

Hospital Management System

By

Name: Md.Soleyman khan

ID: 2012-2-50-005

Name: Md. Ashrafal Alam Siddique

ID: 2012-3-50-003

Supervised By

Dr. Md. Arifuzzaman

Assistant Professor

Electronics and Communications Engineering

East West University

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of
Bachelors of Science in Electronics and Communications Engineering

to the



Department of Electronics and Communications Engineering

East West University

Dhaka, Bangladesh

December, 2016

Abstract

In this project we developed a medical software which is highly useful for any Medical. As modern civilization requires digitalization, as a result the need of software is increasing everywhere, it not only saves our time but also gives fastest output.

We analyzed users' overview and developed this software for the sake of both patient and doctor.

Secondly, both of them have high amount of benefit by utilizing this software. As I already said this is time consuming, it is also capable of saving all kind of data from patients. We also added for admin label extra facility; all users are not capable to so all data.

Declaration

I hereby declare that, this project has been done under ETE498 and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication.

Md. Soleyman khan
2012-2-50-005
Department of ECE
East West University

Md. Ashrafal Alam Siddique
2012-3-50-003
Department of ECE
East West University

Letter of Acceptance

We hereby declare that this project is from the student's own work and best effort of mine, and all other source of information used have been acknowledge. This project has been submitted with our approval.

Dr.Md.Arifuzzaman

Supervisor

Assistant Professor

Department of ECE

East West University

Dr.Md.Mofazzol Hossain

Chairperson

Professor and Chairperson

Department of ECE

East West University

Acknowledgement

First of all, I would like to thank almighty Allah for giving me the strength & proper knowledge to complete my thesis work.

I would like to express my deep sense of gratitude and sincere thanks to my honorable supervisor Dr. Md. Arifuzzaman Assistant Professor, Department of ECE, East West University, Aftabnagar, Dhaka, Bangladesh for his at most direction, encouragement, kind guidance, sharing knowledge and constant inspiration throughout this project work.

My sincere gratefulness for the faculty of ECE whose friendly attitude and enthusiastic support that has given me for four years.

I am very grateful for the motivation and stimulation from my friends and seniors.

Finally my most heartfelt gratitude goes to my beloved parents and sisters for their endless support, continuous inspiration, great contribution and perfect guidance from the beginning to the end.

Table of Contents

Abstract	II
Declaration	III
Letter of Acceptance	IV
Acknowledgement	V
List of Figures	IX-X
List of Tables	XI
Chapter 1:	
Introduction	
1.1 Objective of the Research	08
1.2 Methodology of the Research	09
Chapter 2:	
Software Requirement Specification:	
2.1 Purpose	10
2.2 Scope	10
2.3 Product Perspective	11
2.4 Hardware Interfaces (Minimum)	11
2.5 Software Interfaces	11-12
Operations	
Product Functions	
User Characteristics	
2.6 Description	13
Chapter 3:	
Design Specification:	
Data base info	14
What type of data is included	15
Global Database connection	16
3.1: Login Form Testing	17
3.2: Patient Information Form Testing	17
3.3: Patient Check Out Form Testing	17

4 . Screenshot and Source Code	18-33
5. Conclusion and Future Work	34-35
6. Reference	36

Chapter 1

Introduction

Our life begins with the blessing of Almighty but hospitals are the second place where we first breath, so this is really essential part of our life, contributing best medical facilities to people suffering from various diseases, which may be due to change in climatic conditions, increased work-load, emotional trauma stress etc. It is obligatory for the hospitals to keep way of its day-to-day activities & records of its patients, doctors, nurses, ward boys and other staff personals that keep the hospital running successfully.

But keeping track of all the activities and their records on paper is very slow process. It is also very inefficient and a time-consuming process when observing the continuous increase in population and number of people visiting the hospital. Recording and maintaining all these records is highly unpredictable and systematic. It is not also economically & technically sensible to maintain these records on paper.

Thus keeping the working of the manual system is the point of our project we have to develop an automated version of the manual system, named as “Hospital Management System”.

The main point of our project is to provide a paper-less hospital system up to 95%. It also aims at providing less-cost reliable process of the existing systems. The system contributes excellent security of data at every level of user-system interaction and also contributes reliable memory storage and backup facilities.

1.1 Objective of the project

Our project “Hospital management system” is focused to develop to maintain the day-to-day state of admission/discharge of patients, list of doctors, reports generation, and etc. It is planned to achieve the following objectives:

- ✓ To computerize all data when there is include patient details & hospital details.
- ✓ Scheduling the appointment of patient with doctors to make it suitable for both.
- ✓ Scheduling the services of specialized doctors and emergency properly so that facilities supplied by hospital are fully utilized in effective manner.
- ✓ If the medical store issues medicines to patients, it should reduce the stock status of the medical store.
- ✓ It should be able to maintain the test reports of patients conducted in the pathology lab of the hospital.
- ✓ The record should be updated automatically whenever a transaction is made.

- ✓ The data of the patients should be kept up to date and their record should be kept in the system for historical purposes.

1.2 Methodology of the project

Primary data collection

Primary data is a term for data collected from a source. Raw data has not been subjected to processing or any other manipulation, and also referred to as primary data.

Primary data is a type of documentation that is obtained directly from first-hand sources by means of surveys, observation or experimentation.

Primary data collections are recorded directly from respondents. All the questions that one asks the respondents must be totally formulated so that all the different respondents understand it.

Secondary data collection

Secondary data is data collected by someone other than the purchaser. Main sources of secondary data for social science include censuses, organizational records and data collected through qualitative research.

<http://www.apollohospitals.com/>

Apollo is a private healthcare provider in Asia with hospitals in India, Sri-Lanka, Bangladesh and many other countries. It provides world class services for knee, hip replacement, heart, Transplants, Spine, Cancer Care, and Critical Care etc.

IBOM Multi-Specialty Hospital is a multi-specialty hospital owned

This hospital provides patient services including heart, Cardiology, Endoscopy, Radiology & Imaging, and Physiotherapy i.e. these project paper, all the information has been gathered from secondary sources that is internet.

Chapter 2

Software Requirement Specification

A Software requirements specification (SRS), a representation of a specification for a software system, is a complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software.

The following subsections of Software Requirement Specifications Document should facilitate in providing the entire overview of the Information system “Hospital Management System” under development. This document aims at defining the overall software requirements for admin. Efforts have been made to define the requirements of the Information system exhaustively and accurately.

2.1 Purpose

The main purpose of Software Requirement Specifications Document is to describe in a precise manner all the capabilities that will be provided by the Software Application “Hospital Management System”. It also states the various constraints which the system will be abide to. This document further leads to clear vision of the software requirements, specifications and capabilities. These are to be exposed to the development, testing team and end users of the software

2.2 Scope

The proposed of software product is the Hospital Management System (HMS). The system will be used in any Hospital, Clinic, Dispensary or Pathology labs in any Hospital, Clinic, Dispensary or Pathology labs to get the information from the patients and then storing that data for future. In This current system in use is a paper-based system. It is too slow and cannot provide updated lists of patients within a reasonable time. The main intention of the system was to reduce over-time pay and increase the number of patients that could be treated accurately. Requirements statements in this document are both functional and non-functional.

