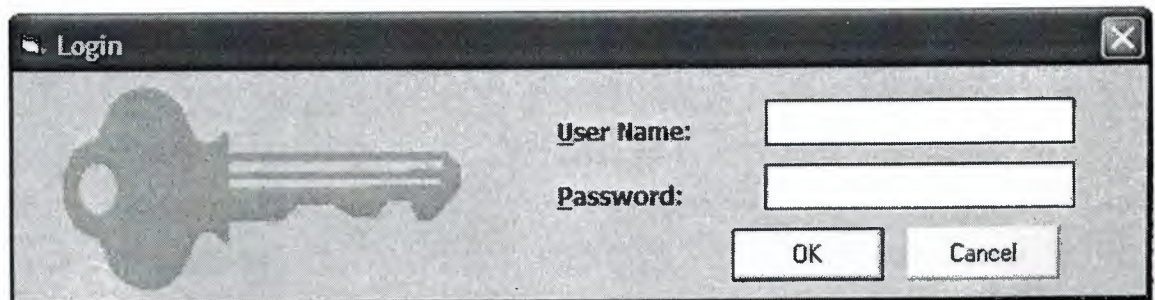


Figure 4.1: Login form

```
Private Sub cmdOK_Click ()  
    If txtPassword.Text = "" Then  
        MsgBox "Please Enter Password", vbOKOnly + vbQuestion, _  
            "Warning"  
        txtPassword.SetFocus  
    End If  
    pass_sql = "select * from pass where pass= '" & txtPassword.Text & "'" "  
    Set tb = db.OpenRecordset (pass_sql)  
    If tb.EOF = False Then
```

```

If txtUserName.Text = tb ("user_name") And txtPassword.Text = tb("pass") Then
    Unload Me
    Form1.Show
Else
    MsgBox "Invalid Password or User Name", vbOKOnly + vbCritical, "Wrong Password"
End If
End If
End Sub

```

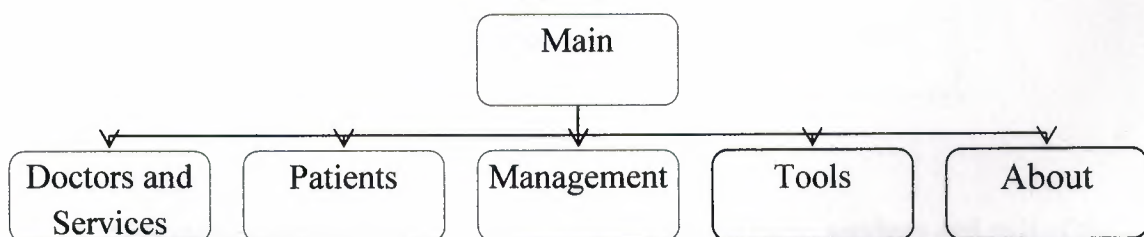
4.2 Main Form

This is the main window of the program, where it leads to the other windows in the program, the main window has a main menu which contains menus and sub menus for other windows.

And also have a statue bar which displays the time and date of the system.

The main menu contains the following:

- Doctors and Services
- Patients
- Management
- Tools
- About



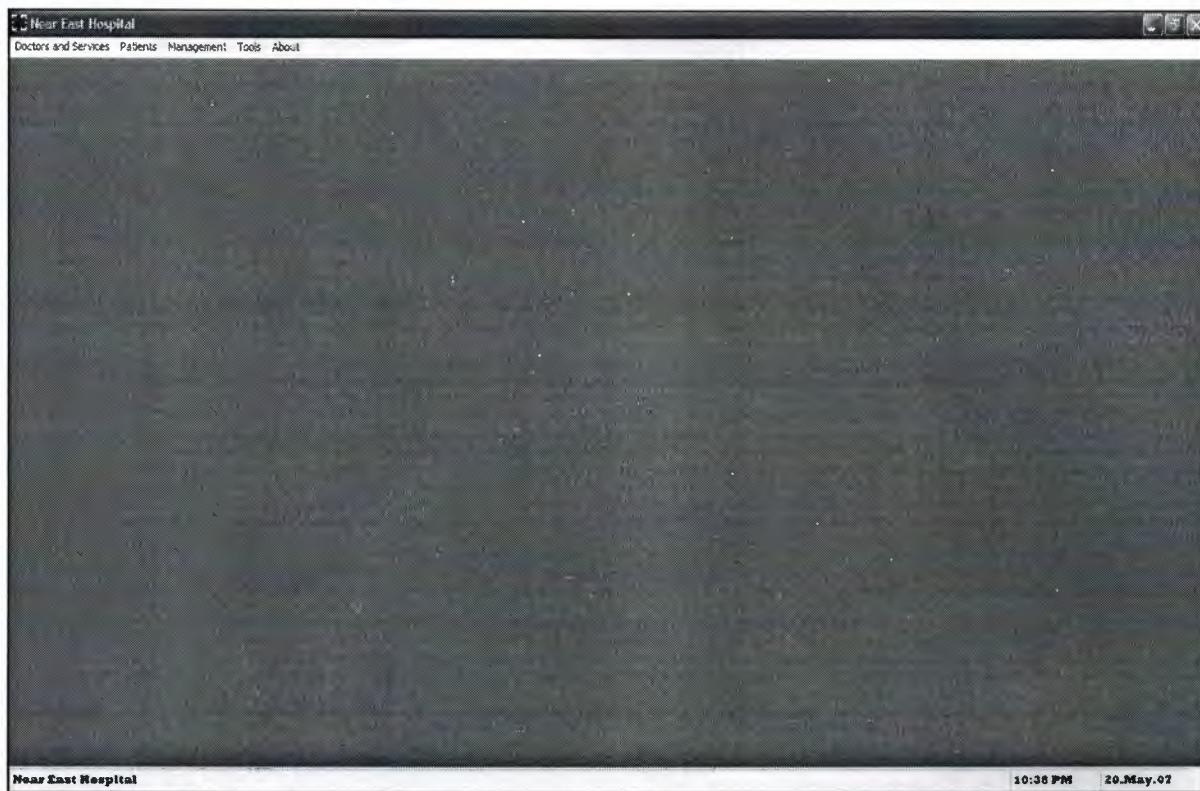
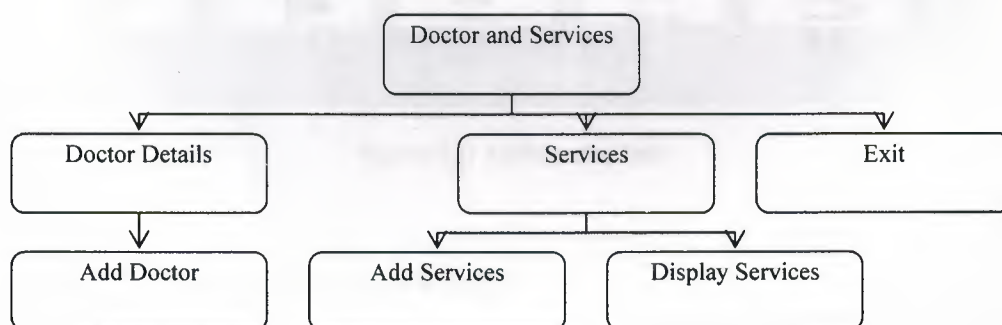


