

Available online at www.sciencedirect.com



Procedia Technology 00 (2011) 000–000

**Procedia
Technology**

www.elsevier.com/locate/procedia

WCIT 2012

University Hospital Management System Software Using MS Visual Studio (C#)

Jamshed Rahimov ^a*, Mustafa Ilkan ^b

^aEastern Mediterranean University, Famagusta, Turkish Republic of North Cyprus via Mersin 10, Turkey

^bEastern Mediterranean University, Famagusta, Turkish Republic of North Cyprus via Mersin 10, Turkey

Abstract

Effective means of software programming to implement it to the university hospital management system is presented. The software application is created using MS Visual Studio (C#), with .NET Framework 4 and fully adapted to the university hospital system, combining all the necessary fields. The steps and results will be presented in screenshots and data tables. From the presented results it's clear that with manipulation of the software, the hospitality management will be more effective and less time consuming. Also with the application of this system an auto control, patients/doctors and other hospital staff satisfaction will be satisfied. This will indirectly improve and develop the staff motivation and efficiency. Management will be much more easier.

Keywords: Hospital Management System; Application using .NET Framework; Hospital Software Manipulation; C#;

1. Introduction

1.1. Aim of the project

The main goal of this project is to automate university hospital management system, so all the information about patients and doctors could be stored in a local database. By automation it is meant faster and easier access to patients' and hospital staff's necessary information. All the user needs to do is to enter a unique identity number into the form.

Secondly, reducing the amount of the paperwork, which most hospitals nowadays are facing, is targeted. It will make job easier and faster. There is no need for hardcopies of patients' treatment history. No need for archive room, which may not be enough later on, as the amount of patients increases. With today's technology user needs a small room, or even a part of his/her office to keep the hard drives, and

* Jamshed Rahimov. Tel.: +90-542-876-9829
E-mail address: jamshed.rahimov@gmail.com

keep the backup of it somewhere else in a case of flood, fire or other natural disasters, which is not possible to do with hardcopies from archive rooms.

It can be concluded that using the software or in another word moving the hospital to automation system will make the hospital maintenance easier, faster and safer.

1.2. Capabilities of the software and project scope

The software maintains two level users: administrator – head doctor(s), user – doctors and assistants. Users are able to add new patients, enter treatment history, take a quick patient reports, set appointments, and see the list of inpatients (university patients) and outpatients (non-university patients). However any of these forms can be limited by administrator in terms of information security.


More details about the administrators' and users' privileges will be shown in Forms Screenshots and Explanations part.

1.3. Development

This project is written by the desktop application in Microsoft Visual Studio .NET Framework environment based on C#.

2. Form explanations

2.1. "Login" form

When the user starts the application, for security reasons *username* and *password* will be asked. If *Login* button pressed without filling one of those or both fields, error icon  will be shown at the end of the unfilled textbox. Error message will be shown, when mouse cursor is on the error icon. If the username and password will be entered wrong more than 3 times, application will be closed.

Usernames and passwords are stored in *users* table.

2.2. "Main Menu" form

After successful login, application gets the user rights from the current user's row in database, and shows *Main Menu* form accordingly. If the user has all the rights, basically it makes him/her an administrator, which sees the form with all buttons and menu items enabled.

User rights are stored in *users* table, in *rights* column, where all the user's rights are stored in one line separated with comas.

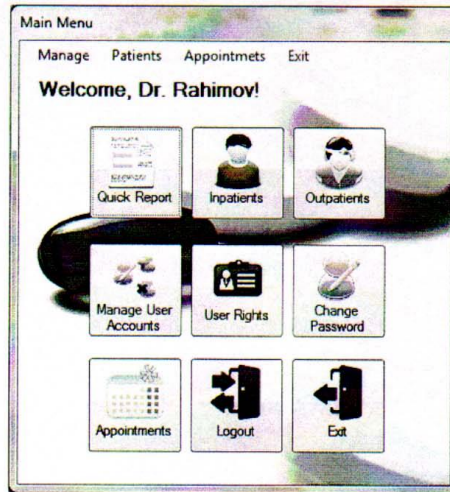



Fig. 1. Main Menu Form

2.3. "Change Password" form

If the user wants to change the password, s/he should enter an old one first (for verification of account), and twice new password. Again like in *login form*, all fields will be checked for correctness. The conditions are: old password must be correct; last two fields should be matched with each other; and the new password cannot be less than three characters. If any of these requirements are not handled, error icon  will be shown in the end of the field with the error message when the user brings cursor over the icon.

