

Final Project Report

Course: Software Construction & Development Lab

Contact Management System

Submitted By

Sir Fayyaz Khan

Abdullah (Group Leader)

FA21-13694

Hashir Ahmed

FA21-13948

M.Irteza Waseem

FA21-13952

Department of Software Engineering



KARACHI INSTITUTE OF ECONOMICS & TECHNOLOGY
College of Engineering

ABSTRACT

A **Contact Management System** (CMS) is a software solution designed to help users efficiently store, organize, and manage personal and professional contact information. This system enables users to store details such as names, phone numbers, email addresses, physical addresses, and other relevant information for each contact. It offers features such as search functionality, categorization, and tagging to help users quickly retrieve and manage their contacts.

The system may also include additional features like integration with email services, reminders for follow-ups, and the ability to import/export contacts. The CMS aims to streamline communication processes, enhance organization, and improve productivity for individuals and businesses by providing easy access to an organized database of contacts.

The design emphasizes usability, data security, and scalability, ensuring the system can be adapted to both personal and enterprise-level needs. Ultimately, the Contact Management System simplifies contact handling, allowing users to focus on building and maintaining stronger relationships.

TABLE OF CONTENTS

ABSTRACT

1.	INTRODUCTION TO PROBLEM	5
1.1	Introduction	5
	Problem Statement	5
1.2	Purpose	5
	Project Significance	5
1.3	Objective	5
1.4	Existing solution	6
1.5	Purposed Solution	6
1.6	Scope	8
2	Literature Review	10
3	DESIGN & IMPLEMENTATION	12
3.1	Use Case Diagram:	12
3.2	Flow Chart:	13
4	TESTING	15
4.1	Testing Procedure & Test Cases:	15
4.2	Results	15
5	Future Enhancements	19

Appendix A: Bibliography / References

Appendix B: Index

CHAPTER # 1
INTRODUCTION TO PROBLEM

1. INTRODUCTION TO PROBLEM

1.1 Introduction

This Contact Management System (CMS) is a desktop-based application developed using C# Windows Forms with an SQLite backend database. The system allows users to manage their contact details, including names, phone numbers, emails, and addresses. In addition, it includes basic user management for authentication.

Problem Statement

In the modern world, individuals and businesses have a large number of contacts, including clients, customers, employees, friends, and family. Managing contact information efficiently and securely has become a significant challenge. The traditional methods, such as maintaining physical address books, spreadsheets, or disorganized notes, are inefficient, error-prone, and lack scalability. In addition, searching for certain contact details and maintaining data integrity that is free from duplication cannot be performed easily using manual or archaic systems. The system developed here is meant to solve the above issues as it provides a Contact Management System: an easy, safe, and efficient application designed for managing contact details.

1.2 Purpose

The main objective of Contact Management System (CMS) is that it should be an effective, organized, and easy-to-use method for saving, managing, and accessing contact information. This approach eliminates the drawbacks of traditional methods and old systems without compromising data integrity, security, and accessibility.

Project Significance

The Contact Management System (CMS) is significant in terms of modern-day challenges associated with organizing and managing contact information. It improves productivity, reduces manual errors, and ensures better data integrity for both individuals and organizations through a secure, efficient, and user-friendly solution.

1.3 Objective

Objects of the Design are:

- To give a veritably simple, stoner-friendly Contact Management System.
- To apply introductory smut operations, produce, Read, Update, and cancel.
- To Allow searching for connections snappily using criteria similar as Name, Dispatch, or communicate Number.
- To Store data persistently using SQLite Database.
- To apply stoner authentication for secure access.

1.4 Existing solution

Contact information has been a necessity for both individuals and organizations for many decades. There were several methods and tools used to manage and store contacts before the advent of advanced digital solutions such as the Contact Management System (CMS). However, existing solutions have limitations that the CMS aims to overcome.

