A PROJECT REPORT ON

**Diagnostic centre management system**

Submitted

By

**Mr. Dwarkesh Harish Thanki**

**(Enrollment No. 2021013686)**

In fulfillment for the award of the degree

Of

**Bachelor of Computer Application**

Guided by

**Mr. Thakrar Zalak**

Shri V J Modha College of IT and Management – Porbandar

Bhakta Kavi Narsinh Mehta University, Junagadh

**Academic Year**

**2021 - 2024**



# Acknowledgement

During the preparation of the project, we have a good fortune of receiving support, in various ways, from several personal, numerous to mention here. We owe a debt of gratitude to all of them.

It is our privilege to express our sincerest regards to our project coordinator, Prof. ZALAK THAKRAR for their valuable inputs, able guidance, Encouragement, wholehearted cooperation and constructive criticism throughout the duration of our project.

It is our great pleasure to represent our project as one web application titled “Diagnostic Centre Management System” and which we conceived in the 5th semester of BCA affiliated with BKNMU (Bhakt Kavi Narshinh Mehta University).

We are also thankful to the BKNMU (Bhakt Kavi Narshinh Mehta University) for including this project development subject in our syllabus. We got a golden opportunity to test and implement our creativity and programming skill simultaneously. Lastly, we would like to extend our sincere thanks to our advisors, classmates as well as all the books and websites who have directly or indirectly helped us.

# Preface

This Desktop Application Provide Efficient, Reliable way to Accounting Make Easy,Manage Doctor, Invoice Generate, Update and Manage

Patient and Test List.

* Manage Doctors’
* Patient
* Test
* Invoice
* Dashboard
* This Software Made With C# and SQL Server Database.
* This Software has only access to ADMIN.
* Customer Role is Receive Invoice and Payment.

# Index

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Topic Name** | **Page No** |
| Chapter – 1 | **INTRODUCTION**  1.1 Purpose  1.2 Scope  1.3 Technology & Literature Review | **1** |
| Chapter – 2 | **SYSTEM ANALYSIS**  2.1Problem Definition  2.2Process Model  2.3Requirement Analysis  2.4SRS  2.5Grant Chart | **6** |
| Chapter – 3 | **SYSTEM DESIGN**  3.1 Data Flow Diagram  3.2 E-R Diagram  3.3 Use Case Diagram | **18** |
| Chapter – 4 | **DATA DICTIONARY**  4.1 Data Dictionary | **23** |
| Chapter – 5 | **INPUT AND OUTPUT DESIGN**  5.1 Admin Layout | **30** |
| Chapter – 6 | **LIMITATIONS AND FUTURE ENHANCEMENT**  6.1 Limitations  6.2 Further Enhancement | **63** |
| Chapter – 7 | **CONCLUSION**  7.1 Conclusion  7.2 Advantages | **66** |
| Chapter – 8 | **BIBLIOGRAPHY** | **68** |
| Chapter – 9 | **REFERENCES** | **70** |

CHAPTER NO: 1 INTRODUCTION

**1.1 Purpose**

**1.2 Scope**

**1.3 Technology and Literature Review**

**1.1 Purpose**

**The purpose of this Software is as follows:**

* This Software Make Easy to Manage Doctors, Patient, Invoice, Test, Dashboard.
* It will Print Invoice and also Save Invoice to Database.

**It provides following facilities to Users:**

o Provide Service of Invoice.

o Provide Username and Password Service to Admin For Various Function of Software.

**1.2 Scope**

* The scope of this Software is to provide an easy option for the who is willing to Digital Management of diagnostic centre routine Task.
* It saves their time.
* This Software can be accessed From Desktop or Laptop, thus providing client’s comfort.
* Considering the benefits of the client, the software has also an additional feature.

**The goals of the system are:**

* + To provide Analysis of Doctors, Patient and Income.
  + To handle more customers, Tests in less time with fewer resources.

**1.3 Technology and Literature Review**

**Front End:**

* **C# .NET** 
  + C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run in the .NET ecosystem.
  + The .NET ecosystem is composed of all the implementations of .NET, including both but not limited to .NET Core, and .NET Framework.
  + You can use C# to create Windows client applications, XML Web services, distributed components, client-server applications, database applications, and much, much more.

**Backend:**

* **SQL Server**
  + SQL Server is a type of database software that is used to store information for test, patient and invoice, doctors etc.
  + With SQL Server, you can analyze large amounts of data faster and more efficiently than with Excel or other types of spreadsheets.
  + SQL is most popular for its tables, forms and queries. The database tables are similar to spreadsheets, so you shouldn’t have much trouble using the basic functions of the program.

However, it does take time to learn the full features.

# CHAPTER NO: 2

**SYSTEM ANALYSIS**

**2.1 Problem Definition**

**2.2 Process Model**

**2.3 Requirement Analysis**

**2.4 SRS**

**2.5 Gantt Chart**

**2.1 Problem Definition**

This Software is designed to overcome those problems using manual system such as usually, the work in the shop is paper-based, it's time-consuming.

The Paper and documentation are occupying more storage space.

**2.2 Process Model**

**Iterative Waterfall Model**

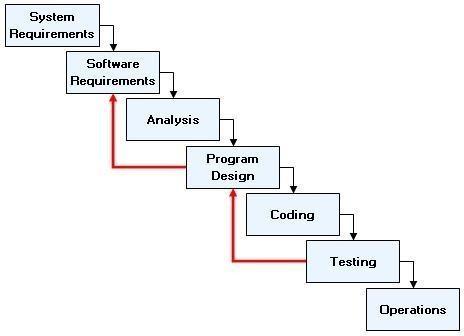


Figure 2.2

* **Advantages of Iterative Waterfall Model** 
  + Simple and Easy to Understand And Each Phase has well Defined Input And Output
  + It Work well for Smaller Project Where Requirement Are Clear

And very well understood

* + It Divide complex task into more manageable works.

* **Application of Iterative Waterfall Model** 
  + 1. This Model is used When Requirements are clear And Fix
    2. Product Definition is Stable & Technology is understood & it used when Project is short.
* **Why Iterative Waterfall Model??**

Online Photography registering website is a large system with all functionality and specification.

ITERATIVE WATERFALL Model is used for development process of online Photography registering website.

The incremental Model is an evolution of the waterfall model, where the waterfall model incrementally applied. The Incremental Process Model combines elements of the linear sequential model (applied repetitively) with the iterative philosophy of prototyping.

* **Implementation of iterative model in this project:**

At first we try to find the requirements of information about the billing system requirement of client.

**2.3 Requirement Analysis**

**Hardware Requirement**

* Operating System : 32 bit/64bit

* RAM : 2 GB

**Software Requirement**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  Front End Tool | : | C# .NET |
|  |  Back End Tool | : | SQL Server (.mdf) |
|  |  Development Tool | : | Visual Studio 2010 (ultimate) |

* Supported Operating Systems:

* + Windows 7 (32-bit/64-bit)
  + Windows 8 (32-bit/64-bit)
  + Windows 10 (32-bit/64-bit).

**2.4 SRS**

**REQUIREMENT SPECIFICATION OF ADMIN**

R1: LOGIN.

R2: DASHBOARD.

R3: DOCTORS’ MANAGES

R4: PATIENTS’ MANAGES.

R5: TEST MANAGES.

R6: INVOICE CONFIGURATION.