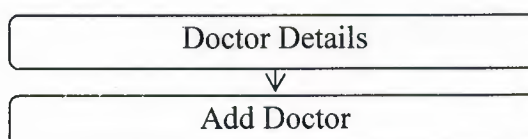
Figure 4.2: Main form

4.3 Doctors and Services



This menu contains sub menus which are doctor details, services and exit of the application.

4.3.1 Doctor Details



4.3.1.1 Add Doctor

The form can add, edit, delete, view and search for doctor. If the doctor found his picture will be displayed.

This is an add doctor window where the doctor's details registered and this window connects to the main database.

Figure 4.3: Add doctor form

- This is the code for adding a new doctor

```
Set db = OpenDatabase (App.Path & "\hospital.mdb")
```

```
If Option1.Value = True Then
```

```
    a = "Male"
```

```
End If
```

```
If Option2.Value = True Then
```

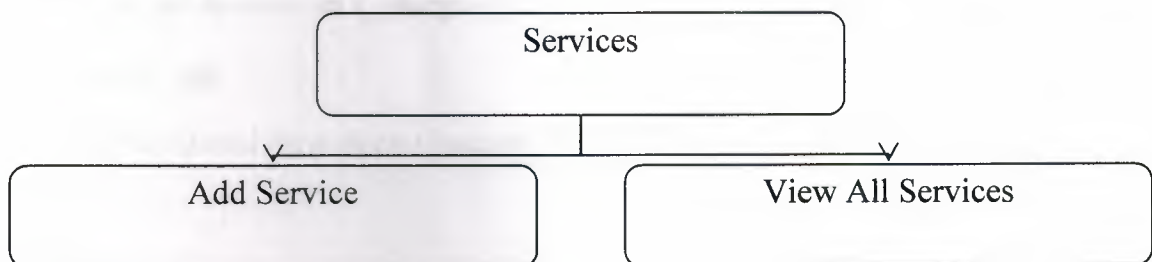
```
    a = "Female"
```

```
End If
```



```
db.Execute ("insert into
doctors(d_id,d_fname,d_lname,d_address,d_tel,d_mob,d_salary,d_spec,d_add,d_notes,d_ge
nder) values ('" & Text8.Text & "','" & text1.Text & "','" & Text7.Text & "','" & text3.Text &
','" & Text5.Text & "','" & Text4.Text & "','" & Val(Text6.Text) & "','" & text2.Text & "','" &
Date & "','" & Text10.Text & "','" & a & "'"))
```

4.3.2 Services



4.3.2.1 Add Service

This is add services window where the user can add, edit, delete, search, navigate the tables in database and view all the available services.

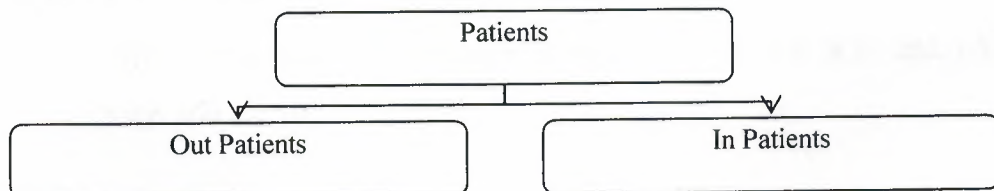
Figure 4.4: Add service form

This is the code for editing an existing Service

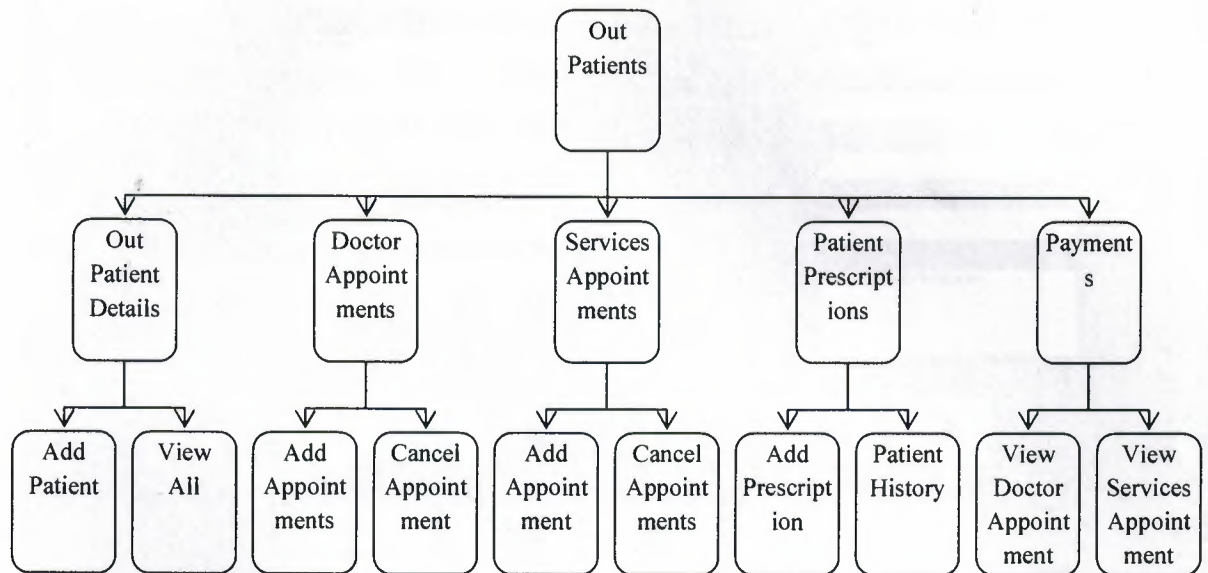
```
Private Sub Edit_Service_Click()  
Set db = OpenDatabase(App.Path & "\hospital.mdb")  
MsgBox "Do You Want To Edit This Record?", vbExclamation + vbYesNo, "Edit?"  
If vbYes Then  
Set tb2 = db.OpenRecordset("services")  
u_sql = "update services set s_charge='" & text3.Text & "' where s_id='" & text1.Text & "'" & ""  
db.Execute (u_sql)  
MsgBox "The Record Have Been Updated", vbInformation + vbOKOnly, "Update"  
End If  
If vbNo Then  
Exit Sub  
End If  
tb2.close  
db.close  
End Sub
```