2.4. "User Accounts" form

User Accounts form has six necessary fields for creating a new user account: a unique *username*, *password*, *name*, *surname*, *gender* and *status* (status can be a doctor, assistant, nurse, etc., those who can have an access to the application). Also it's possible to *delete* and *update* the users account.

After opening the form, it automatically loads on data grid all the available users. And when any of those rows are clicked, the information is loaded on above listed six fields, so it would be possible to see what to modify or delete.

This form includes all columns of *users* ' table except *rights*. Coming page will be about the user rights column. All the entries of this form are saved in *users* table.

2.5. "User Rights" form

When the user is chosen from combo box, user rights for the chosen user will be displayed in checkboxes. After adding/removing some rights, click the *Save* button and re-login of form to apply the changes.

After saving, the rights for the chosen user are modified in *users* table.

2.6. "Add New Patient" form

Add New Patient form has seven necessary fields for creating a new patient: *UID* (*University Identity Number*), *name*, *surname*, *gender*, *country*, *birthdate* and *group*. The *group* field is used to determine the group category of the patient – student, instructor, driver, assistant, etc. Also it's possible after entering the UID and pressing *search* button to check rather this person is registered in system as a patient or not. If UID is found in DB (DataBase), the rest of the information will be shown in the form; else message box shows a message that this UID is not registered.

All the entries of this form are saved in *patients* table.

2.7. "Inpatients" form

This form is used for entering treatment records of the patients. When the *UID* is entered, form will automatically check and load the information of a patient. After observing a patient, doctor will enter *symptoms*, *diagnosis*, *given medicine* and *additional notes* for that treatment. To that treatment information is also added current date and name of the user, where name of the user is "*assigned doctor*" in data table.

2.8. "Quick Report" form

Quick Report form is used to take a quick report of inpatient. Only UID field needs to be entered to find the patient. After entering a correct UID and pressing *Search* button, form displays patient's personal information. When the needed patient is found, print button is enabled in order to print a report.

The patient's report will be in this format:

Report given by: \$Current Doctor\$ on \$DateTime.Now\$

Patient's Personal Info:

University ID: \$uid\$

Fullname: \$pname\$ + \$psurname\$

Age: \$age\$

Gender: \$pgender\$

Country: \$country\$

Category: \$category\$

Treatment History:

(in loop, till treatment history for this UID finishes)

Date: \$date\$

Referred Doctor: \$assigndoc\$

Assigned Doctor: \$refdoc\$

Symptoms: \$symptoms\$

Diagnosis: **\$diagnosis\$**
Given Medicine: **\$medicine\$**
Additional Notes: **\$notes\$**

2.9. "Outpatients" form

Outpatient form is used in situations when a patient is not a member of university. The only difference between outpatients and inpatients forms is that in this form personal information is entered as well before entering treatment details.

2.10. "Add New Appointment" form

Add New Appointment form has six main fields for creating a new appointment: *day*, *start time*, *duration*, *location*, *subject* and *notes*. There is one more field which is disabled – *appointment ID* text box with *load appointment* button coming right after it. It is needed to update or delete an appointment. To do this *load* button is pressed and on the right hand side all appointments of current user are loaded. When an appointment is chosen from the data grid, all information of the appointment are copied to text boxes including and *ID* text box will be an ID of the chosen appointment. When it's there, user can delete or update his/her appointment.

All appointments are saved on a separate data table called *appointments*.

2.11. "List Appointments" form

List Appointments form works like filtering of appointments. This form has two date pickers – for *start date* and *end date*. When the *list* button is clicked, it will check the dates first, so that the day count from current date and start date must be less than that the day count from current date and the end date. If the dates were entered correctly – in data grid view appointments are loaded, else message box with the error message is displayed.

3. Backbone of the coding part

This section includes the main coding parts which were used for creating the application

3.1. Adding items to menu strip

By default all menu strips are visible in main menu form. When the user is logging in, before showing main menu, user rights is checked and the visibility of menu item(s) which user doesn't own (-1) is/are set to false.

```
foreach (ToolStripMenuItem p in mStrip.Items)
{
    if (rights.IndexOf(p.Text) == -1)
    {
        p.Visible = false;
        continue;
    }
    foreach (ToolStripMenuItem m in p.DropDownItems)
        if (rights.IndexOf(m.Text) == -1)
```

```
        m.Visible = false;
    }
}
```