1.5 Purposed Solution

The Contact Management System (CMS) is designed as an efficient, secure, and user-friendly solution to address the limitations of existing contact management methods. It is a desktop-based application developed using C# and SQLite for managing contact records. The proposed system focuses on automating, simplifying, and improving the process of storing, retrieving, updating, and deleting contact details for both individuals and organizations.

i. Secure User Authentication

- The system ensures data security by providing a login mechanism.
- Only authorized users can access or manipulate the stored contact records.
- This prevents unauthorized access to sensitive contact information.

ii. Contact Record Management

- The system offers full CRUD (Create, Read, Update, Delete) operations for contact records.
- Users can add, edit, delete, and view contact information with ease.

iii. Unique and Structured Data Storage

- The database enforces **unique constraints** to avoid duplicate contact entries.
- Contacts are stored in a **structured manner** with fields such as:
 - Name
 - Father Name
 - Gender
 - Address
 - City

- Company
- Designation
- Email
- Multiple Contact Numbers (Contact No. 1, 2, and 3).

iv. **Search Functionality**

- Users can quickly **search contacts** based on specific fields like:
 - **Name**
 - **Email**
 - **Contact Number**
- This improves accessibility and reduces the time taken to retrieve data.

v. **Filtering and Sorting**

- The system allows contacts to be **filtered** based on:
 - Gender
 - City
 - Company
- This feature is useful for organizations that need categorized or segmented contact lists.

vi. **User-Friendly Interface**

- The graphical stoner interface (GUI) is intuitive and easy to navigate.
- Buttons for operations such as **Add New**, **Save**, **Edit**, **Delete**, **Show Data**, and **Search** are clearly labelled for smooth usage.
- The design ensures a **low learning curve**, making it accessible for users with minimal technical knowledge.

vii. **Data Validation**

- Input fields are validated to ensure that required fields are filled and data is in the correct format.
- For instance, unique constraints are applied to **Email** and **Contact Numbers** to prevent duplication.

viii. Scalability

- The system uses **SQLite**, a lightweight and scalable database that allows for efficient data management.
- It can manage a growing volume of contact records without performance issues.

ix. Database Auto-Generation

- If the database does not exist, the system automatically creates one and initializes necessary tables.

1.6 Scope

The Contact Management System (CMS) is a desktop-based application designed to streamline and automate the process of managing contacts. The scope of the project is defined based on the target users, system functionalities, and operational boundaries.

CHAPTER # 2
Literature Review

2 Literature Review

Smart Social Contact Management System for Better Memory Recall.

Shailesh U. Sambhe, Sachin Murab

Date of Issue: 09, March 2014

Link: https://www.ijeit.com/Vol%203/Issue%209/IJEIT1412201403_42.pdf

Description:

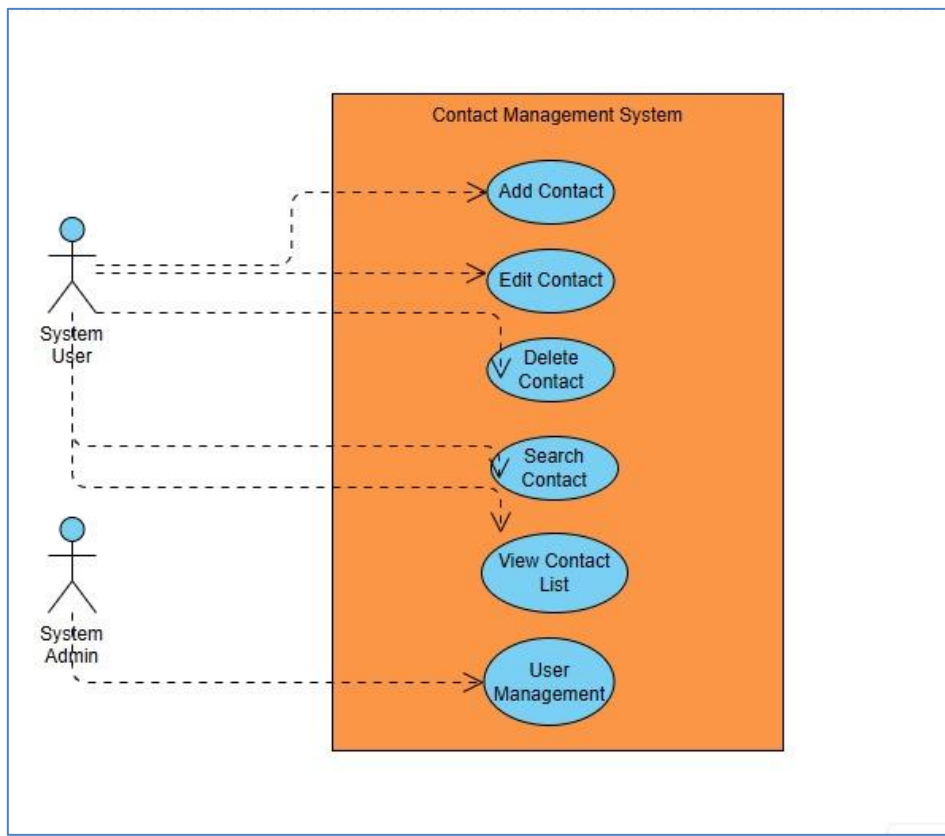
The paper introduces the Social Contact Manager (SCM), an intelligent system designed to help users manage and recall social contacts more effectively. It addresses the issue of forgetting names and details of acquaintances by automatically collecting rich contact data through sensors and web sources. SCM also features associative search, allowing users to retrieve contact information using memory cues. The system was developed based on survey insights and evaluated through user studies, which demonstrated its effectiveness in improving contact recall and highlighted patterns in how people remember social connections.