2.3 Product Perspective

The application will be windows-based, self contained and independent software product.

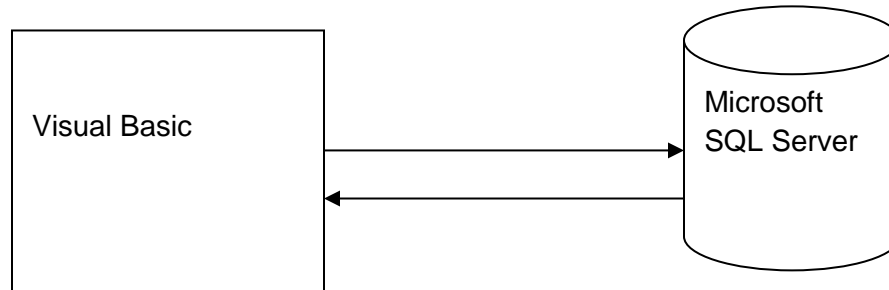


Figure- 2.1: Hierarchical clustering structure.

Interfaces

The application will have a user friendly and menu based interface. Following screens will be provided.

A Login Screen for entering username, password and role (Administrator, operator) will be provided. Access to different screens will be based upon the role of the user.

A Form for Search the details of a patient.

The Form for creating a new patient record will contain text fields where the Patient ID will be machine generated and the rest of the details will have to be filled up.

A Form for generating the tests reports.

The Form to produce a bill will create fields such as Patient ID, Appointment No., Doctor's charges, Hospital charges etc. which will need to be filled up.

The following reports will be generated: Tests reports

2.4 Hardware Interfaces (Minimum)

Processor: Pentium IV AND motherboard

RAM: 512MB or above

Hard Disk: 40GB or above

Input Devices: Keyboard, Mouse

Output Devices: Monitor; -14" VGA

2.5 Software Interfaces

OPERATING SYSTEM: Windows XP, Windows 7, Windows 8.1, Windows 10.

FRONT END: Microsoft Visual Basic Pro

BACK END: Microsoft SQL Server

Operations

This product will not cover any automated housekeeping aspects of database. The DBA at client site will be manually deleting old/ non required data. Database backup and recovery will also have to be handled by DBA.

Product Functions

The system will allow access only to authorized users with specific roles (Administrator, Operator). Depending upon the user's role, he/she will be able to access only specific modules of the system.

A summary of the major functions that the software will perform:

A login facility for enabling only authorized access to the system.

When a patient is admitted, the front-desk staff checks to see if the patient is already registered with the hospital. If he is, his/her Name is entered into the computer. Otherwise a new Patient ID is given to this patient.

If a patient checks out, the administrative staff shall delete his patient ID from the system.

The system generates reports on the following information: List of detailed information regarding the patient who has admitted in the hospital.

User Characteristics

1. Educational Level: At least graduate and should be comfortable with English language.
2. Technical Expertise: Should be a high or middle level employee of the organization comfortable with using general purpose applications on a computer.

System Features

Login module

Description

This module records only user and password of the user.

Patient Registration module

Description:

It keeps track of all details about patient. Patient id, patient name, address, admitted date, doctor name, and room no are entered in a form and stored for future reference. Also particular patient details can be viewed in the table using a separate form with an attribute patient id.

Patient Check out module

Description

Patient should be checked his Blood Pressure Sugar Diabetics and doctors will remark this condition.

Room Info module

Description

This module manages activities related to patient who Check out room fee, Room key, bed configuration etc.

Billing module

Description

This module bills the both inpatient and outpatient who comes to hospital. It also includes Payment details of patients. Depending on the payments bill report is generated.

Logical Database Requirements

The proposed information system contains the following data tables in its database collection.

Patient Registration table

Patient info Table

Patient Check Out Table

Room Detail's Table

Software System Attributes

Reliability

This application is a reliable product that produces fast and verified output of all processes.

Availability

This application will be available to use and help them to carry out their operations flexibly.

Security

The application will be password protected. User will have to enter correct username, password and role in order to access the application.

Maintainability

The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules.

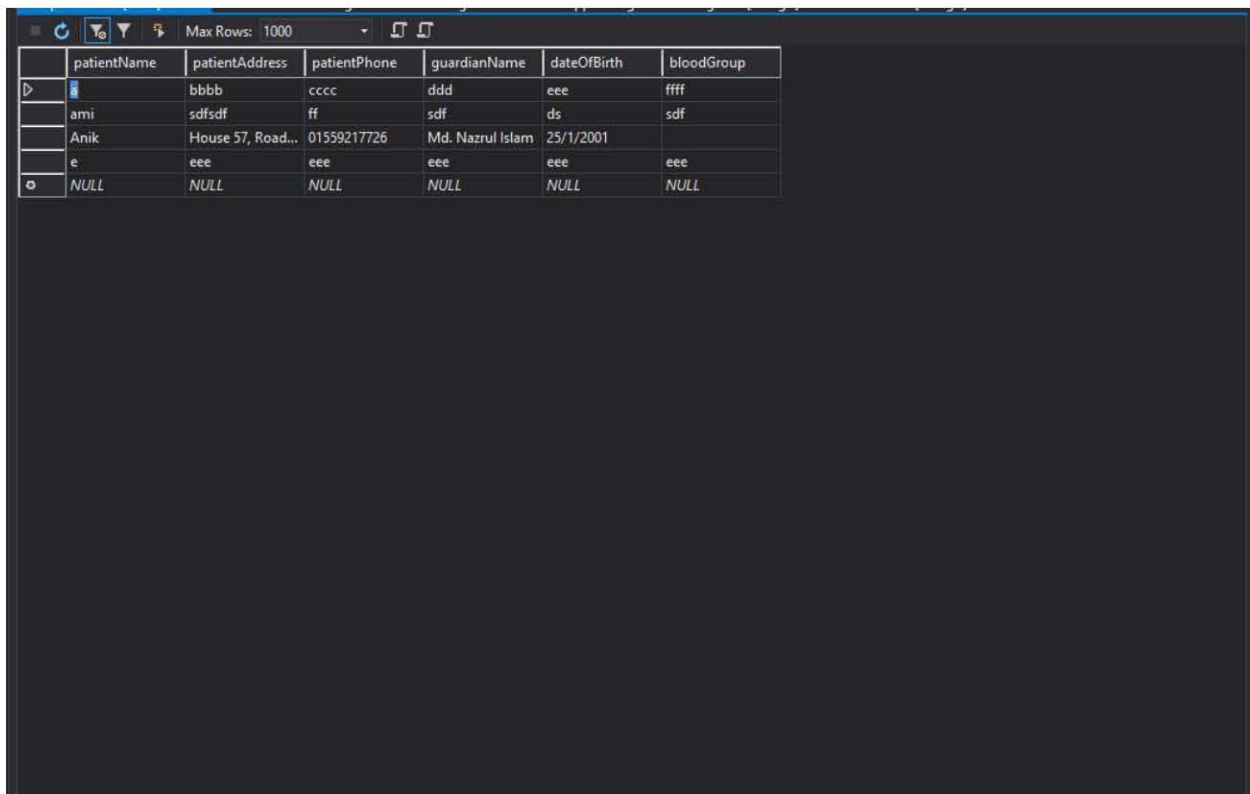
Portability

The application will be easily portable on any windows-based system that has oracle installed.