3.2. Filling data grid (example with "appointments" table)

Form connects to database by using connection string (CString) stored in " Misc.cs" . file. When connection state is open, sql command selects from a table all rows (*) or by choosing one-by-one necessary rows. Data adapter fills data table, after data table equals to data grid view list.

```
SqlDataAdapter da = new SqlDataAdapter();
DataTable dt = new DataTable();
SqlConnection cnn = new SqlConnection(Misc.CString);

int numOfTries = 0;
do
{
    numOfTries++;
    if (cnn.State != ConnectionState.Open) cnn.Open();
    System.Threading.Thread.Sleep(1000);
} while ((cnn.State != ConnectionState.Open) || (numOfTries > 10));

if (cnn.State != ConnectionState.Open)
{
    MessageBox.Show("Connection Error!");
    return;
}

da.SelectCommand = new SqlCommand("select * from appointments", cnn);
da.Fill(dt);
dgrdViewList.DataSource = dt;
```

3.3. Finding age from birthdate

DateTime variable now is set to today's date. Then year of the current date is subtracted from the birthdate year of patient from data table. The result is set to integer age.

```
DateTime now = DateTime.Today;
int age = now.Year - Convert.ToDateTime(rd["bdate"]).Year;
if (now < Convert.ToDateTime(rd["bdate"]).AddYears(age))
    age--;
```

3.4. Updating data table (example with "patients" table)

Before updating, form connects to database by using connection string from a file, which was described above. Then sql code selects from a table necessary rows, if UID exist in table, confirmation message box is shown. When "Yes" is chosen data table's information is replaces with the information from the form.

```
SqlCommand cmd = new SqlCommand();
cmd.Connection = cnn;
cmd.CommandType = CommandType.Text;
cmd.CommandText = "select * from patients where uid=@uid";
cmd.Parameters.Add("@uid", SqlDbType.VarChar).Value = txtUID.Text;
SqlDataReader rd = cmd.ExecuteReader();
```

```

if (rd.HasRows)
{
    rd.Close();
    if (MessageBox.Show("Username exists do you want to overwrite it?", "Update",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2) !=
        DialogResult.Yes)
        return;

    cmd.Parameters.Clear();
    cmd.CommandText = "update patients set uid=@uid, pname=@pname, psurname=@psurname,
        pgender=@pgender, category=@group, bdate=@bdate, country=@country where uid=@uid";

    cmd.Parameters.Add("@uid", SqlDbType.VarChar).Value = txtUID.Text;
    cmd.Parameters.Add("@pname", SqlDbType.VarChar).Value = txtName.Text;
    cmd.Parameters.Add("@psurname", SqlDbType.VarChar).Value = txtSurname.Text;

    cmd.Parameters.Add("@pgender", SqlDbType.VarChar).Value = cboGender.Text;
    cmd.Parameters.Add("@group", SqlDbType.VarChar).Value = cboGroup.Text;
    cmd.Parameters.Add("@bdate", SqlDbType.Date).Value = dtBirthdate.Text;
    cmd.Parameters.Add("@country", SqlDbType.VarChar).Value = txtCountry.Text;

    int i = cmd.ExecuteNonQuery();
    if (i > 0)
    {
        MessageBox.Show("Record Updated");
        clrScr();
    }
}

```

3.5. Printing (example with "outpatients" form)

When the print button is pressed, form checks if all necessary fields are field or not. If result is positive print page dialog is called. Else message box with error message is shown.

```

private void btnPrint_Click(object sender, EventArgs e)
{
    if (txtName.Text != "" && txtSurname.Text != "" && cboGender.Text != "" &&
        rtxtAddress.Text != "" && txtDiagnosis.Text != "")
    {
        PrintDialog printDialog = new PrintDialog();
        PrintDocument printDocument = new PrintDocument();
        printDialog.Document = printDocument;
        printDocument.PrintPage += new PrintPageEventHandler(printDocument_PrintPage);
        DialogResult result = printDialog.ShowDialog();
        if (result == DialogResult.OK)
        {
            printDocument.Print();
        }
    }
    else
    {
        MessageBox.Show("Fill the fields with asterisk (*) sign");
        return;
    }
}