4.4 Patients

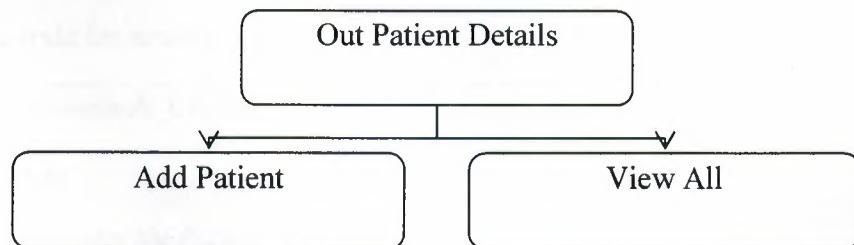
This menu contains out patients and in patients and each one have sub menus that lead to another window.



4.4.1 Out Patients



4.4.1.1 Out Patient Details



4.4.1.1.1 Add Patient

This form connects to the main database and its task is to add, edit, delete, view and search.

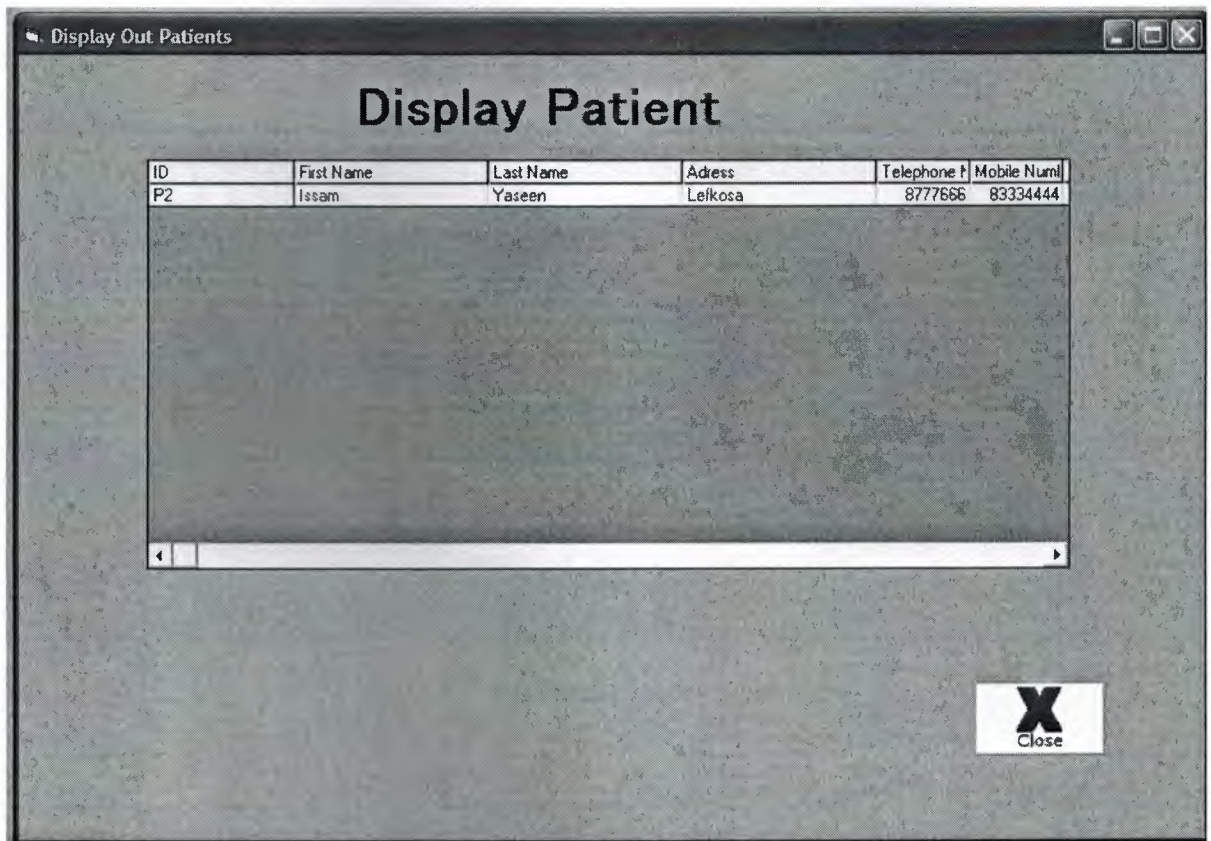
Figure 4.5: Patient details form

This is the code for search a patient

```
Private Sub Command9_Click ()
    Dim id As String
    id = InputBox("Enter the Patient ID", "Search")
    Set db = OpenDatabase (App.Path & "\hospital.mdb", False)
    search_sql = "select * from outpatients where p_id='" & id & "'"
    Set tb = db.OpenRecordset (search_sql)
    If tb.RecordCount < 1 Then
        MsgBox "Patient not found", vbOKOnly + vbCritical, "warning"
    End If
    ShowRecord
End Sub
```


4.4.1.1.2 Display Patients

This form also connects to the database and allow to user to view all the registered patients.