Chapter 3

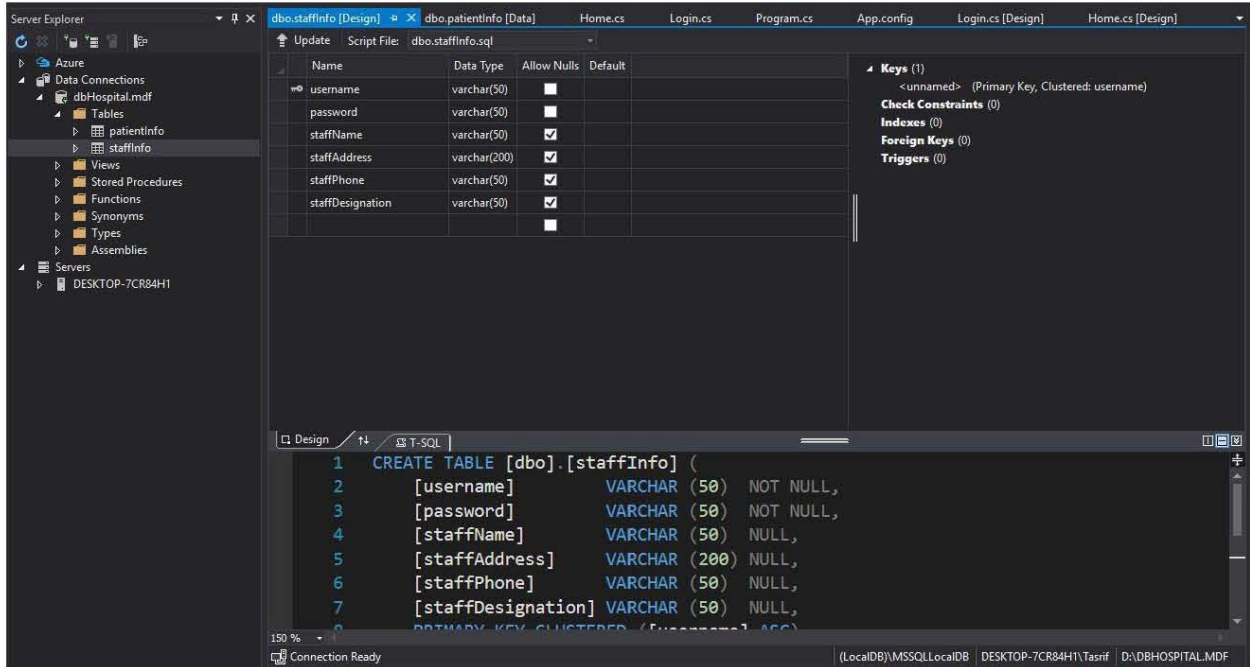
Design Specification (Screenshots)

Data base info:



	patientName	patientAddress	patientPhone	guardianName	dateOfBirth	bloodGroup
	bbb	bbbb	cccc	ddd	eee	fff
	ami	sdfsdf	ff	sdf	ds	sdf
	Anik	House 57, Road...	01559217726	Md. Nazrul Islam	25/1/2001	
	e	eee	eee	eee	eee	eee
	NULL	NULL	NULL	NULL	NULL	NULL

What type of data is included :



Name	Data Type	Allow Nulls	Default
username	varchar(50)	<input type="checkbox"/>	
password	varchar(50)	<input type="checkbox"/>	
staffName	varchar(50)	<input checked="" type="checkbox"/>	
staffAddress	varchar(200)	<input checked="" type="checkbox"/>	
staffPhone	varchar(50)	<input checked="" type="checkbox"/>	
staffDesignation	varchar(50)	<input checked="" type="checkbox"/>	

```
1 CREATE TABLE [dbo].[staffInfo] (  
2     [username]          VARCHAR (50) NOT NULL,  
3     [password]         VARCHAR (50) NOT NULL,  
4     [staffName]        VARCHAR (50) NULL,  
5     [staffAddress]     VARCHAR (200) NULL,  
6     [staffPhone]      VARCHAR (50) NULL,  
7     [staffDesignation] VARCHAR (50) NULL,  
8     PRIMARY KEY CLUSTERED ([username] ASC)
```

Global Database connection

```
staffinfo [Design]  dbo.patientinfo [Data]  Home.cs  Login.cs  Program.cs  App.config  Login.cs [Design]  Home.cs [Design]
1  <?xml version="1.0" encoding="utf-8" ?>
2  <configuration>
3  <configSections>
4  </configSections>
5  <connectionStrings>
6  <add name="Hospital_Database.Properties.Settings.dbHospitalConnectionString"
7  connectionString="Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|D:
8  providerName="System.Data.SqlClient" />
9  </connectionStrings>
10 <startup>
11 <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.2" />
12 </startup>
13 </configuration>
```


Table3.1: Login Form Testing

Test Case ID	Test	Expected Input	Expected Output	Actual Input	Actual Output
1.	Login Form	Username, Password	Homepage	admin, admin	Homepage

Table3.2: Patient Information Form Testing

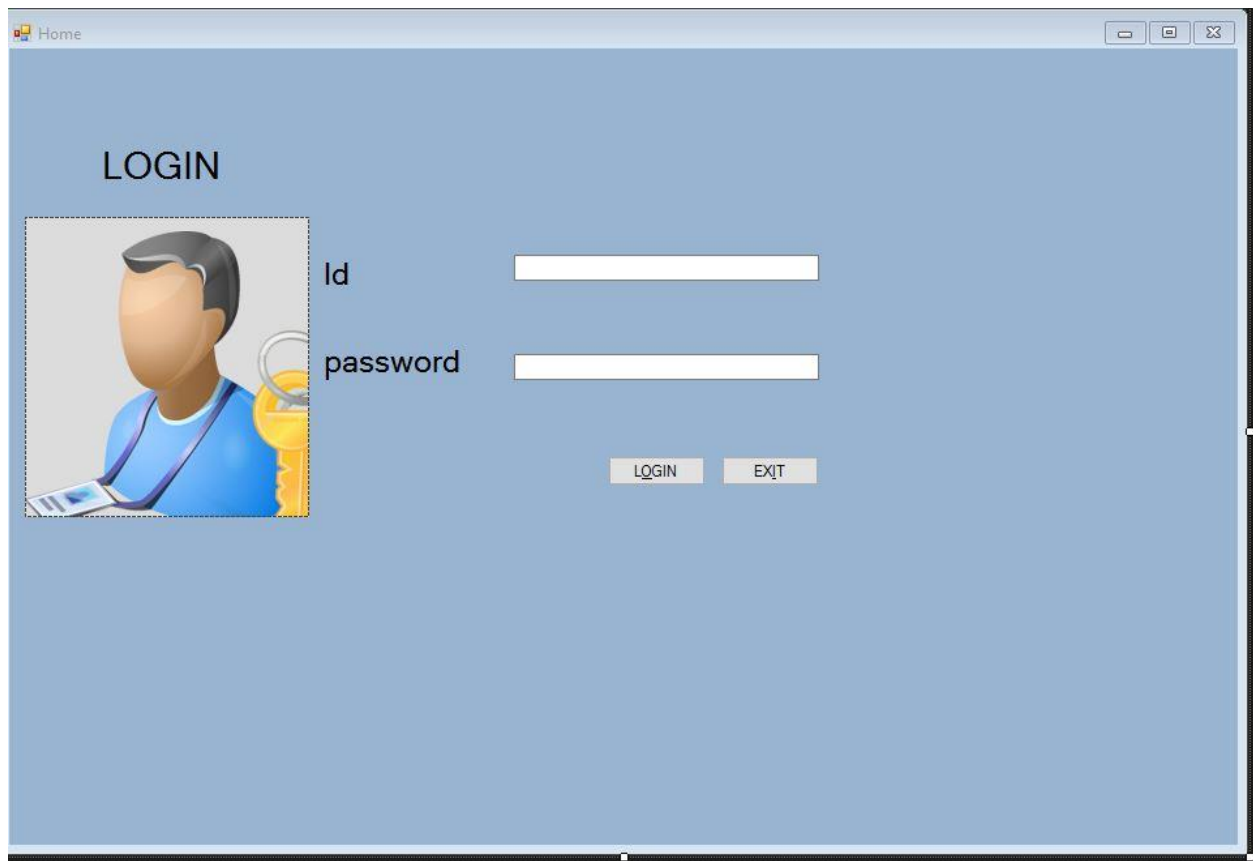
Test Case ID	Test	Expected Input	Expected Output	Actual Input	Actual Output
2.	Patient info Form	Current Condition	Msgbox appears(“data is saved”)	Patient info Form	Msgbox “Record Added Successfully!”