### R1: LOGIN

* Description: This functionality will be used for authenticate access of Admin.
* State: This is the beginning point, an admin screen with username and password will be displayed in Form.
* Input: Input to the system would be password & username. Output: Output will be the result of the authentication process.
* Process: User input will be match against the valid account details and according to its decision will be generated. I.e. authenticate user or not.

### R2: DOCTORS’ REGISTRATION

* Description: Using this functionality ADMIN will add, update or delete Doctor registrations.
* State: admin will be logged in, Doctor registrations with add, update or delete functionality Will be displayed in Form.
* Input: Input to the system would be added, update or delete Doctor registration.
* Output: Output will be the result of added, update or delete Doctor registration.
* Process: User input will be valid then added, update or delete Doctor registration.

### R3: MANAGE PATIENT

* Description: Using this functionality ADMIN will add a product.
* State: admin will be in Form windows displays option add, update and delete Patient.
* Input: Input to the would be added, update and delete Patient.
* Output: Output will be the result of add, update and delete Patient.
* Process: User input will be valid then add, update and delete Patient.

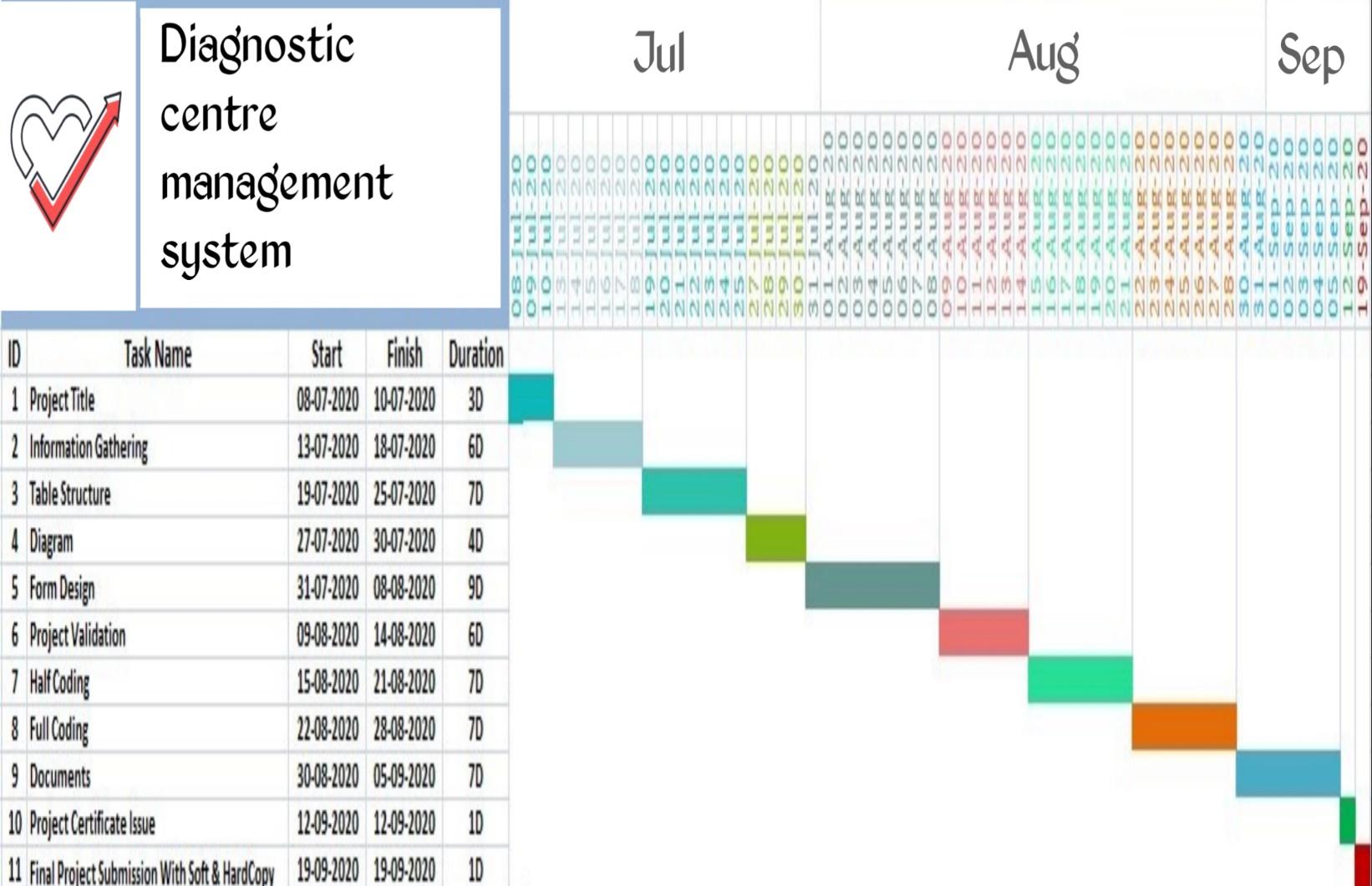
### R4: TEST CONFIGURATION

* Description: Using this functionality ADMIN will add a Test configuration and information.
* State: admin will be in Form windows displays option add, update and delete Test detail.
* Input: Input to the would be added, update and delete Test detail.
* Output: Output will be the result of add, update and delete Test detail.
* Process: User input will be valid then add, update and delete Test detail.

### R5: INVOICE

* Description: This functionality will be used for invoice generate, update and delete.
* State: Admin will be managing invoice displayed in Form.
* Input: Input to the system would be an invoice detail.
* Output: Output will be managed invoice information and print.
* Process: admin input will be valid and invoices add, update and remove.