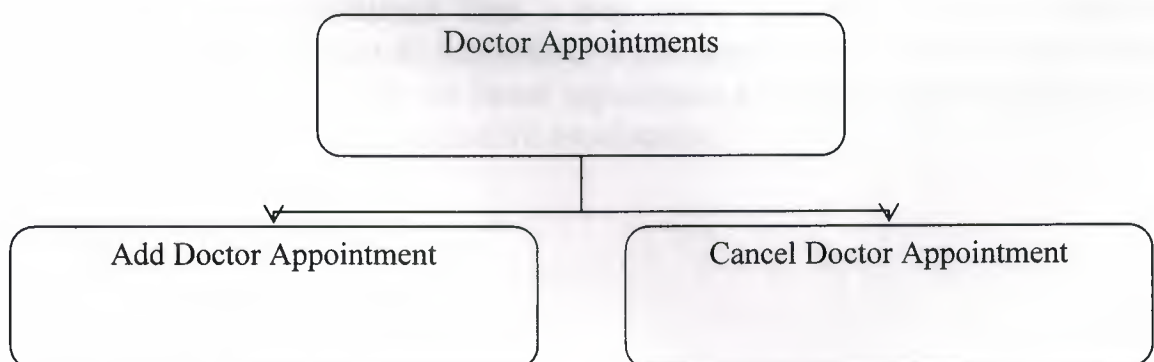
The screenshot shows a window titled "Display Out Patients" with a sub-header "Display Patient". Below the header is a table with the following data:

ID	First Name	Last Name	Adress	Telephone t	Mobile Numl
P2	Issam	Yaseen	Leikosa	8777666	83334444

Below the table is a large empty rectangular area, likely for additional patient details or a list of appointments. In the bottom right corner of the window is a "Close" button with a red 'X' icon.

Figure 4.6: Display patients form

4.4.1.2 Doctor Appointments



4.4.1.2.1 Add Doctor Appointment

This form is to save a doctor appointment; the user can choose which doctor to save the appointment for.

The screenshot shows a software window titled "Patient Appointment" with standard Windows window controls (minimize, maximize, close) in the top right corner. The main title "Patient Appointment" is centered at the top of the window. Below the title, a dashed-line rounded rectangle contains the form fields. The fields are arranged in two columns. The left column contains: "Doctor ID" with a dropdown menu showing "Doc1", "Patient Name" with a text box containing "Issam Yassen", and "Appointment ID" with a text box containing "DApp1". The right column contains: "Date" with a date picker showing "28.May.07" and "Time" with a time picker showing "3:00:00 PM". Below the dashed rectangle, centered, is a button labeled "Save Appointment".

Figure 4.7: Save doctor appointment form

4.4.1.2.2 Cancel Doctor Appointment

This form's task is to cancel an existing appointment the user can choose the appointment from a drop down list, when the user choose the appointment all data related to that appointment will be displayed. When the user clicks on cancel appointment a message appears ask the user if he/she want to cancel the appointment.