CHAPTER # 3
Design & Implementation

3 DESIGN & IMPLEMENTATION

The **Contact Management System** (CMS) is designed to streamline and organize user contact data efficiently. The system allows for input, retrieval, and management of user details such as Name, Address, Email, Company, and Contact Numbers. The graphical user interface (GUI) provides a user-friendly experience and ensures ease of navigation.

3.1 Use Case Diagram:



3.2 Flow Chart:



CHAPTER # 4

Tests & Results

4 TESTING

Testing is a crucial phase in the software development lifecycle that ensures the system functions correctly and meets user requirements. The **Contact Management System (CMS)** underwent comprehensive testing to verify its functionality, usability, and performance. The primary goals of testing were to identify errors, validate data accuracy, and ensure a seamless user experience.

4.1 Testing Procedure & Test Cases:

- **Functional Testing:**

Test Case	Expected Result	Actual Result
Add a new contact	Contact saved successfully	Pass
Search contact by name	Relevant contact displayed	Pass
Edit contact details	Updated information saved	Pass

- **Boundary Testing:**

Test Case	Input	Expected Result	Actual Result
Add duplicate contact	Same Contact No.	Error: Duplicate entry	Pass
Leave required fields blank	Blank Name/Email	Error: Fields cannot be empty	Pass

4.2 Results

The screenshot displays the 'Contacts' application window. On the left, there is a form for adding a new contact with fields for Name, Father Name, Gender, Address, City, Company, Designation, Email, and three Contact No. fields. A 'Search' button is located next to the Email field. Below the form are buttons for 'User Management', 'Save', 'Add New', 'Delete', 'Abort Edit', 'Edit', and 'Show Data'. On the right, there is a table of existing contacts with columns for Name, Father Name, Gender, City, Company, and Designation. The table contains one entry: Hashir, Khalid, Male, Karachi, Microsoft, Software Engineer. A 'Search' button is also present above the table.

Name	Father Name	Gender	City	Company	Designation
Hashir	Khalid	Male	Karachi	Microsoft	Software Engineer

User

User ID

User Name

Password

Search

Delete

Abort Generate New

▼

User ID

User Name

Current Password

New Password

Retype New Password

A

Change Password

User ID	User Name
1	

✖ TestRowCounter		2 ms
✔ Add_NewUser_Add	CMSBU.Tests.DatabaSeServerTests.TestDatabaseCreation	1 sec
✔ Delete_User_PromptsConfirmation		10 sec