**2.5 Gantt Chart**



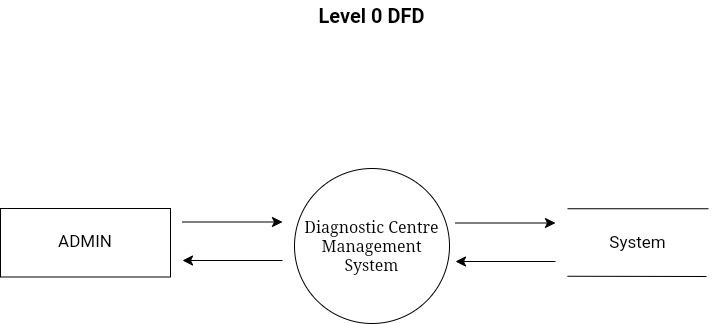
# CHAPTER NO: 3 SYSTEM DESIGN

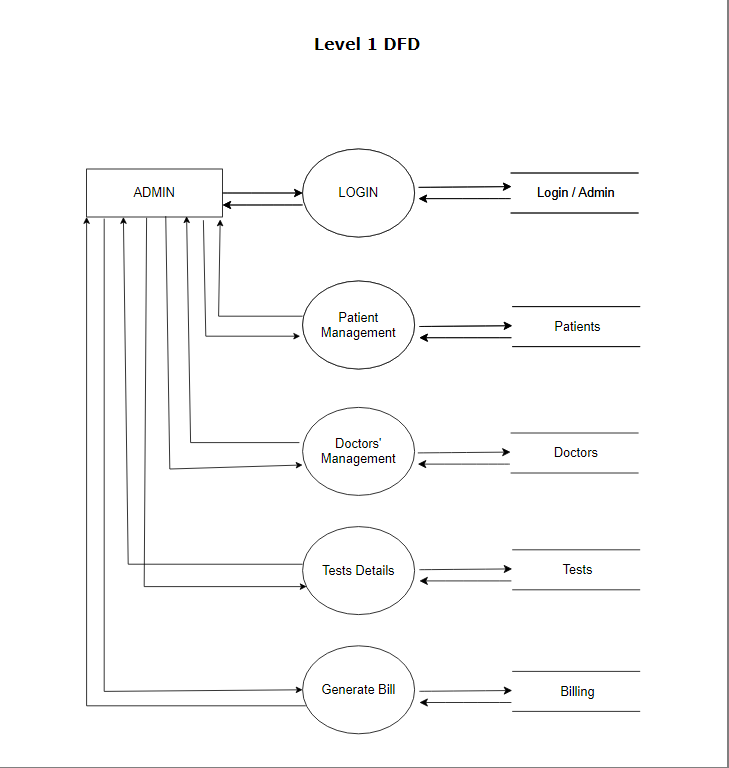
**3.1 Data Flow Diagram**

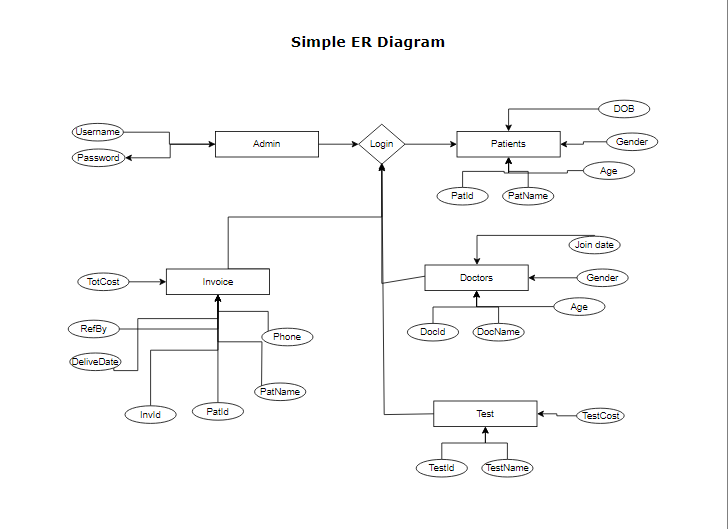
**3.2 E.R Diagram**

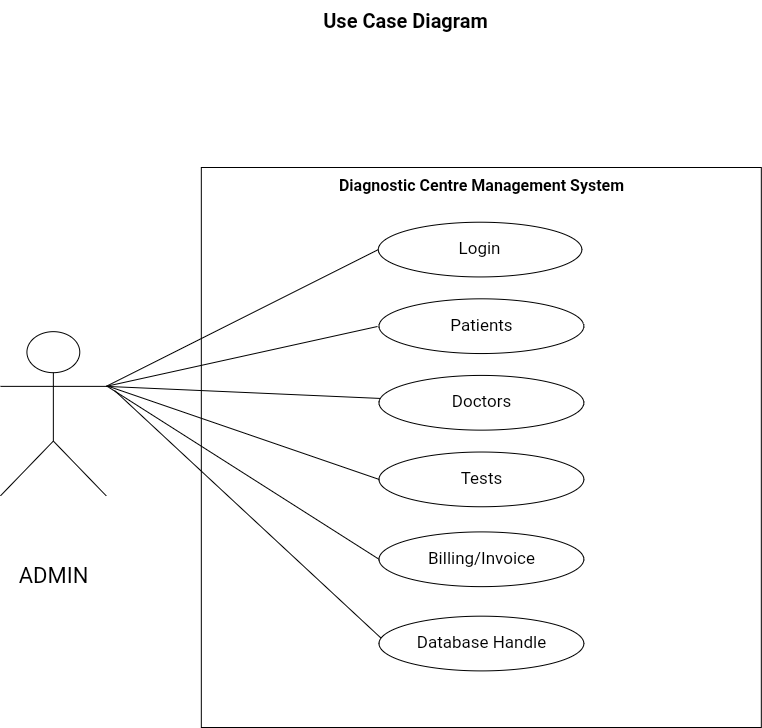
**3.3 Use Case Diagram**

**Diagrams**









# CHAPTER NO: 4

**DATA DICTIONARY**

**4.1 Data Dictionary**

**Database Table Structure**

Table 1 : TestTbl

|  |  |  |
| --- | --- | --- |
| Sr No | Name | Data Types |
| 1 | TestId | Int |
| 2 | TestDesc | Varchar(50) |
| 3 | TestCost | Int |

Description:-

* TestId

It is number format, use to identify the rows number.

* TestDesc

It is character format, use to present the description to patient.

* TestCost

It is number format, use to identify the cost number.

Table 2 : DoctorTbl

|  |  |  |
| --- | --- | --- |
| Sr No | Name | Data Types |
| 1 | DocId | Int |
| 2 | DocName | Varchar(50) |
| 3 | DocDOB | Date |
| 4 | DocPhone | Varchar(50) |
| 5 | DocAdd | Varchar(50) |
| 6 | Designation | Varchar(50) |
| 7 | JoinDate | Date |

Description:-

* DocId

It is number format, use to identify the rows number.

* DocName

It is character format, use to present the name to doctor.

* DocDOB

It is number format, use to identify the Date Of Birth.

* DocPhone

It is number format, use to identify the phone number.

* DocAdd

It is character format, use to add an doctor.

* Designation

It is character format, use to know doctors’ designation.

* JoinDate

It is date format, use to know doctors’ joining date.

Table 3 : PatientTbl

|  |  |  |
| --- | --- | --- |
| Sr No | Name | Data Types |
| 1 | PatId | Int |
| 2 | PatName | Varchar(50) |
| 3 | Age | Int |
| 4 | Gender | Varchar(50) |
| 5 | Phone | Varchar(50) |

Description:-

* PatId

It is number format, use to identify the rows number.

* PatName

It is character format, use to get the patients’ name.

* Age

It is number format, use to identify the age of particular patient.

* Gender

It is character format, use to know the gender.

* Phone

It is character format, use to identify the particular patient phone number.

Table 4 : InvoiceTbl

|  |  |  |
| --- | --- | --- |
| Sr No | Name | Data Types |
| 1 | InvId | Int |
| 2 | PatId | Int |
| 3 | PatName | Varchar(50) |
| 4 | Phone | Varchar(50) |
| 5 | DeliveDate | Date |
| 6 | RefBy | Varchar(50) |
| 7 | TotCost | Int |

Description:-

* InvId

It is number format, use to identify the invoice rows’ number.

* PatId

It is number format, use to identify the patient id particularly.

* PaName

It is character format, use to identify the patient’s name individually.

* Phone

It is character format, use to identify the phone number of a registered patient.

* DeliveDate

It is date format, use to know the date of patient data delivery.

* RefBy

It is character format, use to get doctors’ reference to patient.

* TotCost

It is Int format, use to get Net Cost of patient Diagnostics.