Table 3.3: Patient Check Out Form Testing

Test Case ID	Test	Expected Input	Expected Output	Actual Input	Actual Output
2.	Patient(Current) and Patient(Checked Out)	Room fee, Hospital fee, Room key(Cabin), Bed(Normal) and others	Msgbox appears(“data is saved”)	Patient(Current) and Patient(Checked Out)	Msgbox “Record Added Successfully!”

4 . Screenshot and Source Code

Login:



Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
```

```
namespace MedicalSoftwareStart
{
    public partial class Home : Form
    {
```

```

public Home()
{
    InitializeComponent();
}
public void ConnectDB()
{

}
private void pictureBox1_Click(object sender, EventArgs e)
{

}
private void button1_Click(object sender, EventArgs e)
{
    //SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Tasrif\documents\visual
studio 2015\Projects\MedicalSoftwareStart\MedicalSoftwareStart\Database1.mdf;Integrated
Security=True");
    //SqlDataAdapter da = new SqlDataAdapter("Select Count (*) From Login Where Id = ""
+ Box1.Text + "" and password = "" + Box2.Text + """, con) ;
    //DataTable dt = new DataTable();
    //da.Fill(dt);
    //if (dt.Rows[0][0].ToString() == "1")
    //{
    //    this.Hide();
    //    MedicalSoftware msd = new MedicalSoftware();
    //    msd.Show();
    //}
    //else
    //{
    //    MessageBox.Show("Please Check your Id and password");
    //}
    SqlConnection con = ConnectDB.GetSqlConnection();
    con.Open();
    if (Box1.Text == "" || Box2.Text == "")
    {
        MessageBox.Show("Please Enter Username & Password And Try Again.", "Error",
        MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }
}
try

```

```
{
    var c = new Model.LoginTB()
    {
        UserName = Box1.Text,
        UserPassword = Box2.Text
    };
SqlCommand command = new SqlCommand();
    command.Connection = con;
    command.CommandText = "select * from LoginTB where UserName = '" +
c.UserName + "' And UserPassword = '" + c.UserPassword + "'";
    SqlDataReader reader = command.ExecuteReader();
    if (reader.Read())
    {

        Form1 f = new Form1();

            }
    else
    {
        MessageBox.Show("Username & Password Don't Match. Please Try Again.",
"Error", MessageBoxButtons.OK, MessageBoxIcon.Hand);
        con.Close();
    }
}
catch (Exception ex)
{
    MessageBox.Show("error" + ex);
}
}

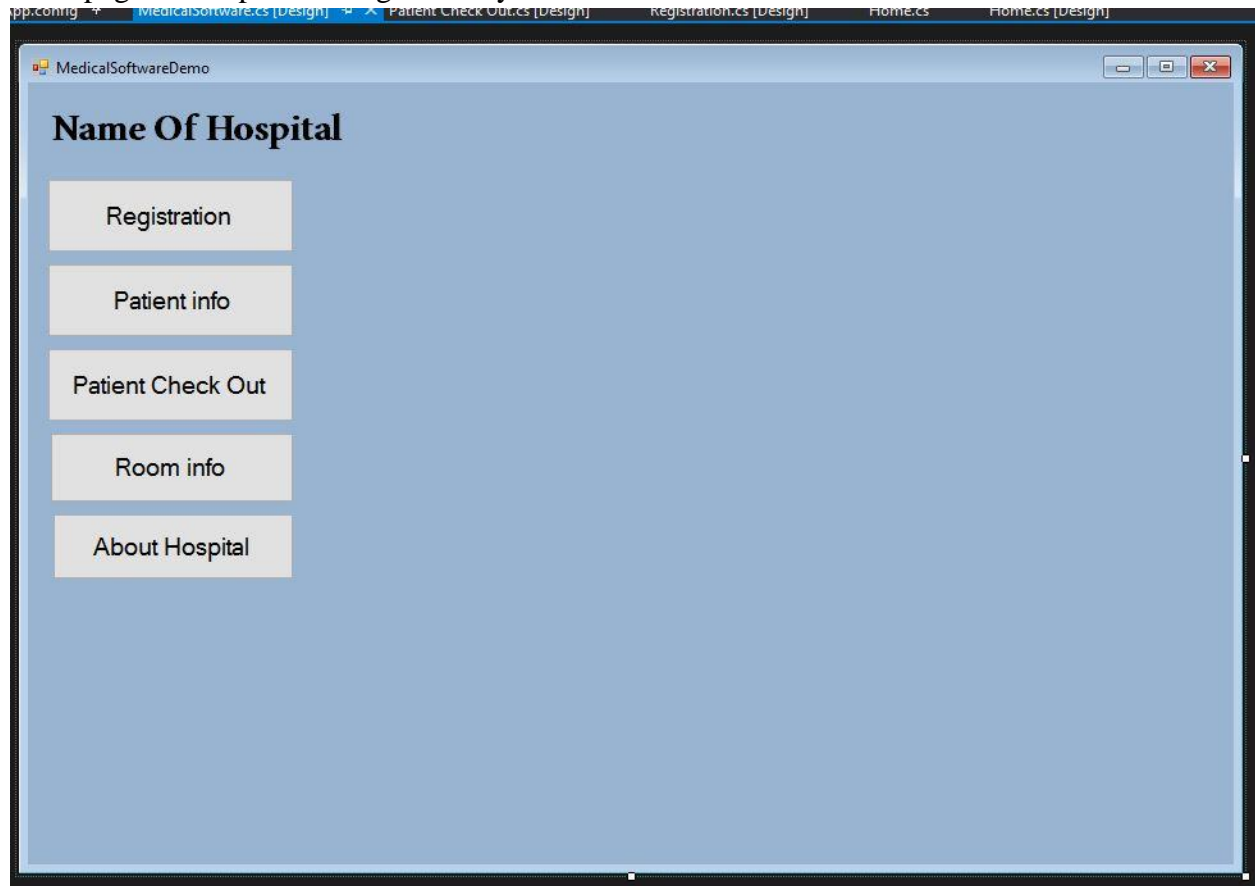
private void Home_Load(object sender, EventArgs e)
{
}