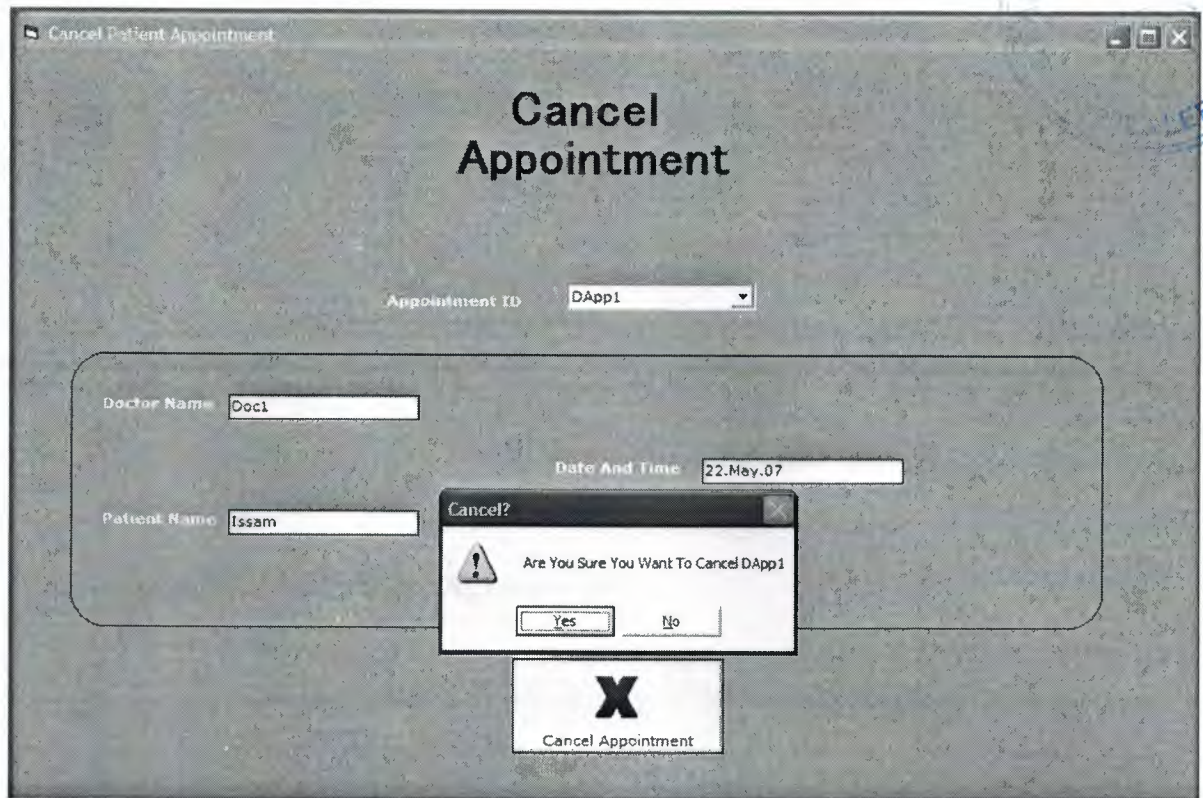


Figure 4.8: Cancel doctor appointment form

This is the code for canceling an existing appointment

```
Private Sub Cancel_Click ()

a = MsgBox ("Are You Sure You Want To Cancel" & Combo2.Text, vbYesNo + vbExclamation,
"Cancel?")

If a = vbYes Then

db.Execute ("Delete from dappointments where a_number=" & Combo2.Text & " ")

End If

If a = vbNo Then

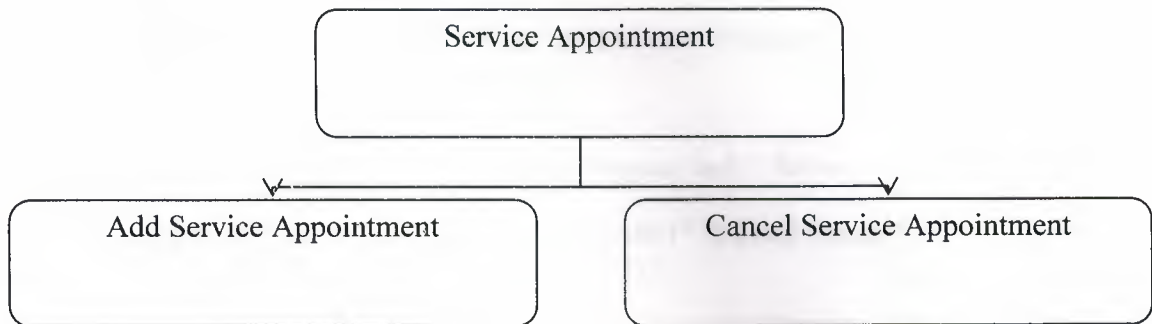
MsgBox "Continue "

End If

clear1

End Sub
```

4.4.1.3 Service Appointments



4.4.1.3.1 Add Service Appointment

This form is designed to add an appointment for a chosen service; the user can select one of the available services, and determine the time and date for the given patient.

The screenshot shows a Windows-style window titled "Service Appointment". Inside the window, the title "Service Appointment" is centered at the top. Below it, there is a "Service Type" dropdown menu with "Blood Test" selected. A large rounded rectangle contains four input fields: "Patient Name" with the text "Issam", "Date" with "24.May.07", "Appointment ID" with "5App1", and "Time" with "5:00:00 PM". At the bottom center of the window is a "Save Appointment" button.

Figure 4.9: Service appointment form

This is code for saving an appointment

```
Private Sub Command1_Click ()  
Dim b As String  
a = DTPicker1.Value
```



```

b = DTPicker2.Value

If Text1.Text = "" Then

MsgBox "Please Fill All The Form", vbOKOnly + vbQuestion, "Warning"

End If

Set db = OpenDatabase (App.Path & "\hospital.mdb", False)

db.Execute ("insert into appointments values (" & Text1.Text & "," & Text3.Text & "," &
& a & "," & b & "," & Combo2.Text & ")")

clear1

End Sub

```

4.4.1.3.2 Cancel Service Appointment

The can cancel an existing service appointment, the user can choose an appointment and click on cancel appointment button, a message will show up asking the user to cancel an appointment or not.

The screenshot shows a Windows-style application window titled "Cancel Service Appointment". The window has a dark gray background. At the top center, the text "Cancel Appointment" is displayed in a large, bold, black font. Below this, there is a label "Appointment ID" followed by a white dropdown menu. Underneath, a rounded rectangular container holds four text input fields arranged in two rows. The first row contains "Patient Name" and "Date", and the second row contains "Service Name" and "Time". At the bottom center of the window, there is a button with a black border, a large black "X" icon, and the text "Cancel Appointment" below it.

Figure 4.10: Cancel service appointment form

The code for choose an appointment from combo box

```
Set db = OpenDatabase (App.Path & "\hospital.mdb", False)

search_sql = "select * from sappointments where sa_number=" & Combo3.Text & " "

Set tb = db.OpenRecordset (search_sql)

If tb.RecordCount < 1 Then

MsgBox "Appointment not found", vbOKOnly + vbCritical, "warning"

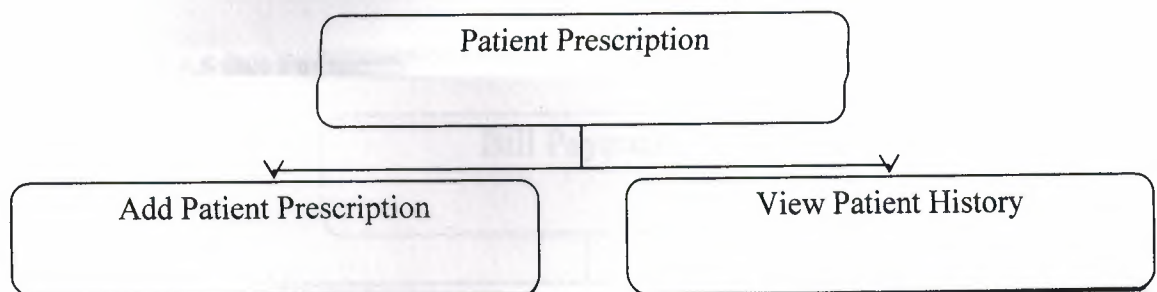
Else

ShowRecord

End If

End Sub
```

4.4.1.4 Patient Prescription



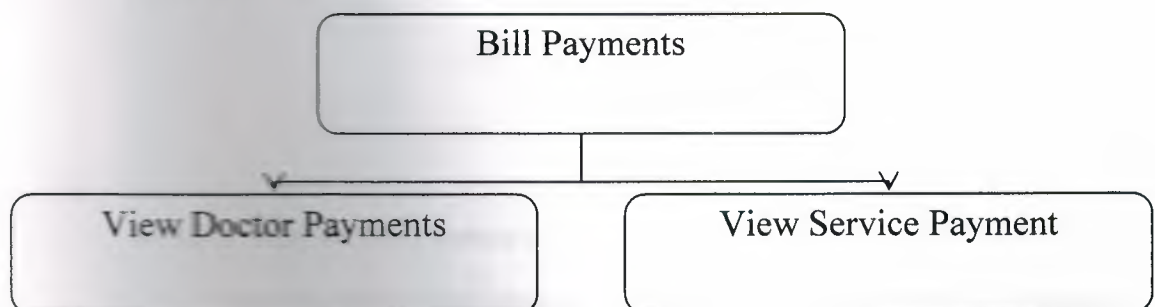
4.4.1.4.1 Add Prescription

This form's task is to let the doctor to store a prescription in the database, the doctor can choose the patient name, and write the prescription.