```

The final step is printing. Form declares font type and sets start positions for X axis and Y axis. Then it prints each line one by one.

```

void printDocument_PrintPage(object sender, PrintPageEventArgs e)
{
    Graphics graphic = e.Graphics;
    Font font = new Font("Courier New", 12);
    float fontHeight = font.GetHeight();
    int startX = 10;
    int startY = 10;
    graphic.DrawString("Report given by " + user + " on " + DateTime.Now, new Font("Courier
new", 14, FontStyle.Underline), new SolidBrush(Color.Black), startX, startY);
    graphic.DrawString("Outpatient's Personal Info", new Font("Courier New", 14,
FontStyle.Bold), new SolidBrush(Color.Black), startX, startY + 40);
    graphic.DrawString("Name: ".PadRight(20) + txtName.Text, new Font("Courier New", 12),
new SolidBrush(Color.Black), startX, startY + 60);
    graphic.DrawString("Surname: ".PadRight(20) + txtSurname.Text, new Font("Courier New",
12), new SolidBrush(Color.Black), startX, startY + 80);
    graphic.DrawString("Gender: ".PadRight(20) + cboGender.Text, new Font("Courier New",
12), new SolidBrush(Color.Black), startX, startY + 100);
    graphic.DrawString("BirthDate: ".PadRight(20) + dtpBirthdate.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 120);
    graphic.DrawString("Current Address: ".PadRight(20) + txtAddress.Text, new
Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 140);
    graphic.DrawString("Country: ".PadRight(20) + txtCountry.Text, new Font("Courier New",
12), new SolidBrush(Color.Black), startX, startY + 160);
    graphic.DrawString("Assigned Doctor: ".PadRight(20) + lblAssignDoc.Text, new
Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 180);
    graphic.DrawString("Referred Doctor: ".PadRight(20) + cboRefDoc.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 200);
    graphic.DrawString("Symptoms: ".PadRight(20) + rtxtSymptoms.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 220);
    graphic.DrawString("Diagnosis: ".PadRight(20) + txtDiagnosis.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 240);
    graphic.DrawString("Given Medicine: ".PadRight(20) + txtMedicine.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 260);
    graphic.DrawString("Additional Notes: ".PadRight(20) + rtxtNotes.Text, new Font("Courier
New", 12), new SolidBrush(Color.Black), startX, startY + 280);
}
    
```

4. Data tables

Table 1 Users table

Field Name	Data Type	Allow nulls	Description
uname	varchar(10)	no	PK
pwd	varchar(50)	no	
rights	varchar(250)	yes	
name	varchar(30)	no	
surname	varchar(30)	no	
gender	varchar(6)	no	
status	varchar(15)	no	

Table 2 Patients table

Field Name	Data Type	Allow nulls	Description
pid	int	no	PK AI
uid	varchar(6)	no	
pname	varchar(50)	no	
psurname	varchar(50)	no	
pgender	char(1)	no	
category	varchar(20)	no	
bdate	date	yes	
country	varchar(50)	yes	

Table 3 Inpatients table

Field Name	Data Type	Allow nulls	Description
ipid	int	no	PK AI
uid	varchar(6)	no	
ipfullname	varchar(100)	no	
ipage	int	no	
ipgender	char(1)	no	
ipcategory	varchar(20)	no	
medicine	varchar(50)	yes	
refdoc	varchar(100)	yes	Referred Dr.
symptoms	varchar(250)	yes	
diagnosis	varchar(50)	no	
notes	varchar(250)	yes	
uname	varchar(100)	no	Assigned Dr.
date	date	no	To record treatment date

Table 4 Outpatients table

Field Name	Data Type	Allow nulls	Description
opid	int	no	PK AI
opname	varchar(50)	no	
opsurname	varchar(50)	no	
opgender	char(1)	no	
opbdate	date	no	
currentaddress	varchar(250)	no	
country	varchar(50)	yes	
refdoc	varchar(100)	yes	Referred Dr.
symptoms	varchar(250)	yes	
diagnosis	varchar(250)	no	
medicine	varchar(50)	yes	
notes	varchar(250)	yes	
uname	varchar(100)	no	Assigned Dr.
date	date	no	To record treatment date

Table 5 Appointments table

Field Name	Data Type	Allow nulls	Description
aid	int	no	PK AI
day	date	no	
time	datetime	no	
duration	int	yes	
location	varchar(50)	yes	
subject	varchar(50)	no	
notes	varchar(250)	yes	
uname	varchar(10)	no	To know to whom belongs appointment(s)

5. Conclusion

Before starting this study, previous two similar works have been found and analyzed. The first paper has very detailed data flow diagrams and data tables, well organized forms created by using Visual Basic. However, it is complicated and made more for big cities rather than universities or small town hospital systems. For example there are some forms which are too complicated, hard to understand and get used to them.

Second project is the reverse comparing with previous one. It is created using C# and made as simple as possible, which is good, but it also has some disadvantages. For this project except general explanations and form screenshots nothing else is shown - no data tables and data flow diagrams.

Also in both projects some missing parts have been found, like detailed explanation of each form and how the errors are handled. Moreover in none of them is shown key coding parts which are extremely necessary, especially while dealing with this kind of projects.