private void textBox1_TextChanged(object sender, EventArgs e)
{
}
}
```

```
private void button2_Click(object sender, EventArgs e)
{
    this.Close();
}

private void label3_Click(object sender, EventArgs e)
{
}
}
}
```

Homepage of Hospital Management System



Code :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalSoftwareStart
{
    public partial class MedicalSoftware : Form
    {
        public MedicalSoftware()
        {
            InitializeComponent();
        }

        private void label5_Click(object sender, EventArgs e)
        {
        }

        private void MedicalSoftware_Load(object sender, EventArgs e)
        {
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Registration ss = new Registration();
            ss.Show();
            this.Hide();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
        }

        {
            this.Hide();
            Patient_info pi = new Patient_info();
            pi.Show();
        }

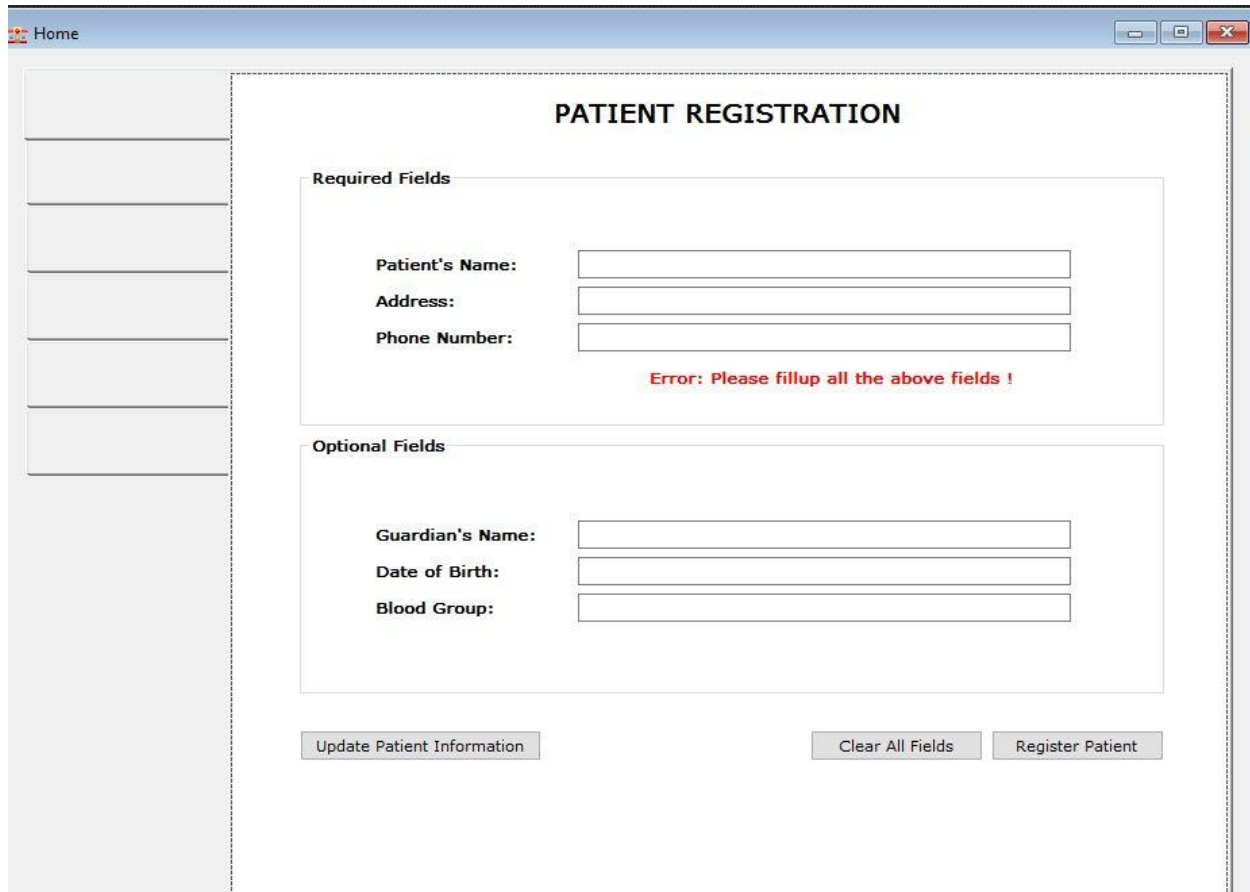
        }

        private void button3_Click(object sender, EventArgs e)
        {
            this.Hide();
            Patient_Check_Out pco = new Patient_Check_Out();
            pco.Show();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            this.Hide();
            Room_info ri = new Room_info();
            ri.Show();
        }
    }
}
```

```
private void button5_Click(object sender, EventArgs e)
{
    this.Hide();
    About_Hospital ah = new About_Hospital();
    ah.Show();
}
}
```

Patient Registration Form:



Home

PATIENT REGISTRATION

Required Fields

Patient's Name:

Address:

Phone Number:

Error: Please fillup all the above fields !

Optional Fields

Guardian's Name:

Date of Birth:

Blood Group:

Update Patient Information Clear All Fields Register Patient

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hospital_Database
{
    public partial class Home : Form
    {
        SqlConnection con = new SqlConnection(@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\SUNNY\Documents\dbHosp
ital.mdf;Integrated Security=True;Connect Timeout=30");
        bool adminEntrance;
        public Home(bool admin)
        {
            InitializeComponent();
            adminEntrance = admin;
        }

        private void Home_Load(object sender, EventArgs e)
        {
            if(adminEntrance == false) tabPageAdmin.Remove(tabPageAdmin);
        }

        private void Home_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (e.CloseReason == CloseReason.UserClosing)
            {
                Application.Exit();
            }
        }

        private void tabPageAdmin_DrawItem(object sender, DrawItemEventArgs e)
        {
            Graphics g = e.Graphics;
            Brush _textBrush;

            // Get the item from the collection.
            tabPageAdmin.Text = tabPageAdmin.TabPages[e.Index];
        }
    }
}
```



```
// Get the real bounds for the tab rectangle.
Rectangle _tabBounds = tabControl1.GetTabRect(e.Index);

if (e.State == DrawItemState.Selected)
{
    // Draw a different background color, and don't paint a focus rectangle.
    _textBrush = new SolidBrush(Color.Black);
    g.FillRectangle(Brushes.AliceBlue, e.Bounds);
}
else
{
    _textBrush = new System.Drawing.SolidBrush(e.ForeColor);
    e.DrawBackground();
}