The screenshot shows a window titled "Perscription" with a standard Windows title bar. The main area has a dark grey background. At the top center, the word "Perscription" is written in a large, bold, black font. Below this, there is a rounded rectangular container with a light grey background. Inside this container, on the left, are three labels with corresponding input fields: "Perscription ID" with a text box containing "DP2", "Doctor Name" with a dropdown menu showing "Hayan", and "Patient Name" with a dropdown menu showing "Issam". To the right of these fields is a larger text area labeled "Perscription" containing the text "1 little spoon of Setamol after meal.". Below the rounded container, centered, is a button labeled "Save Perscription".

Figure 4.11: Add prescription form

4.4.1.5 Bill Payments



4.4.1.5.1 View Doctor Payments

This is a very important form that allows the user to calculate the bill and establish a bill for the out patient.

Doctor Payments

OUT PATIENT PAYMENTS

Bill Date :

Patient Name :

Patient ID:

Doctor Charges :

Bill Amount :

Discount :

Payment

Payment Complete

Bill Number :

Net Value :

Payment Info:

☒ CASH
 ☐ CREDIT CARD
 ☐ CHEQUE

BANK :

SAVE

CLOSE

Figure 4.12: Payments form

4.4.1.5 .2 View Service Payments

Also this one is a very important that allows the user to generate bill for the services that used by a specific patient and how much this service cost.

Service Payments

OUT PATIENT SERVICES PAYMENTS

Bill Date : 21.May.07

Patient Name: Issam Service Name: Blood Test

Patient ID: P2

Services Charges : 50

Bill Amount : 50

Discount : 25

Bill Number : SBP1

Net Value : 37.5

Payment

Payment Complete

OK

Payment Info

☒ CASH ☐ CREDIT CARD ☐ CHEQUE

BANK :


 SAVE  CLOSE

Figure 4.13: Service payments form

The code for completing a payment

```
Private Sub cmdSave_Click ()
Dim str2 As String
txtBillDate.Text = Date
Set db = OpenDatabase (App.Path & "\hospital.mdb")
Set tb = db.OpenRecordset ("OutPatientSerPayments")
tb.AddNew
If optCash.Value = True Then
Text1.Enabled = False
str2 = optCash.Caption
ElseIf optDD.Value = True Then
Text1.Enabled = False
str2 = optDD.Caption
```

```

ElseIf optCheque.Value = True Then

Text1.Enabled = True

str2 = optCheque.Caption

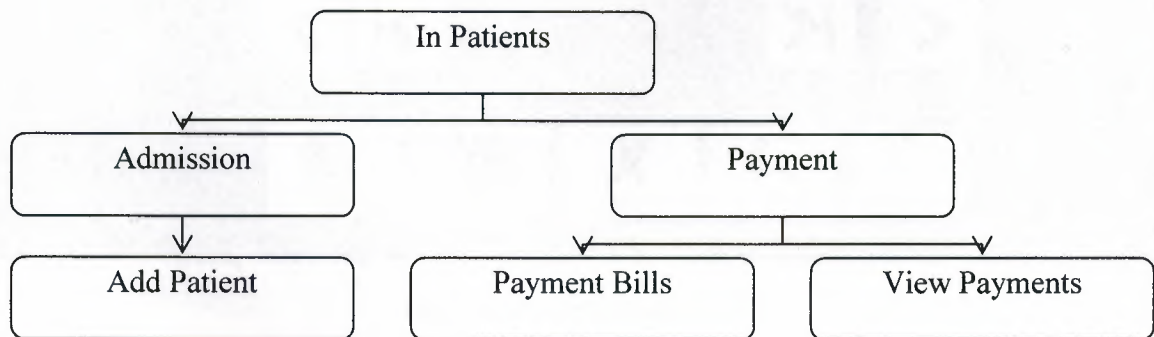
End If

db.Execute ("insert into OutPatientSerPayments values ('" & text2.Text & "','" &
Combo1.Text & "','" & Val (txtSer_Amount.Text) & "','" & str2 & "','" &
Text1.Text & "','" & txtBillNo.Text & "','" & txtBillDate.Text & "','" &
txtNetValue.Text & "')")

MsgBox "Payment Complete", vbOKOnly + vbInformation, "Payment"

```

4.4.2 In Patients



4.4.2.1 Admission

4.4.2.1.1 Add In Patient

This form is designed to register a patient that wants to stay in the hospital for an operational purpose, and assign a room for him/her. The form can add, edit, delete, view all in patients and search for a particular one.

Patient Details

Patient ID: IP1

First Name:	Salam	Last Name:	Abd Al-Amir
Address:	Lefkosa	Telephone Number:	0392345555
Mobile Number:	9533849678	Date of Add:	21.May.07
Age:	21	Blood Type:	A+
Date of Birth:	21.May.86	Gender:	Male
Diagnosis:	A possible cancer	Notes:	Smoker the ping may not work o
Room Number:	10		

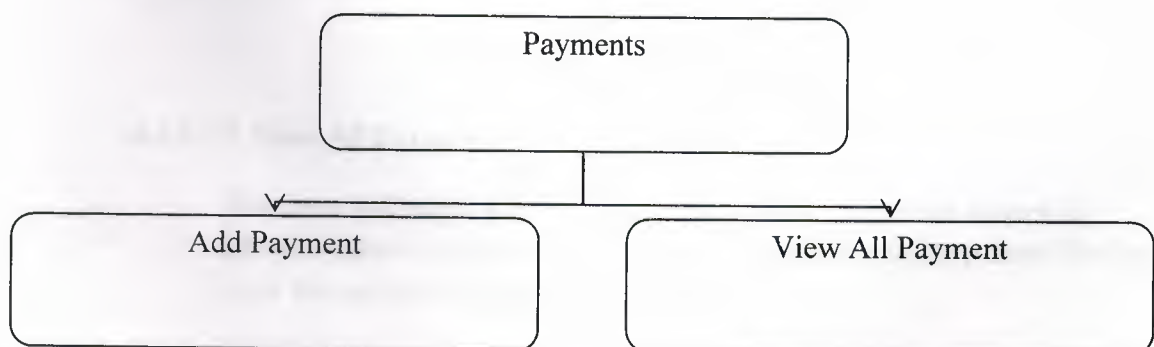
Navigator

Control

Add Edit Delete View All Search

Figure 4.14: Add in patient form

4.2.2.2 Payments



4.2.2.2.1 Add Payment

The form allows the user to generate a bill for in patient, the user can choose the desired in patient and calculate how much the patient should pay to the hospital.