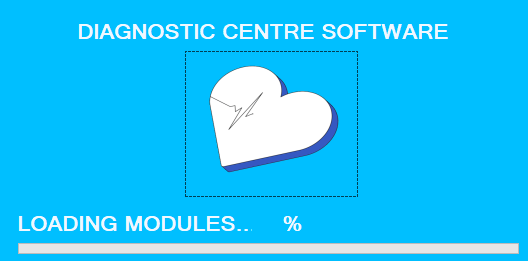
# CHAPTER NO: 5

**INPUT AND OUTPUT DESIGN**

**5.1 Admin Layout**

**Form Design**

1. Splash.Design.cs



🔄 "Loading..." 🔄

This C#.NET form application embraces the charm of anticipation! 🕰️ A delightful design 🌀 dances with excitement while your data, files, or content are prepared to appear. The form's sleek design and smooth animations 🎇 make the wait a joy. Watch as the application readies itself for action! 💫

CODING :-

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 WindowsFormsApp1

{

public partial class Splash : Form

{

public Splash()

{

InitializeComponent();

}

int startpos = 0;

private void timer1\_Tick(object sender, EventArgs e)

{

startpos += 1;

progressBar1.Value = startpos;

label3.Text = startpos + "%";

if (progressBar1.Value == 100)

{

progressBar1.Value = 0;

timer1.Stop();

Login log = new Login();

log.Show();

this.Hide();

}

}

private void Splash\_Load(object sender, EventArgs e)

{

timer1.Start();

}

private void label4\_Click(object sender, EventArgs e)

{

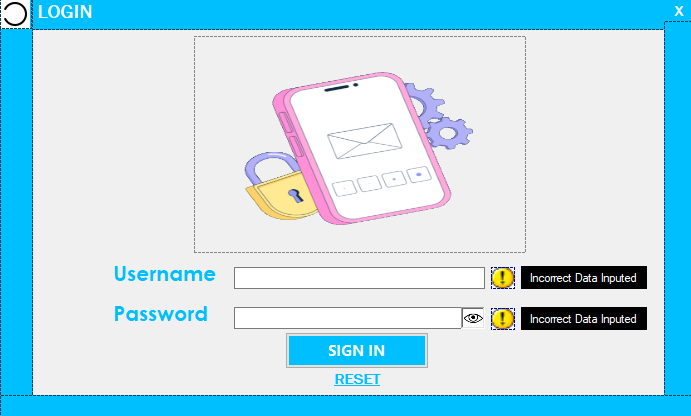
Application.Exit();

}

}

}

1. Login.Design.cs



🔒 "Login form" application in C#.NET is a secure gateway 🚪💻 that enables admin to access restricted areas or features. Users input credentials 📝✨, and the form verifies authentication before granting access. Error handling ⚠️ and password encryption 🛡️ ensure data safety. A crucial component for secure app access! 🚀🔐

CODING :-

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.Runtime.InteropServices;

namespace WindowsFormsApp1

{

public partial class Login : Form

{

public Point mouseLocation;

public Login()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.None;

}

private const int dp = 0x00020000;

protected override CreateParams CreateParams

{

get

{

CreateParams cp = base.CreateParams;

cp.ClassStyle |= dp;

return cp;

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "Admin" && textBox2.Text == "admin")

{

MessageBox.Show("Login Successfully Done");

Dashboard log = new Dashboard();

log.Show();

this.Hide();

}

else if(textBox1.Text == "" && textBox2.Text == "")

{

MessageBox.Show("Fill Details for Login");

}

else if (textBox1.Text != "Admin" && textBox2.Text != "admin")

{

MessageBox.Show("Incorrect Username or Password");

}

}

private void label3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

textBox2.UseSystemPasswordChar = false;

}

private void label5\_Click(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

pictureBox1.Hide();

pictureBox4.Hide();

pictureBox5.Hide();

}

private void Login\_Load(object sender, EventArgs e)

{

}

private void pictureBox1\_DoubleClick(object sender, EventArgs e)

{

textBox2.UseSystemPasswordChar = true;

}

private void button1\_MouseEnter(object sender, EventArgs e)

{

button1.BackColor = Color.Black;

button1.ForeColor = Color.AntiqueWhite;

}

private void button1\_MouseLeave(object sender, EventArgs e)

{

button1.BackColor = Color.DeepSkyBlue;

button1.ForeColor = Color.GhostWhite;

}

private void label5\_MouseEnter(object sender, EventArgs e)

{

//label5.BackColor = Color.Black;

label5.ForeColor = Color.Red;

}

private void label5\_MouseLeave(object sender, EventArgs e)

{

label5.ForeColor = Color.DeepSkyBlue;

}

private void label3\_MouseEnter(object sender, EventArgs e)

{

label3.ForeColor = Color.Red;

}

private void label3\_MouseLeave(object sender, EventArgs e)

{

label3.ForeColor = Color.GhostWhite;

}

private void textBox2\_Click(object sender, EventArgs e)

{

pictureBox1.Show();

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

pictureBox1.Show();

}

private void pictureBox1\_MouseEnter(object sender, EventArgs e)

{

pictureBox1.BorderStyle = BorderStyle.FixedSingle;

}

private void pictureBox1\_MouseLeave(object sender, EventArgs e)

{

pictureBox1.BorderStyle = BorderStyle.Fixed3D;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text != "Admin")

{

pictureBox4.Show();

}

else if (textBox1.Text == "Admin")

{

pictureBox4.Hide();

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text != "admin")

{

pictureBox5.Show();

}

else if (textBox2.Text == "admin")

{

pictureBox5.Hide();

}

}

private void pictureBox4\_MouseEnter(object sender, EventArgs e)

{

label6.Show();

}

private void pictureBox4\_MouseLeave(object sender, EventArgs e)

{

label6.Hide();

}

private void pictureBox5\_MouseEnter(object sender, EventArgs e)

{

label7.Show();

}

private void pictureBox5\_MouseLeave(object sender, EventArgs e)

{

label7.Hide();

}

private void Login\_MouseDown(object sender, MouseEventArgs e)

{

mouseLocation = new Point( -e.Y,-e.X);

}

private void Login\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

Point mousePose = Control.MousePosition;

mousePose.Offset(mouseLocation.X,mouseLocation.Y);

Location = mousePose;

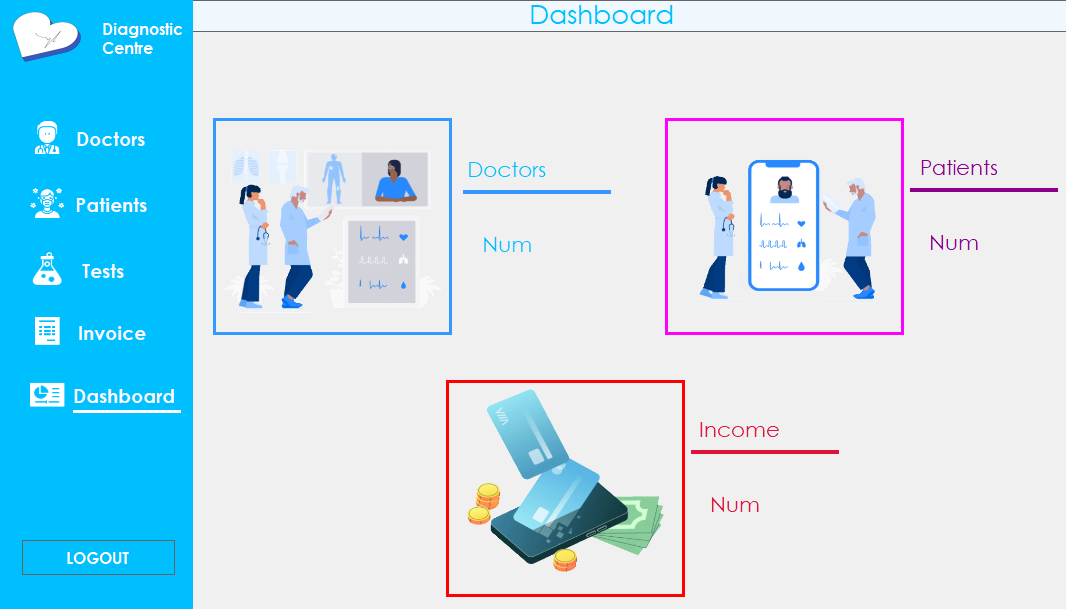
}

}

}

}

1. Dashboard.Design.cs



📊 "Dashboard" Application (C#.NET) 📊

Track essential healthcare metrics effortlessly with this dynamic diagnostic "Dashboard" form application! 🏥💻 Stay on top of vital data, including the number of Doctors 👩‍⚕️, Patients 👨‍⚕️, and Income 💰, in real-time! 🔄💡 Empower medical professionals with insights and optimize clinic performance. 🚀📈 Simplify decision-making, improve patient care, and boost financial efficiency. 🤝💙 Embrace the power of data visualization and elevate your healthcare management to new heights! 📊🏆