After reviewing the materials and analyzing positive and negative sides, current study is done. The aim of the present work is to show the effective use of the software created by using MS Visual Studio (C#). The application is not broad software which could satisfy all needs of the hospitals, but would be very useful for university hospitals and health centers.

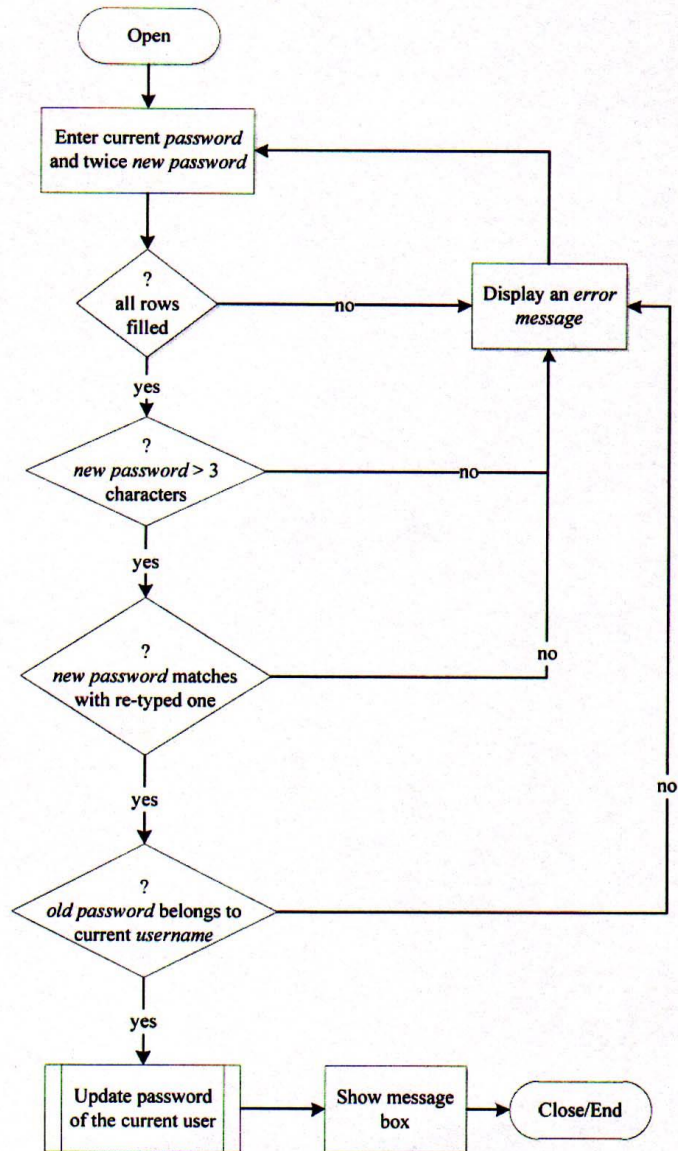
By using this type of application in hospitals, processes can be done faster, easier and safer. Screenshots above show that the forms interface are easy to understand and use. User doesn't have to have extraordinary knowledge to use the application. Data tables prove that database covers the most necessary fields which are needed for daily use of health centers.

While preparing this work two hospitals are visited to make sure what are the basics which they need when a patients comes. Also it was a very useful experience for me while implementing some parts of this work in MS Visual Studio and to make it work.