The screenshot shows a software window titled "Inpatient Payments". At the top, there is a title bar with standard window controls. Below the title bar, the main title "Inpatient Payments" is centered. The form contains several input fields: "Bill Date" with the value "21.May.07", "Patient Name" with a dropdown menu showing "Salam", "Patient ID" with the value "IP1", "Doctor Charges" with the value "5000", "Bill Amount" with the value "5000", "Discount" with the value "50", "Bill Number" with the value "IPB1", and "Net Value" with the value "2500". A "Payment" dialog box is open in the center, displaying an information icon, the text "Payment Complete", and an "OK" button. Below the main form, there is a "Payment Info" section with three radio buttons: "CASH" (selected), "CREDIT CARD", and "CHEQUE". Below these is a "BANK" label followed by an empty text input field. At the bottom of the window, there are two buttons: "SAVE" with a floppy disk icon and "CLOSE" with a large "X" icon.

Figure 4.15: inpatient payment form

4.2.2.2.2 View All Payments

The form will allow the user to view all the payments that related to specific patient, the user can choose the patient from a drop down list and view the payments related to that patient.

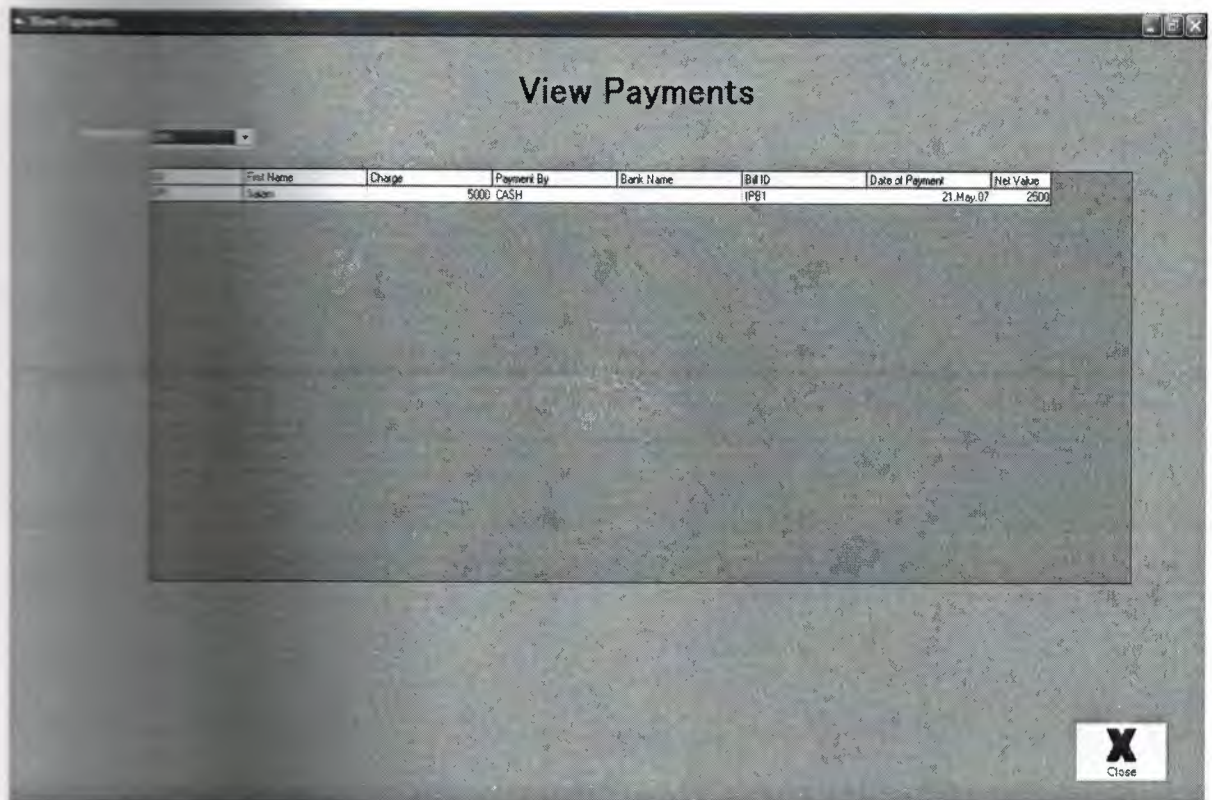


Figure 4.16: view all payment

The code for viewing payments in MS FlexGrid

```
Private Sub Combo1_Click ()
Set db = OpenDatabase (App.Path & "\hospital.mdb", False)

With MSFlexGrid1
.clear

.Rows = 1

.Cols = 8

.Col = 0

.Row = 0

.Text = "ID"

.Col = 1

.Row = 0

.Text = "First Name"

.Col = 2

.Row = 0
```



```

.Text = "Charge"
.Col = 3
.Row = 0
.Text = "Payment By"
.Col = 4
.Row = 0
.Text = "Bank Name"
.Col = 5
.Row = 0
.Text = "Bill ID"
.Col = 6
.Row = 0
.Text = "Date of Payment"
.Col = 7
.Row = 0
.Text = "Net Value"
.ColWidth(0) = 1500
.ColWidth(1) = 2000
.ColWidth(2) = 2000
.ColWidth(3) = 2000
.ColWidth(4) = 2000
.ColWidth(5) = 2000
.ColWidth(6) = 2000
End With

view_sql = "select * from inpatientpayments"
Set tb = db.OpenRecordset(view_sql)

While Not tb.EOF