CODING :-

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class Dashboard : Form

{

public Dashboard()

{

InitializeComponent();

GETDocData();

GETPatData();

GETIncomeData();

}

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\Csharp(.NET)\PROJECT\WindowsFormsApp1\WindowsFormsApp1\DiagnostiDb.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True");

private void GETDocData()

{

con.Open();

SqlCommand cmd = new SqlCommand("select DocName from DoctorTbl", con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("DocName", typeof(string));

dt.Load(rdr);

DocLbl.Text = dt.Rows.Count.ToString();

con.Close();

}

private void GETPatData()

{

con.Open();

SqlCommand cmd = new SqlCommand("select PatName from PatientTbl", con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("PatName", typeof(string));

dt.Load(rdr);

PatLbl.Text = dt.Rows.Count.ToString();

con.Close();

}

private void GETIncomeData()

{

con.Open();

SqlCommand cmd = new SqlCommand("select TotCost from InvoiceTbl",con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("TotCost", typeof(string));

dt.Load(rdr);

IncomeLbl.Text = dt.Rows.Count.ToString();

con.Close();

}

private void label2\_Click(object sender, EventArgs e)

{

MessageBox.Show("Logged Out");

Login log = new Login();

log.Show();

this.Hide();

}

private void label4\_Click(object sender, EventArgs e)

{

Patient pat = new Patient();

pat.Show();

this.Hide();

}

private void label5\_Click(object sender, EventArgs e)

{

Test te = new Test();

te.Show();

this.Hide();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

Test te = new Test();

te.Show();

this.Hide();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

Patient pat = new Patient();

pat.Show();

this.Hide();

}

private void label3\_Click(object sender, EventArgs e)

{

Doctor doc = new Doctor();

doc.Show();

this.Hide();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

Doctor doc = new Doctor();

doc.Show();

this.Hide();

}

private void label6\_Click(object sender, EventArgs e)

{

Invoice i = new Invoice();

i.Show();

this.Hide();

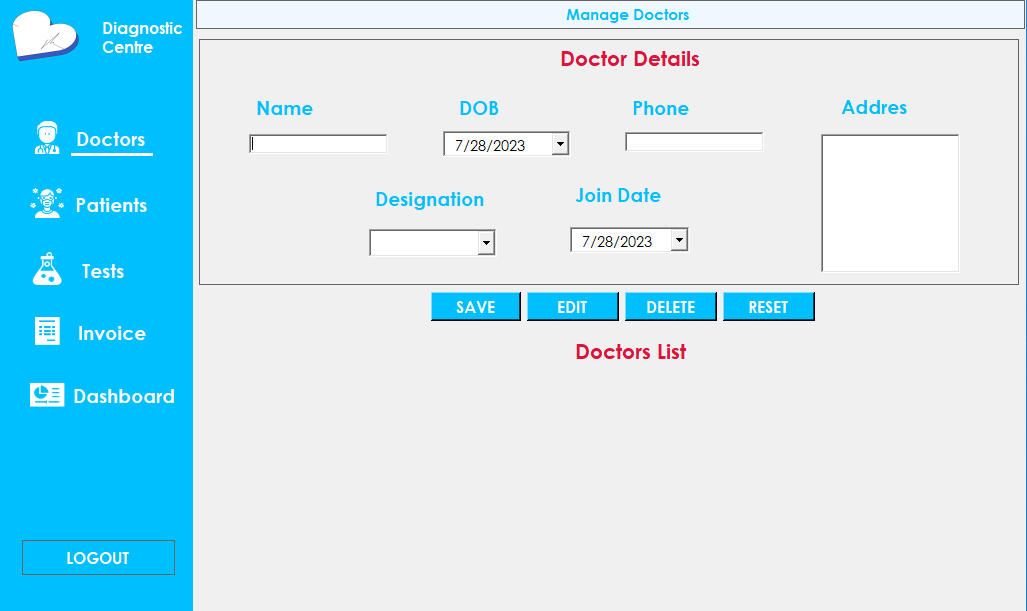
}

public string PatId { get; set; }

}

}

1. Doctor.Design.cs



🏥👩‍⚕️ "Doctor Details" Form Application 📋🔍

This C#.NET application captures essential information about doctors. Users can enter the Doctor's Name, Date of Birth (DOB), Age, Address, Phone, Designation, and Join Date. 🧑‍⚕️💼📅

The form ensures efficient record-keeping for up to 50 doctors, empowering healthcare facilities with organized and easily accessible data. 🏢💻

CODING :-

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 WindowsFormsApp1

{

public partial class Doctor : Form

{

public Doctor()

{

InitializeComponent();

populate();

}

private void label2\_Click(object sender, EventArgs e)

{

MessageBox.Show("Logged Out");

Login log = new Login();

log.Show();

this.Hide();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

Patient log = new Patient();

log.Show();

this.Hide();

}

private void label4\_Click(object sender, EventArgs e)

{

Patient log = new Patient();

log.Show();

this.Hide();

}

private void label7\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void label5\_Click(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

private void DocDGV\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

DocNameTb.Text = DocDGV.Rows[e.RowIndex].Cells[1].Value.ToString();

DocDOB.Text = DocDGV.Rows[e.RowIndex].Cells[2].Value.ToString();

DocPhone.Text = DocDGV.Rows[e.RowIndex].Cells[3].Value.ToString();

DocAdd.Text = DocDGV.Rows[e.RowIndex].Cells[4].Value.ToString();

DocDesiCb.SelectedItem = DocDGV.Rows[e.RowIndex].Cells[5].Value.ToString();

DocJoin.Text = DocDGV.Rows[e.RowIndex].Cells[6].Value.ToString();

if (DocNameTb.Text == "")

{

key = 0;

}

else

{

key = Convert.ToInt32(DocDGV.Rows[e.RowIndex].Cells[0].Value.ToString());

}

}

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\Csharp(.NET)\PROJECT\WindowsFormsApp1\WindowsFormsApp1\DiagnostiDb.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True");

private void populate()

{

con.Open();

string Query = "select \* from DoctorTbl";

SqlDataAdapter sda = new SqlDataAdapter(Query, con);

SqlCommandBuilder build = new SqlCommandBuilder(sda);

var ds = new DataSet();

sda.Fill(ds);

DocDGV.DataSource = ds.Tables[0];

con.Close();

}

int key = 0;

private void reset()

{

DocNameTb.Text = "";

DocPhone.Text = "";

DocDesiCb.SelectedIndex = -1;

DocAdd.Text = "";

key = 0;

}

private void SaveBtn\_Click(object sender, EventArgs e)