// Use our own font.
Font _tabFont = new Font("MS Reference Sans Serif", (float)10.0, FontStyle.Bold,
GraphicsUnit.Pixel);

// Draw string. Center the text.
StringFormat _stringFlags = new StringFormat();
_stringFlags.Alignment = StringAlignment.Center;
_stringFlags.LineAlignment = StringAlignment.Center;
g.DrawString(_tabPage.Text, _tabFont, _textBrush, _tabBounds, new
StringFormat(_stringFlags));
}

private void btnRegisterPatient_Click(object sender, EventArgs e)
{
    if (tbPatientName.Text.Equals("") || tbPatientAddress.Text.Equals("") ||
tbPatientPhone.Text.Equals("")) labelError.Visible = true;
    else
    {
        labelError.Visible = false;

        try
        {
            con.Open();
            SqlCommand cmd = con.CreateCommand();
            cmd.CommandType = CommandType.Text;
```

```

cmd.CommandText = "insert into patientInfo values('" + tbPatientName.Text + "','"
+ tbPatientAddress.Text + "','" + tbPatientPhone.Text + "','" + tbGuardianName.Text + "','" +
tbPatientDateOfBirth.Text + "','" + tbPatientBloodGroup.Text + "')";
cmd.ExecuteNonQuery();
con.Close();

MessageBox.Show("Patient registered successfully.");
tbPatientName.Text = "";
tbPatientAddress.Text = "";
tbPatientPhone.Text = "";
tbGuardianName.Text = "";
tbPatientDateOfBirth.Text = "";
tbPatientBloodGroup.Text = "";
btnUpdatePatientBasicInformaton.Visible = false;
}
catch(Exception err)
{
    MessageBox.Show(err.Message);
}
}
}

//Search patient information
private void btnSearchPatient_Click(object sender, EventArgs e)
{
    try
    {
        SqlDataAdapter sda = new SqlDataAdapter("Select * From patientInfo where
patientName = '"+tbSearchPatient.Text+'"', con);
        DataTable dt = new DataTable();
        sda.Fill(dt);

        try
        {
            labelPatientName.Text = dt.Rows[0][0].ToString();
            labelPatientAddress.Text = dt.Rows[0][1].ToString();
            labelPatientPhone.Text = dt.Rows[0][2].ToString();
            labelGuardianName.Text = dt.Rows[0][3].ToString();
            labelPatientDateOfBirth.Text = dt.Rows[0][4].ToString();
        }
    }
}

```

```
        labelPatientBloodGroup.Text = dt.Rows[0][5].ToString();
        con.Close();
    }
    catch(Exception)
    {
        MessageBox.Show("No match found !");
    }
}
catch(Exception err)
{
    MessageBox.Show(err.Message);
}
}

//Edit basic information of patient
private void btnEditPatientBasicInfo_Click(object sender, EventArgs e)
{
    if (labelPatientName.Text != "...")
    {
        tbPatientName.Text = labelPatientName.Text;
        tbPatientAddress.Text = labelPatientAddress.Text;
        tbPatientPhone.Text = labelPatientPhone.Text;
        tbGuardianName.Text = labelGuardianName.Text;
        tbPatientDateOfBirth.Text = labelPatientDateOfBirth.Text;
        tbPatientBloodGroup.Text = labelPatientBloodGroup.Text;

        btnUpdatePatientBasicInformaton.Visible = true;
        tabControl1.SelectTab(0);
        tbPatientName.Enabled = false;
    }
}

//Edit medical information of patient
private void btnEditPatientMedicalInfo_Click(object sender, EventArgs e)
{
}

private void btUpdatePatientInformaiton_Click(object sender, EventArgs e)
{
}
```

```
if (tbPatientName.Text.Equals("") || tbPatientAddress.Text.Equals("") ||
tbPatientPhone.Text.Equals("")) labelError.Visible = true;
else
{
    labelError.Visible = false;

    try
    {
        con.Open();
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "update patientInfo set patientAddress=" +
tbPatientAddress.Text + ",patientPhone=" + tbPatientPhone.Text + ",guardianName=" +
tbGuardianName.Text + ",dateOfBirth=" + tbPatientDateOfBirth.Text + ",bloodGroup=" +
tbPatientBloodGroup.Text + " where patientName="+ tbPatientName.Text + """;
        cmd.ExecuteNonQuery();
        con.Close();

        MessageBox.Show("Patient information updated successfully.");
        tbPatientName.Text = "";
        tbPatientAddress.Text = "";
        tbPatientPhone.Text = "";
        tbGuardianName.Text = "";
        tbPatientDateOfBirth.Text = "";
        tbPatientBloodGroup.Text = "";

        btnUpdatePatientBasicInformaton.Visible = false;
        tabControl1.SelectTab(1);
        btnSearchPatient.PerformClick();

    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message);
    }
}

btnUpdatePatientBasicInformaton.Visible = false;
```

```
}
```

```
private void btnClearRegistrationFields_Click(object sender, EventArgs e)
```

```
{
```

```
    tbPatientName.Text = "";  
    tbPatientAddress.Text = "";  
    tbPatientPhone.Text = "";  
    tbGuardianName.Text = "";  
    tbPatientDateOfBirth.Text = "";  
    tbPatientBloodGroup.Text = "";  
    btnUpdatePatientBasicInformaton.Visible = false;
```

```
}
```

```
private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
}
```

```
DataTable dt;
```

```
private void btnSearchToRemoveRecord_Click(object sender, EventArgs e)
```

```
{
```

```
    con.Open();  
    SqlCommand cmd = con.CreateCommand();  
    cmd.CommandType = CommandType.Text;
```

```
    if(comboBoxRemoveRecord.Text.Equals("Patient")) cmd.CommandText = "select *  
from patientInfo where patientName = '"+tbNameToRemove.Text+"'";  
    else cmd.CommandText = "select * from staffInfo where staffName = " +  
tbNameToRemove.Text + "'";
```

```
    cmd.ExecuteNonQuery();
```

```
    dt = new DataTable();  
    SqlDataAdapter da = new SqlDataAdapter(cmd);  
    da.Fill(dt);  
    dataGridView1.DataSource = dt;
```

```
    con.Close();
```

```
}
```

```
private void btnRemoveRecord_Click(object sender, EventArgs e)
```

```
{
    con.Open();
    SqlCommand cmd = con.CreateCommand();
    cmd.CommandType = CommandType.Text;

    if (comboBoxRemoveRecord.Text.Equals("Patient")) cmd.CommandText = "delete from
patientInfo where patientName='"+tbNameToRemove.Text+"'";
    else cmd.CommandText = "delete from staffInfo where staffName =
 '"+tbNameToRemove.Text+"'";

    cmd.ExecuteNonQuery();
    con.Close();

    dt.Clear();
    MessageBox.Show("Record deleted successfully.");
}

private void btnRegisterStaff_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = con.CreateCommand();
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "insert into staffInfo values('" + tbStaffUsername.Text + "','" +
tbStaffPassword.Text + "','" + tbStaffName.Text + "','" + tbStaffAddress.Text + "','" +
tbStaffPhone.Text + "','" + tbStaffDesignation.Text + "')";
        cmd.ExecuteNonQuery();
        con.Close();