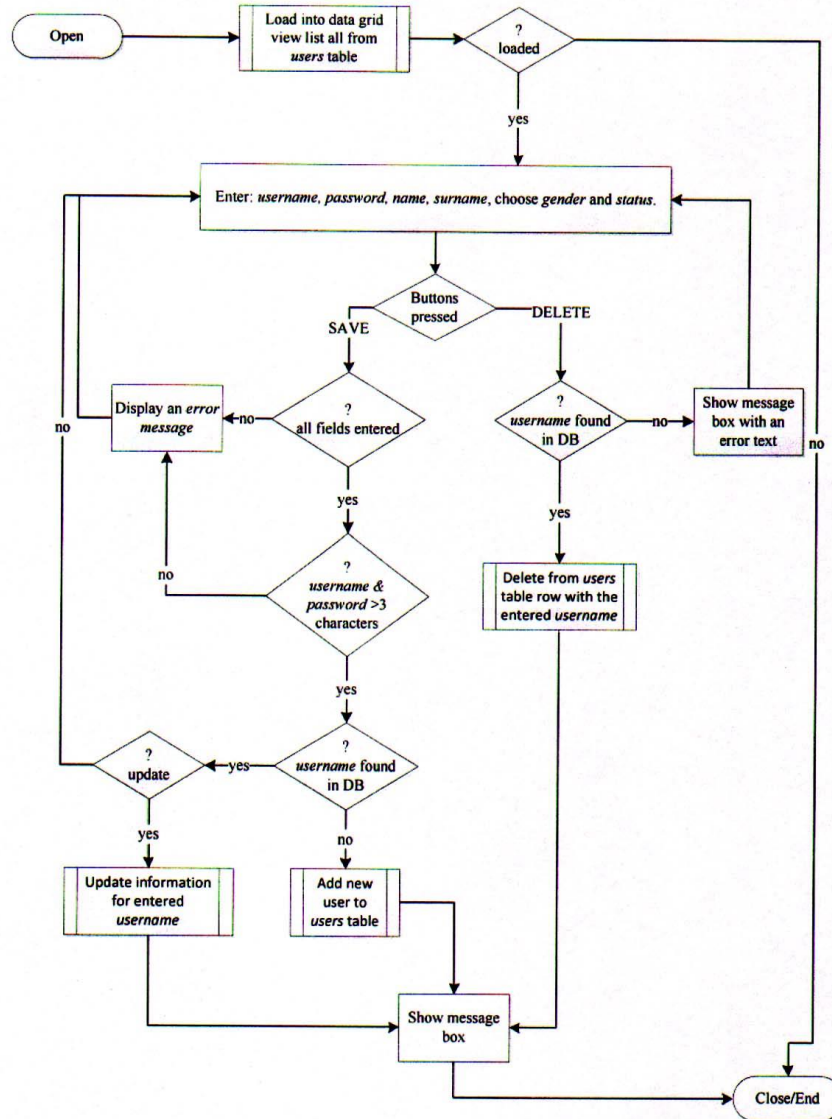
References

- [1] Touseef Ahmed, A.B. Qureshi. Hospital Management System;
- [2] M. Chitkara, N. Khandelwal, A. Chaporkar. Hospital Management System. Available from: <http://www.slideshare.net>;
- [3] Video course by user "andyruncinman" from YouTube, March 2010. Available from: <http://goo.gl/Dcd3P>;
- [4] C# Tutorials. Available from: <http://goo.gl/iulWB>;
- [5] Обучающие уроки по C#. Available from: <http://goo.gl/qxFNj>;
- [6] Printing in C#, January 26, 2010. Available from: <http://goo.gl/3KEQE>;
- [7] Печать документа, April 23, 2009. Available from: <http://goo.gl/rAS8v>;