{

if (DocNameTb.Text == "" || DocPhone.Text == "" || DocDesiCb.SelectedIndex == -1 || DocAdd.Text == "")

{

MessageBox.Show("Missing Information");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("insert into DoctorTbl values

('" + DocNameTb.Text + "','" + DocDOB.Value.Date + "','" + DocPhone.Text + "','" + DocAdd.Text + "','"+DocDesiCb.SelectedItem.ToString() +"','"+DocJoin.Value.Date+"' )", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Doctor Saved Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void ResetBtn\_Click(object sender, EventArgs e)

{

reset();

}

private void DeleteBtn\_Click(object sender, EventArgs e)

{

if (key == 0)

{

MessageBox.Show("Select Doctor to delete");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("delete from DoctorTbl where DocId = '" + key + "';", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Doctor Deleted Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void EditBtn\_Click(object sender, EventArgs e)

{

if (DocNameTb.Text == "" || DocPhone.Text == "" || DocDesiCb.SelectedIndex == -1 || DocAdd.Text == "")

{

MessageBox.Show("Missing Information");

}

else

{

try

{

string Query = "update DoctorTbl set DocName = '" + DocNameTb.Text + "',DocDOB = '" + DocDOB.Value.Date + "',DocPhone = '" + DocPhone.Text + "',DocAdd = '" + DocAdd.Text + "',Designation ='"+DocDesiCb.SelectedItem.ToString()+"',Joindate = '"+DocJoin.Value.Date+"'where DocId = '" + key + "'";

con.Open();

SqlCommand cmd = new SqlCommand(Query, con);

cmd.ExecuteNonQuery();

MessageBox.Show("Doctor Updated Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void label5\_Click\_1(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

private void label6\_Click(object sender, EventArgs e)

{

Invoice inv = new Invoice();

inv.Show();

this.Hide();

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

Invoice inv = new Invoice();

inv.Show();

this.Hide();

}

}

}

1. Patient.Design.cs



🏥📋 Patient Detail Form 📋🏥

This user-friendly C#.NET app gathers essential information for medical records. It allows you to input Patient Name, Age, Mobile Number, and Gender. 📝👤📅📱💁‍♀️💁‍♂️

With a simple, intuitive interface, you can efficiently manage data for up to date and accurate medical records. 🚀💻💼

CODING :-

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class Patient : Form

{

public Patient()

{

InitializeComponent();

populate();

}

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\Csharp(.NET)\PROJECT\WindowsFormsApp1\WindowsFormsApp1\DiagnostiDb.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True");

private void label3\_Click(object sender, EventArgs e)

{

Doctor doc = new Doctor();

doc.Show();

this.Hide();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

Doctor doc = new Doctor();

doc.Show();

this.Hide();

}

private void populate()

{

con.Open();

string Query = "select \* from PatientTbl";

SqlDataAdapter sda = new SqlDataAdapter(Query,con);

SqlCommandBuilder build = new SqlCommandBuilder(sda);

var ds = new DataSet();

sda.Fill(ds);

PatientDGV.DataSource = ds.Tables[0];

con.Close();

}

private void SaveBtn\_Click(object sender, EventArgs e)

{

if (PatNameTb.Text == "" || PatAgeTb.Text == "" || PatPhoneTb.Text == "" || PatGenCb.SelectedIndex == -1)

{

MessageBox.Show("Missing Information");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("insert into PatientTbl values('" +PatNameTb.Text+ "','" +PatAgeTb.Text+ "','" +PatPhoneTb.Text+ "','"+PatGenCb.SelectedItem.ToString()+"')", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Patient Saved Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void label2\_Click(object sender, EventArgs e)

{

MessageBox.Show("Logged Out");

Login log = new Login();

log.Show();

this.Hide();

}

private void label7\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void label5\_Click(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

private void pictureBox3\_Click(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

int key = 0;

private void PatientDGV\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

PatNameTb.Text = PatientDGV.Rows[e.RowIndex].Cells[1].Value.ToString();

PatAgeTb.Text = PatientDGV.Rows[e.RowIndex].Cells[2].Value.ToString();

PatPhoneTb.Text = PatientDGV.Rows[e.RowIndex].Cells[3].Value.ToString();

PatGenCb.SelectedItem = PatientDGV.Rows[e.RowIndex].Cells[4].Value.ToString();

if (PatNameTb.Text == "")

{

key = 0;

}

else

{

key = Convert.ToInt32(PatientDGV.Rows[e.RowIndex].Cells[0].Value.ToString());

}

}

private void reset()

{

PatNameTb.Text = "";

PatAgeTb.Text = "";

PatPhoneTb.Text = "";

PatGenCb.SelectedIndex = -1;

key = 0;

}

private void DeleteBtn\_Click(object sender, EventArgs e)

{

if (key == 0)

{

MessageBox.Show("Select Patient to delete");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("delete from PatientTbl where PatId = '"+key+"';", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Patient Deleted Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void ResetBtn\_Click(object sender, EventArgs e)

{

reset();

}

private void EditBtn\_Click(object sender, EventArgs e)

{

if (PatNameTb.Text == "" || PatAgeTb.Text == "" || PatPhoneTb.Text == "" || PatGenCb.SelectedIndex == -1)

{

MessageBox.Show("Missing Information");

}

else

{

try

{

string Query = "update PatientTbl set PatName = '"+PatNameTb.Text+"',Age = '"+PatAgeTb.Text+"',Phone = '"+PatPhoneTb.Text+"',Gender = '"+PatGenCb.SelectedItem.ToString()+"'where PatId = '"+key+"'";

con.Open();

SqlCommand cmd = new SqlCommand(Query,con);

cmd.ExecuteNonQuery();

MessageBox.Show("Patient Updated Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void label6\_Click(object sender, EventArgs e)

{

Invoice inv = new Invoice();

inv.Show();

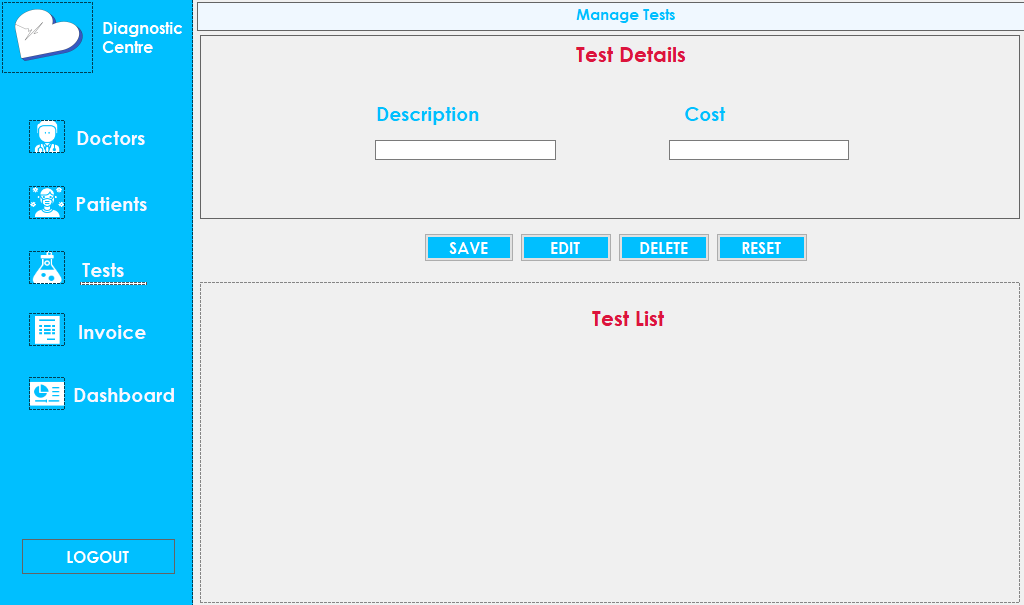
this.Hide();