        MessageBox.Show("Staff registered successfully.");
        tbStaffName.Text = "";
        tbStaffAddress.Text = "";
        tbStaffPhone.Text = "";
        tbStaffDesignation.Text = "";
        tbStaffUsername.Text = "";
        tbStaffPassword.Text = "";
    }
    catch (Exception err)
    {

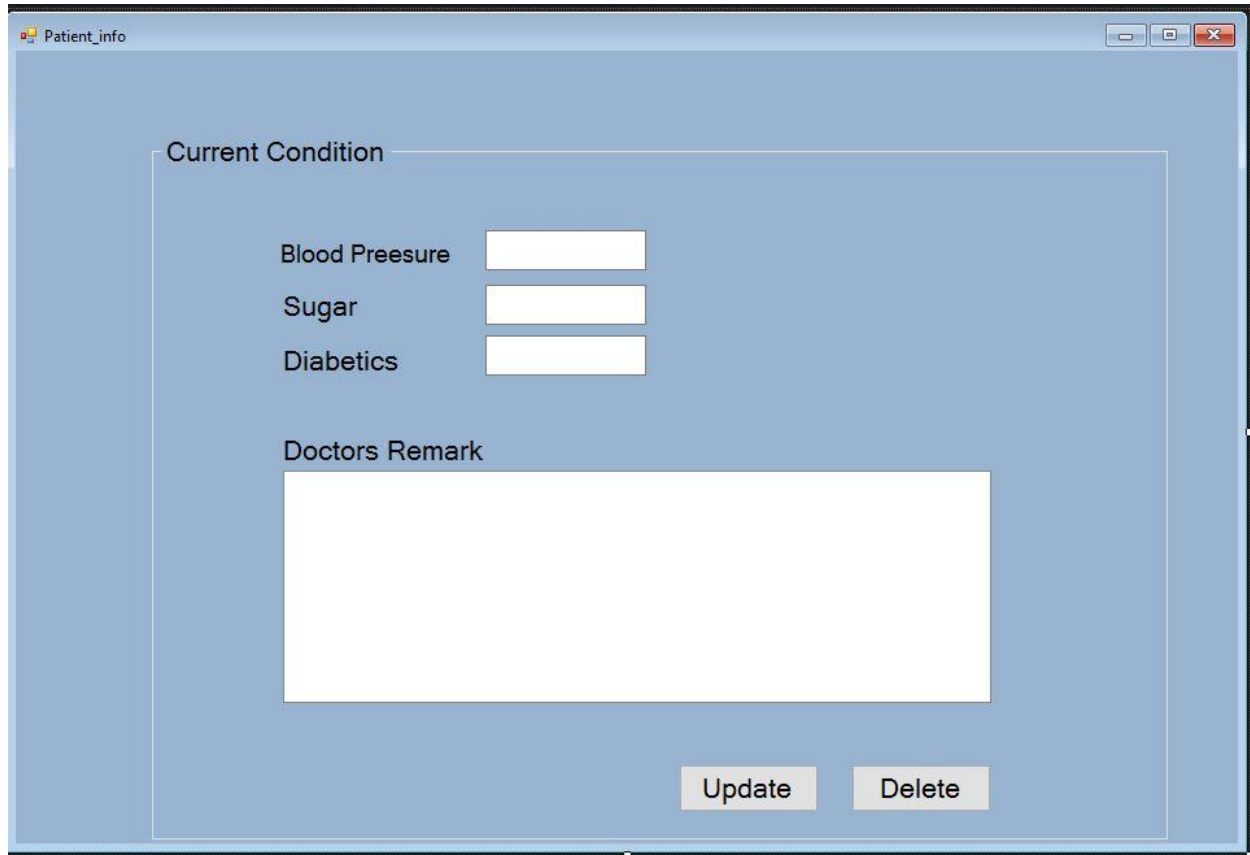
```

```

        MessageBox.Show(err.Message);
    }
}
}
}

```

Patient InfoForm :



Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalSoftwareStart
{
    publicpartialclassPatient_info : Form
    {

```

```

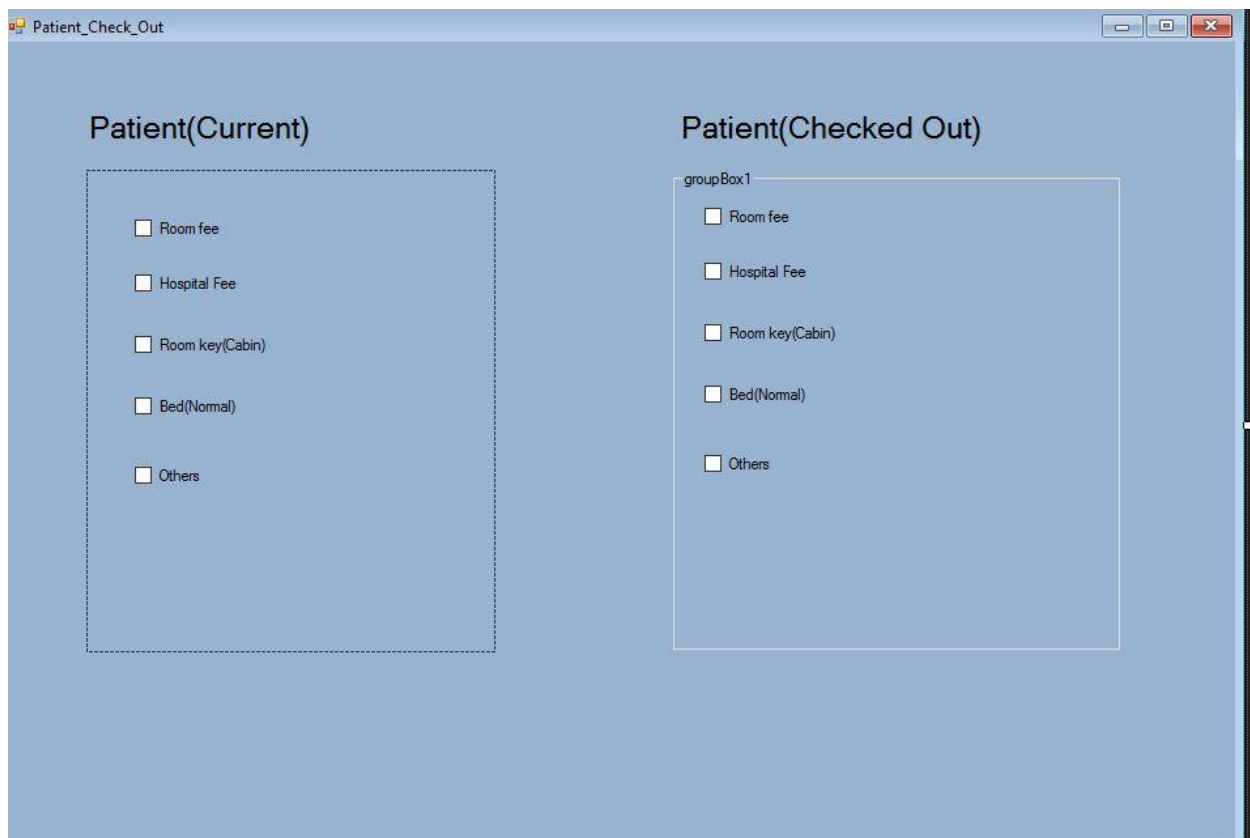
public Patient_info()
{
    InitializeComponent();
}

private void Patient_info_Load(object sender, EventArgs e)
{
}

private void button2_Click(object sender, EventArgs e)
{
}
}
}