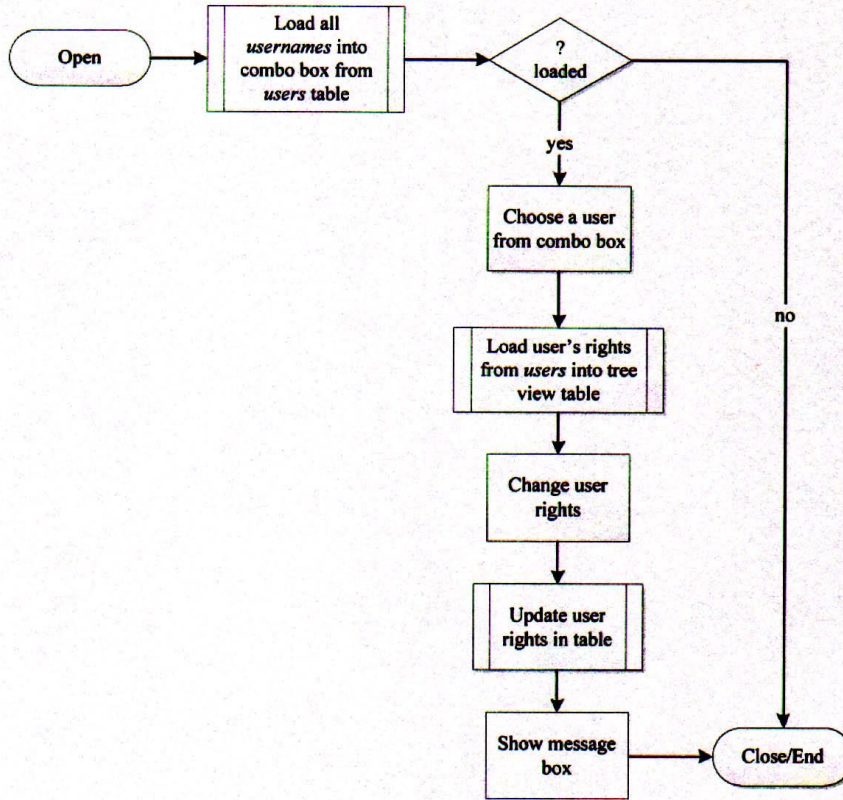
2. Change Password flowchart



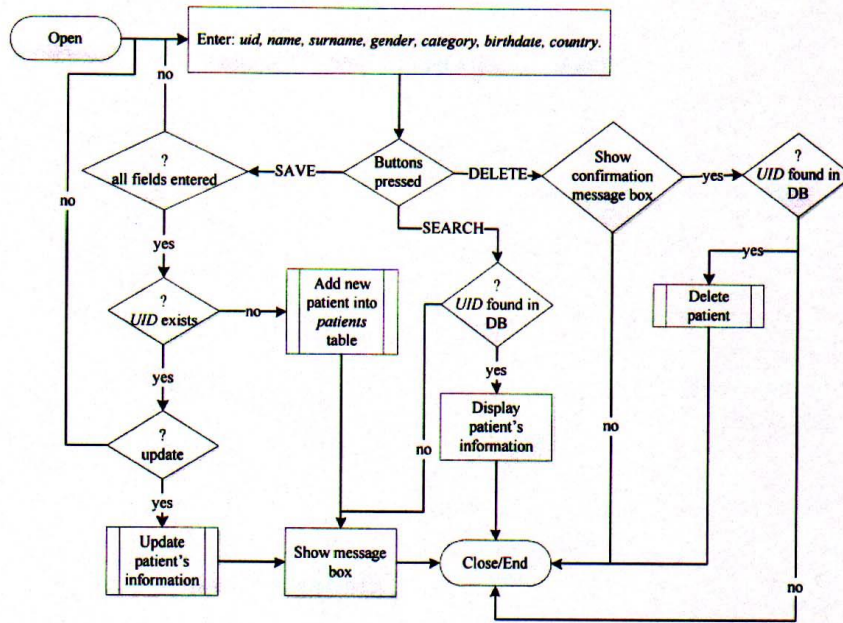
3. Manage Users flowchart



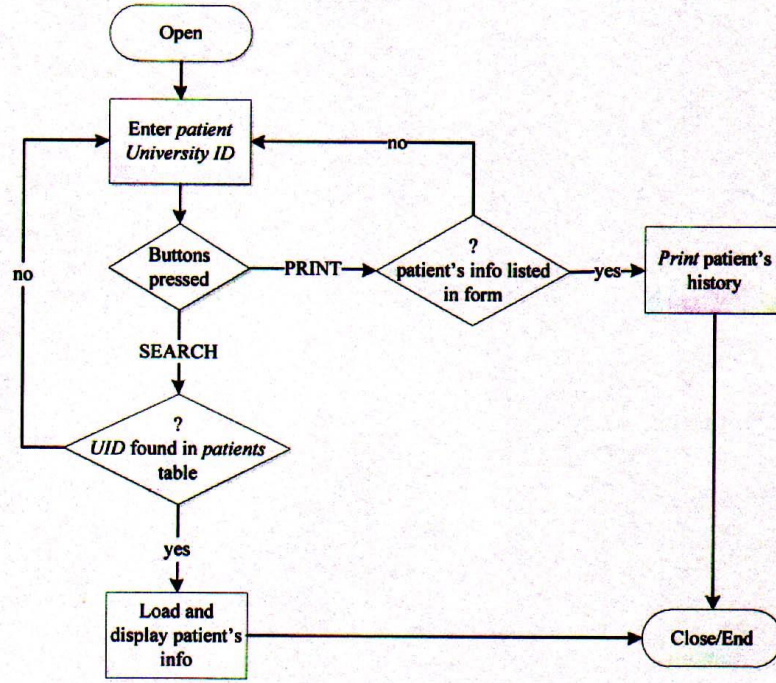
4. User Rights flowchart



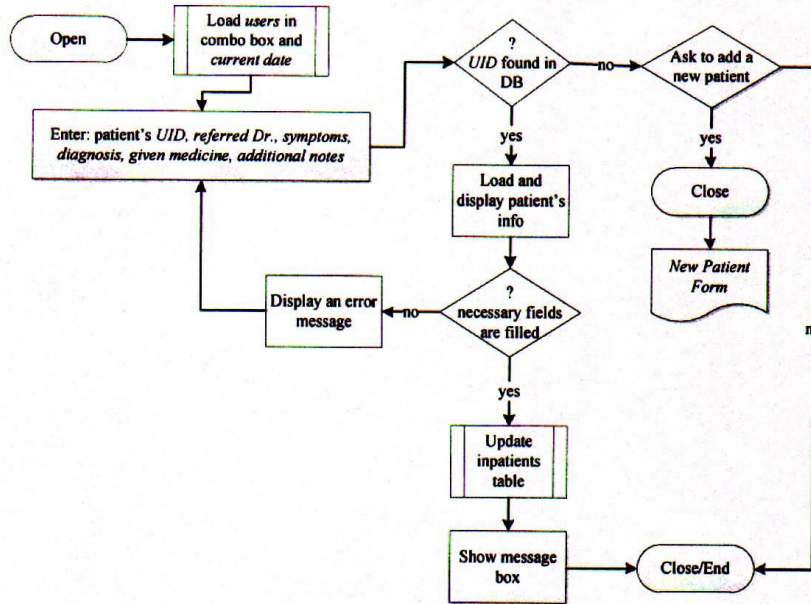