```



```

MSFlexGrid1.AddItem tb("ip_id") & Chr(9) & tb("ip_fname") & Chr(9) & tb("d_charge") &
Chr(9) & tb("payment_by") & Chr(9) & tb("bank_name") & Chr(9) & tb("bill_no") &
Chr(9) & _

tb("bill_date") & Chr(9) & tb("net_value")

tb.MoveNext

Wend

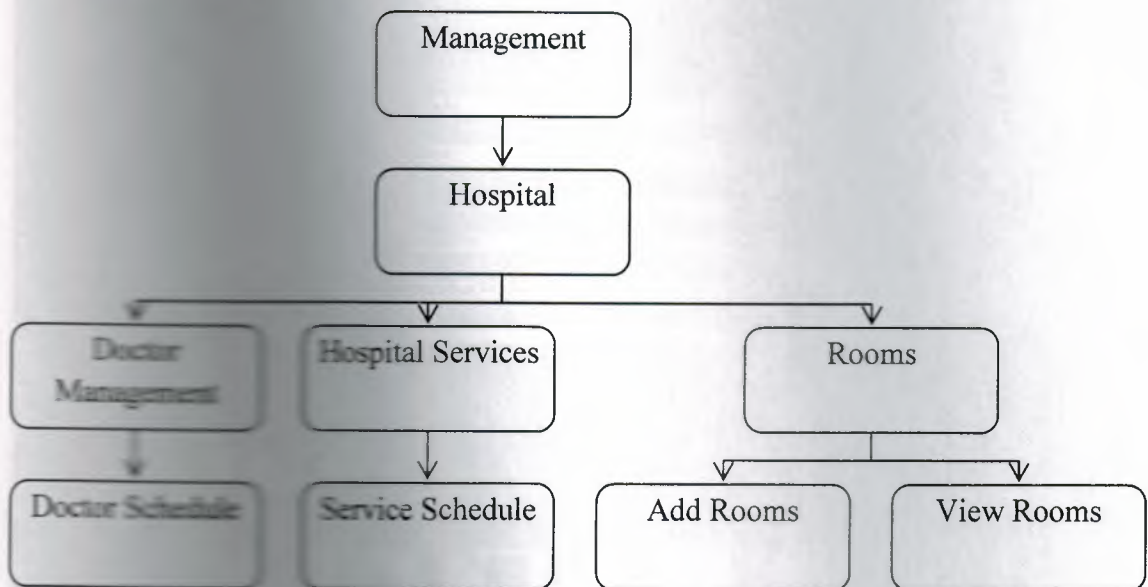
tb.close

db.close

End Sub

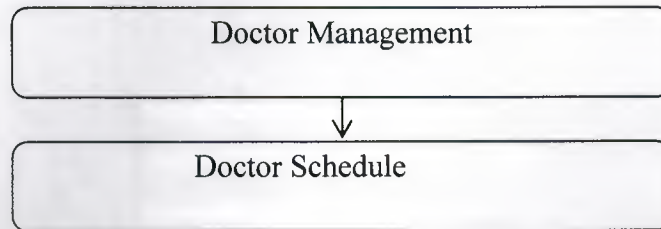
```

4.5 Management



4.5.1 Hospital

4.5.1.1 Doctor Management



4.5.1.1.1 Doctor Schedule

This form allows the user to set a schedule for the doctor, which days and time the doctor is available.

The screenshot shows a window titled "Doctor Schedule". The main title "Doctor Schedule" is centered at the top. Below it, there are two main sections: "Enter Data" on the left and "Display Data" on the right. The "Enter Data" section contains several input fields: "Schedule ID" with the value "DSch1", "Doctor ID" with a dropdown menu showing "Doc1", "Time In" with the value "09:00 AM", "Time Out" with the value "09:00 PM", "Available Days" with the value "Sun,Tue,Thu,Sat", and an empty "Available Time" field. The "Display Data" section contains a list of days with checkboxes: Sunday (checked), Monday (unchecked), Tuesday (checked), Wednesday (unchecked), Thursday (checked), Friday (unchecked), and Saturday (checked). Below these sections, there is a "Record Navigation" section with buttons for previous and next records. At the bottom, there is a "Record Operations" section with buttons for Add, Edit, Delete, and Close.

Figure 4.17: Doctor Schedule form

4.5.1.2 Services Management

4.5.1.2.1 Services Schedule

This form allows the user to set a schedule for the services, which days and time the services is available.

Figure 4.18: service schedule form

4.5.1.3 Rooms

4.5.1.3.1 Add Room

The form allows the user to add new room, edit existing room and delete room.

Figure 4.19: Add room form

4.5.2 Employee Management

4.5.2.1 Add Employee

This form allows the user to add new employee, edit existing employee, delete an employee, search for a particular employee and view all the employees.

The screenshot shows a window titled "Employee Details" with a form for adding or editing employee information. The form fields are as follows:

Field	Value
Employee ID	Doc1
Employee First Name	Hayan
Employee Last Name	Saleh
Address	Iefkosa
Telephone 1	6444914
Telephone 2	8483423
Date of Add	31, Mar, 05
Sex	male
Employee Type	doctor
Basic Salary	1500

Below the form is an "Operations" section with the following buttons:

- Add (Icon: Person with plus sign)
- Edit (Icon: Pencil)
- Search (Icon: Magnifying glass)
- Delete (Icon: X)
- View All (Icon: Document with magnifying glass)
- Close (Icon: X)

Figure 4.20: Add employee form

CONCLUSION

This software prepared using Microsoft Visual Basic 6.0 and Microsoft Access. Both of them are powerful software.

Visual Basic 6.0 is a flexible language. It has many tools to help programmer to create an applications.

Access is also a flexible program to a relational database and to connect tables between each other using a relationships.

Using a DAO in VB6 is very useful because it connects the application with the any database program like Access and Oracle.

REFERENCES

- [1]. <http://www.startvb.com/vb.aspx>
- [2]. Microsoft Developer Network for Microsoft Visual Studio 6.0
- [3]. Cary N. Prague and Michael R. Irwin (2002) Microsoft Access Bible
- [4]. James Hoffman (1999) Introduction to SQL
- [5]. http://en.wikipedia.org/wiki/Open_Database_Connectivity