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

Jamsheed Rahimov a *, Mustafa Ilkan b

aEastern Mediterranean University, Famagusta, Turkish Republic of North Cyprus via Meram 10, Turkey
bEastern Mediterranean University, Famagusta, Turkish Republic of North Cyprus via Meram 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
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.
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: $fname$ + $surname$
Age: $age$
Gender: $gender$
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.

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

```csharp
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);
grvViewList.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.

```csharp
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.

```csharp
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=@category, 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("@pgender", SqlDbType.VarChar).Value = cboGender.Text;
    cmd.Parameters.Add("@category", SqlDbType.VarChar).Value = cboCategory.Text;
    cmd.Parameters.Add("@bdate", SqlDbType.Date).Value = dtBirthDate.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 filled 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 == "" && txtAddress.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 startY = 10;
    int startX = 10;
    graphic.DrawString("Report given by " + user + " on " + DateLine.Now, new Font("Courier New", 14, FontStyle.Oblique), 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("Patient: " + txtName.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 60);
    graphic.DrawString("Surname: " + txtSurname.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 80);
    graphic.DrawString("Gender: " +欻Attend.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 100);
    graphic.DrawString("BirthDate: " + dateTimePicker1.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 120);
    graphic.DrawString("Current Address: " + txtAddress.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 140);
    graphic.DrawString("Referred Doctor: " + txtRefDoc.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 180);
    graphic.DrawString("Diagnosis: " + txtDiagnosis.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 220);
    graphic.DrawString("Additional Notes: " + txtNotes.Text, new Font("Courier New", 12), new SolidBrush(Color.Black), startX, startY + 260);
}

4. Data tables

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Allow nulls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int</td>
<td>no</td>
<td>PK</td>
</tr>
<tr>
<td>user</td>
<td>varchar50</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>rights</td>
<td>varchar50</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>varchar50</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>surname</td>
<td>varchar50</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>varchar50</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>varchar5</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
### Table 2 Patients table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Allow nulls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>int</td>
<td>no</td>
<td>PK AI</td>
</tr>
<tr>
<td>uid</td>
<td>varchar(6)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>pname</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>psname</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>gender</td>
<td>char(1)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>varchar(20)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>bdate</td>
<td>date</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>country</td>
<td>varchar(50)</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3 Inpatients table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Allow nulls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>int</td>
<td>no</td>
<td>PK AI</td>
</tr>
<tr>
<td>uid</td>
<td>varchar(6)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ipatientname</td>
<td>varchar(100)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>ipage</td>
<td>int</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>igender</td>
<td>char(1)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>icategory</td>
<td>varchar(20)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>medicine</td>
<td>varchar(30)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>refdoctor</td>
<td>varchar(100)</td>
<td>yes</td>
<td>Referred Dr.</td>
</tr>
<tr>
<td>symptoms</td>
<td>varchar(250)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>diagnosis</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>notes</td>
<td>varchar(250)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>surname</td>
<td>varchar(100)</td>
<td>no</td>
<td>Assigned Dr.</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>no</td>
<td>To record treatment date</td>
</tr>
</tbody>
</table>

### Table 4 Outpatients table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Allow nulls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opid</td>
<td>int</td>
<td>no</td>
<td>PK AI</td>
</tr>
<tr>
<td>opname</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>opssname</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>opsgender</td>
<td>char(1)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>opbdate</td>
<td>date</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>currentaddress</td>
<td>varchar(250)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>country</td>
<td>varchar(50)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>refdoctor</td>
<td>varchar(100)</td>
<td>yes</td>
<td>Referred Dr.</td>
</tr>
<tr>
<td>symptoms</td>
<td>varchar(250)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>diagnosis</td>
<td>varchar(250)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>medicine</td>
<td>varchar(50)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>notes</td>
<td>varchar(250)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>surname</td>
<td>varchar(100)</td>
<td>no</td>
<td>Assigned Dr.</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
<td>no</td>
<td>To record treatment date</td>
</tr>
</tbody>
</table>
### Table 5 Appointments table

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Allow nulls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aid</td>
<td>int</td>
<td>no</td>
<td>PR, AI</td>
</tr>
<tr>
<td>day</td>
<td>date</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>datetime</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>duration</td>
<td>int</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>location</td>
<td>varchar(50)</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>subject</td>
<td>varchar(50)</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>notes</td>
<td>varchar(250)</td>
<td>yes</td>
<td>To know to whom belongs appointment(s)</td>
</tr>
<tr>
<td>uname</td>
<td>varchar(10)</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>

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

2. Change Password flowchart

Open

Enter current password and twice new password

? all rows filled
  yes
  no

? new password > 3 characters
  yes
  no

? new password matches with re-typed one
  yes
  no

? old password belongs to current username
  yes
  no

Update password of the current user

Show message box

Close/End
3. Manage Users flowchart

Open → Load into data grid → view list all from users table → loaded → yes → Enter: username, password, name, surname, choose gender and status.

- Display an error message
- all fields entered → username & password >5 characters → username found in DB → yes → update → username found in DB → yes → Update information for entered username → no → Add new user to users table → Show message box
- no → Show message box
- username found in DB → no → Show message box

Close/End
4. User Rights flowchart

Open → Load all usernames into combo box from users table

? loaded

yes → Choose a user from combo box

no →

Load user's rights from users into tree view table

Change user rights

Update user rights in table

Show message box → Close/End
5. New Patient flowchart

Open → Enter: uid, name, surname, gender, category, birthdate, country.

? all fields entered? → SAVE

? UID exists? → UID found in DB

? update? → Update patient's information

Show confirmation message box → yes → UID found in DB

Delete patient → yes

Show message box → no

Close/End → no

Add new patient into patients table → yes

Display patient's information → no
6. Quick Report flowchart

Open

Enter patient University ID

no

Buttons pressed

PRINT

patient's info listed in form

yes

Print patient's history

SEARCH

UID found in patients table

yes

Load and display patient's info

Close/End
7. Inpatients flowchart

Open → Load users in combo box and current date

Enter: patient’s UID, referred Dr., symptoms, diagnosis, given medicine, additional notes

Display an error message

UID found in DB

Load and display patient’s info

necessary fields are filled

Update inpatients table

Show message box

Ask to add a new patient

Close

New Patient Form
8. Outpatients flowchart

- Open
- Load current user’s name and fill the data grid view list
- Enter: name, surname, gender, birthdate, current address, country, referred dr., symptoms, diagnosis, given medicine and additional notes
- Display an error message
  - no
  - all necessary fields are filled
    - yes
      - Add record to outpatients table
      - Show message box
      - Close/End
    - no
      - Display an error message
- Buttons pressed
- PRINT
  - yes
    - all necessary fields are filled
      - yes
        - Print patient’s report
      - no
        - Close/End
    - no
      - Display an error message
9. New Appointment Flowchart

Open

Enter day, start time, duration, location, subject and notes

Show confirmation message box

- yes
  - Show message box

- no
  - DELETE

Buttons pressed

- yes
  - ADD
  - all necessary fields are filled
    - no
      - Display an error message
    - yes
      - LOAD APPOINTMENTS

- no
  - Display an error message

Show message box

Close/End
10. List Appointments flowchart

Open

Choose start date & end date

? start date < end date

yes

Load appointments between the dates

no

Display an error message

Close/End

List appointments in data grid view list