5. New Patient flowchart



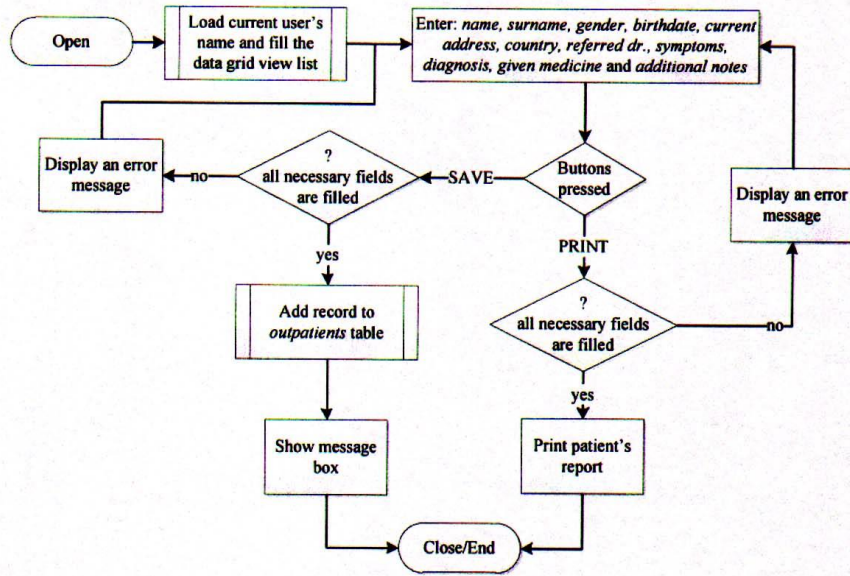
6. Quick Report flowchart



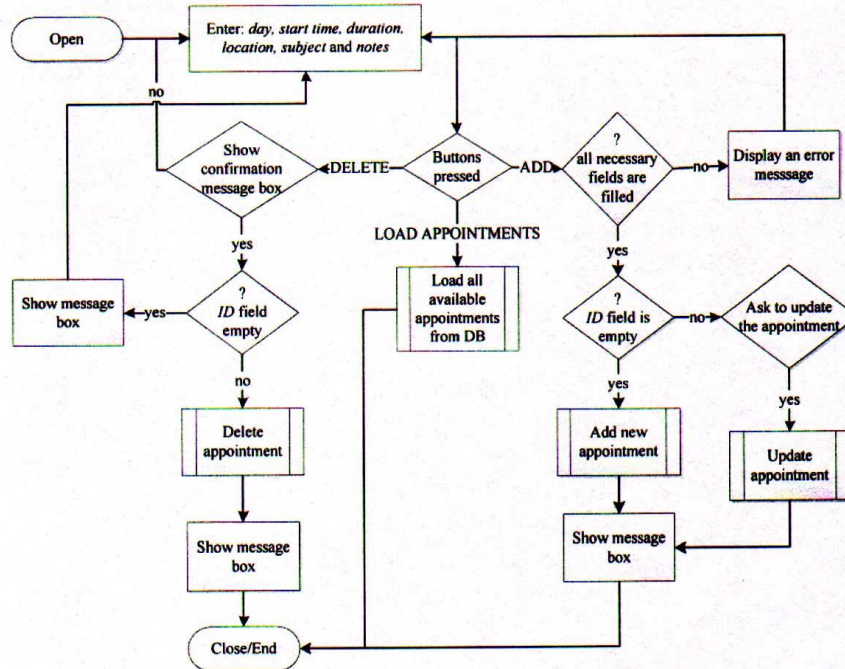
7. Inpatients flowchart



8. Outpatients flowchart



9. New Appointment Flowchart



10. List Appointments flowchart