}

}

}

1. Test.Design.cs



🧪 Test Detail Form 📝💻

This C#.NET application allows users to input and view essential details of diagnostic tests. The form includes a Test Description section where users can provide information about the test. Additionally, it incorporates a Test Cost field to enter the price of the test. Users can easily manage and track diagnostic tests, enhancing the efficiency of healthcare processes. 🏥💉💊

CODING :-

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class Test : Form

{

public Test()

{

InitializeComponent();

populate();

}

private void label4\_Click(object sender, EventArgs e)

{

Patient pat = new Patient();

pat.Show();

this.Hide();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

Patient pat = new Patient();

pat.Show();

this.Hide();

}

private void label3\_Click(object sender, EventArgs e)

{

Doctor pat = new Doctor();

pat.Show();

this.Hide();

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

Doctor pat = new Doctor();

pat.Show();

this.Hide();

}

private void label2\_Click(object sender, EventArgs e)

{

MessageBox.Show("Logged Out");

Login log = new Login();

log.Show();

this.Hide();

}

private void label7\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void pictureBox5\_Click(object sender, EventArgs e)

{

Dashboard das = new Dashboard();

das.Show();

this.Hide();

}

private void TestDGV\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

DescTb.Text = TestDGV.Rows[e.RowIndex].Cells[1].Value.ToString();

CostTb.Text = TestDGV.Rows[e.RowIndex].Cells[2].Value.ToString();

if (DescTb.Text == "")

{

key = 0;

}

else

{

key = Convert.ToInt32(TestDGV.Rows[e.RowIndex].Cells[0].Value.ToString());

}

}

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\Csharp(.NET)\PROJECT\WindowsFormsApp1\WindowsFormsApp1\DiagnostiDb.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True");

private void populate()

{

con.Open();

string Query = "select \* from TestTbl";

SqlDataAdapter sda = new SqlDataAdapter(Query, con);

SqlCommandBuilder build = new SqlCommandBuilder(sda);

var ds = new DataSet();

sda.Fill(ds);

TestDGV.DataSource = ds.Tables[0];

con.Close();

}

int key = 0;

private void reset()

{

DescTb.Text = "";

CostTb.Text = "";

key = 0;

}

private void SaveBtn\_Click(object sender, EventArgs e)

{

if (DescTb.Text == "" || CostTb.Text == "" )

{

MessageBox.Show("Missing Information");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("insert into TestTbl values('" + DescTb.Text + "','" + CostTb.Text + "');", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Test Saved Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void DeleteBtn\_Click(object sender, EventArgs e)

{

if (key == 0)

{

MessageBox.Show("Select Test to delete");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("delete from TestTbl where TestId = '" + key + "';", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Test Deleted Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void EditBtn\_Click(object sender, EventArgs e)

{

if (DescTb.Text == "" || CostTb.Text == "" )

{

MessageBox.Show("Missing Information");

}

else

{

try

{

string Query = "update TestTbl set TestDesc = '" + DescTb.Text + "',TestCost = '" + CostTb.Text + "'where TestId = '" + key + "'";

con.Open();

SqlCommand cmd = new SqlCommand(Query, con);

cmd.ExecuteNonQuery();

MessageBox.Show("Test Updated Successfully");

con.Close();

populate();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

private void label6\_Click(object sender, EventArgs e)

{

Invoice inv = new Invoice();

inv.Show();

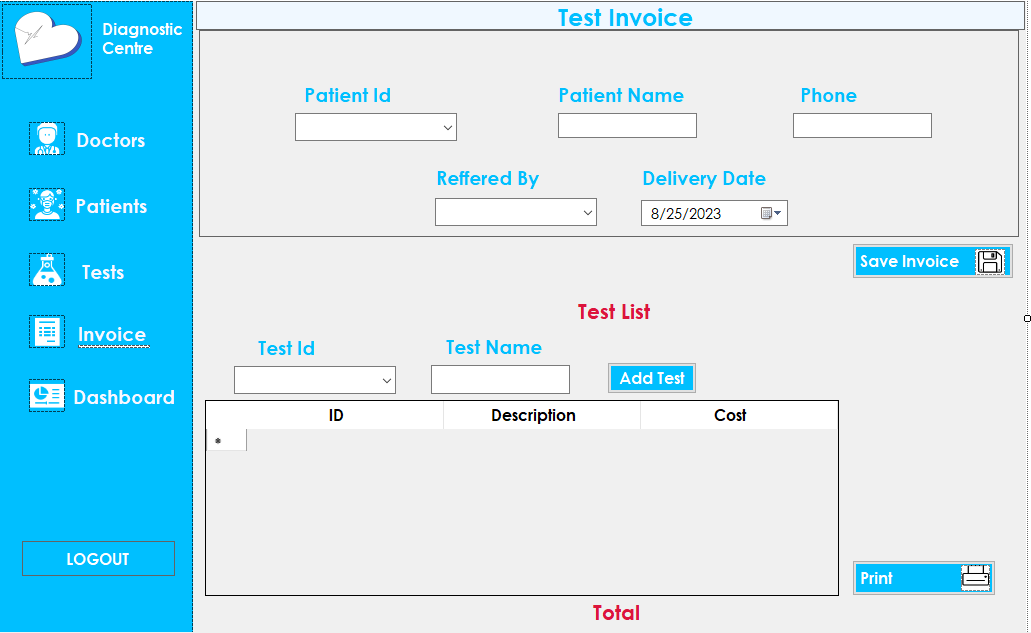
this.Hide();

}

}

}

7) Invoice.Design.cs



🧪 Test Detail Form 📝💻

This C#.NET application allows users to input and view essential details of diagnostic tests. The form includes a Test Description section where users can provide information about the test. Additionally, it incorporates a Test Cost field to enter the price of the test. Users can easily manage and track diagnostic tests, enhancing the efficiency of healthcare processes. 🏥💉💊

CODING :-

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class Invoice : Form

{

public Invoice()

{

InitializeComponent();

GETPatId();

GETDocId();

GETTestId();

}

SqlConnection con = new SqlConnection(@"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\Csharp(.NET)\PROJECT\WindowsFormsApp1\WindowsFormsApp1\DiagnostiDb.mdf;Integrated Security=True;Connect Timeout=30;User Instance=True");

private void GETPatId()

{

con.Open();

SqlCommand cmd = new SqlCommand("select PatId from PatientTbl",con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("PatId", typeof(int));

dt.Load(rdr);

PatId.ValueMember = "PatId";

PatId.DataSource = dt;

con.Close();

}

private void GETTestId()

{

con.Open();

SqlCommand cmd = new SqlCommand("select TestId from TestTbl", con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("TestId", typeof(int));

dt.Load(rdr);

TestId.ValueMember = "TestId";

TestId.DataSource = dt;

con.Close();

}

private void GETDocId()

{

con.Open();

SqlCommand cmd = new SqlCommand("select DocName from DoctorTbl", con);

SqlDataReader rdr;

rdr = cmd.ExecuteReader();

DataTable dt = new DataTable();

dt.Columns.Add("DocName", typeof(string));

dt.Load(rdr);