```

Patient Check Out Form:



Code:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;

```



```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace MedicalSoftwareStart
{
    public partial class Patient_Check_Out : Form
    {
        public Patient_Check_Out()
        {
            InitializeComponent();
        }

        private void label1_Click(object sender, EventArgs e)
        {
        }

        private void Patient_Check_Out_Load(object sender, EventArgs e)
        {
        }
    }
}
```

C sharp Functions:

Clone ()	Make clone of string.
CompareTo ()	Compare two strings and returns integer value as output. It returns 0 for true and 1 for false.
Contains ()	The C# Contains method checks whether specified character or string is exists or not in the string value.
EndsWith ()	This Ends With Method checks whether specified character is the last character of string or not.
Equals ()	The Equals Method in C# compares two string and returns Boolean value as output.
GetHashCode ()	This method returns Hash Value of specified string.
GetType ()	It returns the System. Type of current instance.
GetTypeCode ()	It returns the System .Type Code for class System. String.
IndexOf ()	Returns the index position of first occurrence of specified character.
ToLower ()	Converts String into lower case based on rules of the current culture.
ToUpper ()	Converts String into Upper case based on rules of the current culture.
Insert ()	Insert the string or character in the string at the specified position.
IsNormalized ()	This method checks whether this string is in Unicode normalization form C.

LastIndexOf()	Returns the index position of last occurrence of specified character.
Length	It is a string property that returns length of string.
Remove()	This method deletes all the characters from beginning to specified index position.
Replace()	This method replaces the character.
Split()	This method splits the string based on specified value.
StartsWith()	It checks whether the first character of string is same as specified character.
Substring()	This method returns substring.
ToCharArray()	Converts string into char array.
Trim()	It removes extra whitespaces from beginning and ending of string.

Database:

SqlConnection(String, SqlConnectionCredential)	Initializes a new instance of the SqlConnection class given a connection string, that does not use Integrated Security = true and a SqlConnectionCredential object that contains the user ID and password.
SqlConnection(String)	Initializes a new instance of the SqlConnection class when given a string that contains the connection string.
SqlConnection(String, SqlConnectionCredential)	Initializes a new instance of the SqlConnection class given a connection string, that does not use Integrated Security = true and a SqlConnectionCredential object that contains the user ID and password.

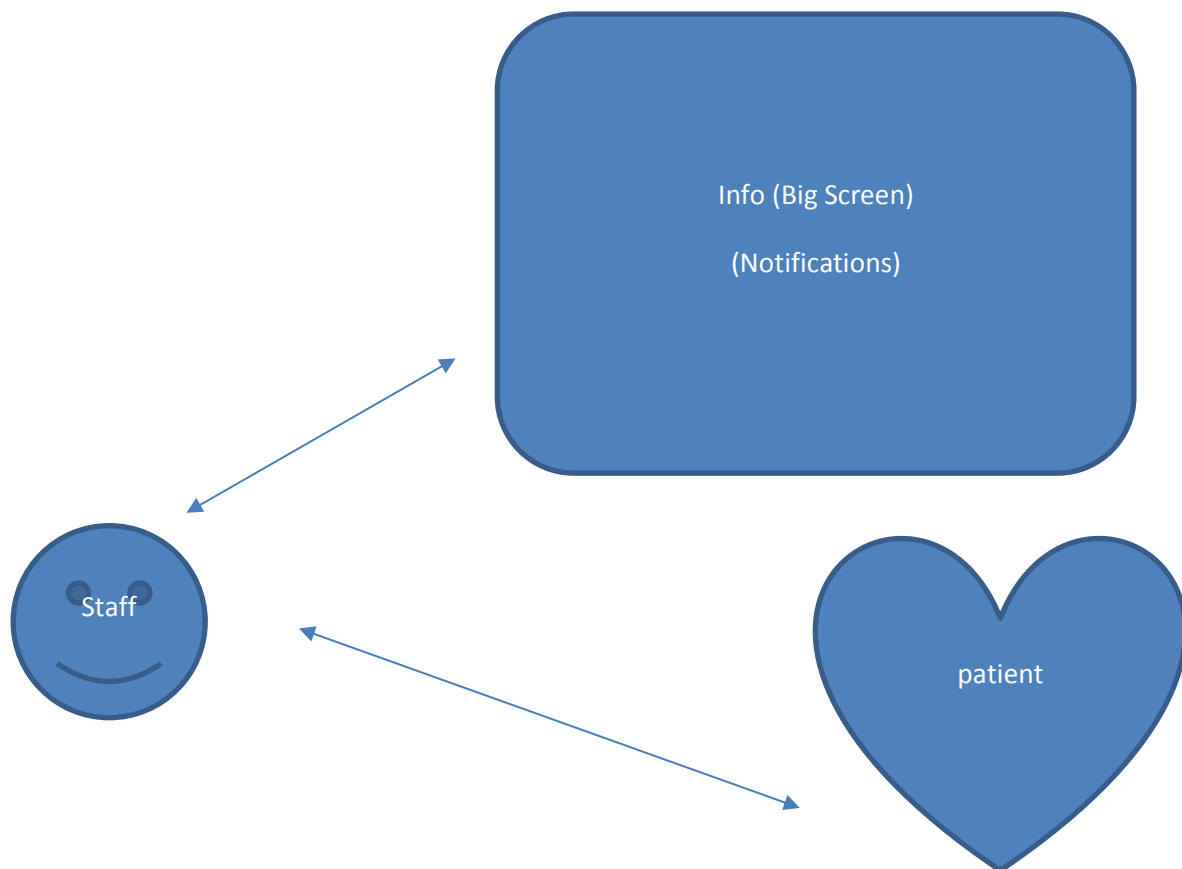
Chapter 5

Conclusion and Future Work

As this system saves both money and time, this software should be included in every Hospitals. As modern trade begins and future depends on technology fully so upgrade in every Hospital must be needed.

We also going to add a notification system in the software which will alarm the condition of patients whom are admitted in the I.C.U.

This Notification will be shown in a big screen for the sake of I.C.U patients. Their food chart, medicines and other important goods are going to be shown which will let the staffs know very faster through this process.



References

1. **C# 5.0 in a Nutshell: The Definitive Reference**

By: Joseph Albahari , Ben Albahari

2. **Head First C#**

By: Jennifer Greene, Andrew Stellman

.

3. **www.google.com**

4. <https://www.youtube.com/watch?v=tcmCmMs8yU&t=4s>

5. <http://csharp.net-informations.com/data-providers/csharp-sql-server-connection.htm>

6. <http://www.completecsharptutorial.com/csharp-articles/csharp-string-function>

7. [https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection(v=vs.110).aspx)