RefBy.ValueMember = "DocName";

RefBy.DataSource = dt;

con.Close();

}

private void GETPatData()

{

con.Open();

string sql = "select \* from PatientTbl where PatId = "+PatId.SelectedValue.ToString()+"";

SqlCommand cmd = new SqlCommand(sql,con);

DataTable dt = new DataTable();

SqlDataAdapter sda = new SqlDataAdapter(cmd);

sda.Fill(dt);

foreach (DataRow dr in dt.Rows)

{

PatName.Text = dr["PatName"].ToString();

PatPhone.Text = dr["Phone"].ToString();

}

con.Close();

}

int Cost;

private void GETTestData()

{

con.Open();

string sql = "select \* from TestTbl where TestId = " + TestId.SelectedValue.ToString() + "";

SqlCommand cmd = new SqlCommand(sql, con);

DataTable dt = new DataTable();

SqlDataAdapter sda = new SqlDataAdapter(cmd);

sda.Fill(dt);

foreach (DataRow dr in dt.Rows)

{

TestName.Text = dr["TestDesc"].ToString();

Cost = Convert.ToInt32(dr["TestCost"].ToString());

}

con.Close();

}

private void label3\_Click(object sender, EventArgs e)

{

Doctor doc = new Doctor();

doc.Show();

this.Hide();

}

private void label4\_Click(object sender, EventArgs e)

{

Patient pa = new Patient();

pa.Show();

this.Hide();

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

Test tc = new Test();

tc.Show();

this.Hide();

}

private void label7\_Click(object sender, EventArgs e)

{

Dashboard d = new Dashboard();

d.Show();

this.Hide();

}

private void label2\_Click(object sender, EventArgs e)

{

Login log = new Login();

log.Show();

this.Hide();

}

private void PatId\_SelectionChangeCommitted(object sender, EventArgs e)

{

GETPatData();

}

private void TestId\_SelectionChangeCommitted(object sender, EventArgs e)

{

GETTestData();

}

private void AddBtn\_Click(object sender, EventArgs e)

{

if (TestName.Text == "")

{

MessageBox.Show("Select The Test");

}

else

{

DataGridViewRow dg = new DataGridViewRow();

dg.CreateCells(InvDGV);

dg.Cells[0].Value = n + 1;

dg.Cells[1].Value = TestName.Text;

dg.Cells[2].Value = Cost;

InvDGV.Rows.Add(dg);

n++;

GrdTotal = GrdTotal + Cost;

TotalLbl.Text = "Rs" + GrdTotal;

}

}

int n = 0, GrdTotal = 0;

int TestCost, pos = 60;

private void PriBtn\_Click(object sender, EventArgs e)

{

printDocument1.DefaultPageSettings.PaperSize = new System.Drawing.Printing.PaperSize("pprnm",285,600);

if (printPreviewDialog1.ShowDialog() == DialogResult.OK)

{

printDocument1.Print();

}

}

private void printDocument1\_PrintPage\_1(object sender, System.Drawing.Printing.PrintPageEventArgs e)

{

int TestId;

string TestName;

e.Graphics.DrawString("Diagnostic Centre", new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Red, new Point(80));

e.Graphics.DrawString("ID TEST COST", new Font("Century Gothic", 10, FontStyle.Bold), Brushes.Red, new Point(75, 40));

foreach (DataGridViewRow row in InvDGV.Rows)

{

TestId = Convert.ToInt32(row.Cells["Column1"].Value);

TestName = "" + row.Cells["Column2"].Value;

TestCost = Convert.ToInt32(row.Cells["Column3"].Value);

e.Graphics.DrawString("" + TestId, new Font("Century Gothic ", 8, FontStyle.Bold), Brushes.Blue, new Point(75, pos));

e.Graphics.DrawString("" + TestName, new Font("Century Gothic ", 8, FontStyle.Bold), Brushes.Blue, new Point(120, pos));

e.Graphics.DrawString("" + TestCost, new Font("Century Gothic ", 8, FontStyle.Bold), Brushes.Blue, new Point(210, pos));

pos = pos + 25;

}

e.Graphics.DrawString("Grand Total: Rs" + GrdTotal, new Font("Century Gothic", 12, FontStyle.Bold), Brushes.Crimson, new Point(50, pos + 50));

e.Graphics.DrawString("\*\*\*\*\*\*\*\*\*\*\*\*Diagnostic Centre\*\*\*\*\*\*\*\*\*\*\*\*\*", new Font("Century Gothic", 10, FontStyle.Bold), Brushes.Crimson, new Point(11, pos + 85));

InvDGV.Rows.Clear();

InvDGV.Refresh();

pos = 100;

GrdTotal = 0;

}

private void reset()

{

PatId.Text = "";

PatName.Text = "";

PatPhone.Text = "";

TotalLbl.Text = "Total";

GrdTotal = 0;

TestId.Text = "";

TestName.Text = "";

InvDGV.Rows.Clear();

}

private void SaveInBtn\_Click(object sender, EventArgs e)

{

if (PatId.Text == "" || RefBy.SelectedIndex == -1 || TotalLbl.Text == "Total")

{

MessageBox.Show("Missing Information");

}

else

{

try

{

con.Open();

SqlCommand cmd = new SqlCommand("insert into InvoiceTbl values('" + PatId.SelectedValue.ToString() + "','" + PatName.Text + "','" + PatPhone.Text + "','" +DeliDate.Value.Date + "','"+RefBy.SelectedValue.ToString()+"','"+GrdTotal+"')", con);

cmd.ExecuteNonQuery();

MessageBox.Show("Invoice Saved Successfully");

con.Close();

reset();

}

catch (Exception Ex)

{

MessageBox.Show(Ex.Message);

}

}

}

}

}

# CHAPTER NO: 6 LIMITATIONS AND FUTURE ENHANCEMENT

**6.1 Limitations**

**6.2 Future Enhancement**

## 6.1 Limitations

* Only works in Windows OS.
* No Remote access.
* No Backup And Restore Utilities Are Incorporated.
* We are not accepting online payment using any credit card or net banking for security reason.

## 6.2 Future Enhancement

* We will also provide a web site to customer can check about payment, invoice information and about product details.
* We will try to provide A.I in future in case.

**CHAPTER** **NO: 7**

**CONCLUSION**

**7.1 Conclusion**

**7.2 Advantages**

## 7.1 Conclusion

* The application manages diagnostic centre.

## 7.2 Advantages

* The application makes easy the difficult calculation of data table.
* User friendly interface.
* Fast access to database.
* Reliable and efficient.
* Security of data.
* Easy to manage information.

**CHAPTER** **NO: 8**

**BIBILIOGRAPHY**

## Bibliography

* Introduction to .NET framework - Worx publication.
* C# 5.0 and .NET 4.5 Framework (By: Andrew Troelsen )

**CHAPTER** **NO: 9**

**REFERENCES**

## References

We are really thankful to our guider Prof. Mr. ZALAK THAKRAR to guide us and inspire us. We also Thankful to the whole staff of computer Department to gives us a huge support in our project.

Web sites:

* [www.stackoverflow.com](http://www.stackoverflow.com/)  [www.codeproject.com](http://www.codeproject.com/)
* [www.c-sharpcorner.com](http://www.c-sharpcorner.com/)
* [www.javatpoint.com](http://www.javatpoint.com/)
* [www.geeksforgeeks.org](http://www.geeksforgeeks.org/)