CHAPTER # 5
Future Enhancements

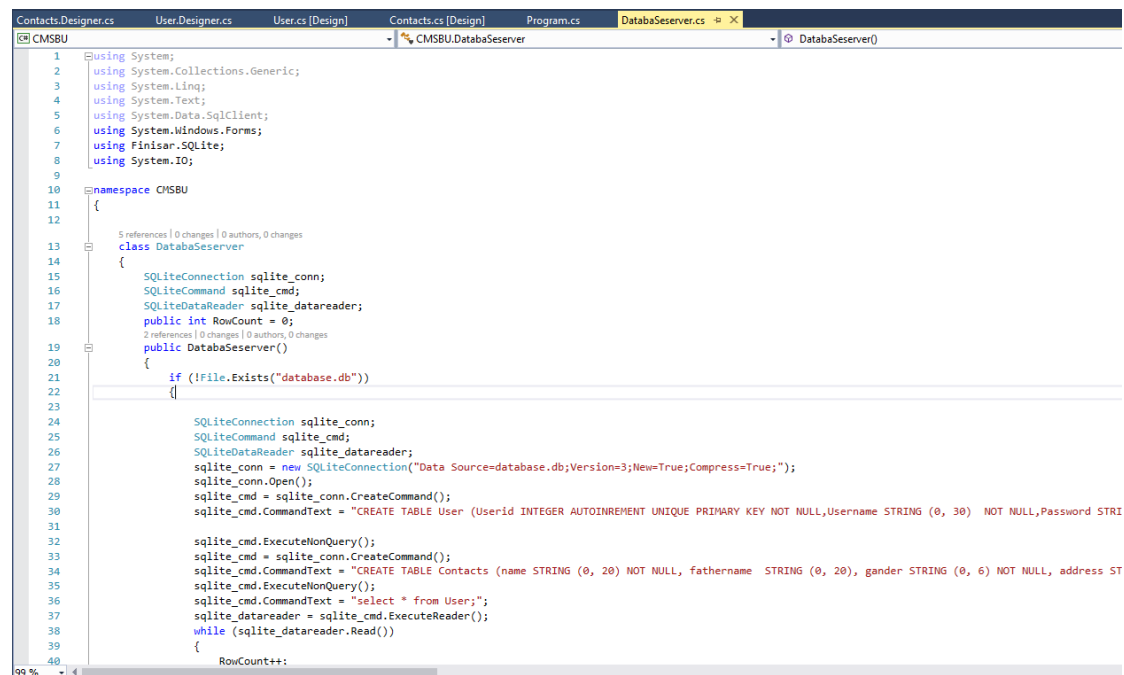
5 Future Enhancements

Following are the tools and ways used for the medication of design

The Contact Management System (CMS) can be further enhanced to include advanced features and functionalities for better scalability, usability, and effectiveness. Below are the proposed future advancements for the design, enforced using C# and SQLite in Visual Studio.

CODE:

DatabaseHandler:



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Data.SqlClient;
6 using System.Windows.Forms;
7 using Finisar.SQLite;
8 using System.IO;
9
10 namespace CMSBU
11 {
12     5 references | 0 changes | 0 authors, 0 changes
13     class DatabaSeServer
14     {
15         SQLiteConnection sqlite_conn;
16         SQLiteCommand sqlite_cmd;
17         SQLiteDataReader sqlite_datareader;
18         public int RowCount = 0;
19         2 references | 0 changes | 0 authors, 0 changes
20         public DatabaSeServer()
21         {
22             if (!File.Exists("database.db"))
23             {
24                 SQLiteConnection sqlite_conn;
25                 SQLiteCommand sqlite_cmd;
26                 SQLiteDataReader sqlite_datareader;
27                 sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=True;Compress=True;");
28                 sqlite_conn.Open();
29                 sqlite_cmd = sqlite_conn.CreateCommand();
30                 sqlite_cmd.CommandText = "CREATE TABLE User (UserId INTEGER AUTOINREMENT UNIQUE PRIMARY KEY NOT NULL,Username STRING (8, 30) NOT NULL,Password STRI
31
32                 sqlite_cmd.ExecuteNonQuery();
33                 sqlite_cmd = sqlite_conn.CreateCommand();
34                 sqlite_cmd.CommandText = "CREATE TABLE Contacts (name STRING (8, 20) NOT NULL, fathename STRING (8, 20), gander STRING (8, 6) NOT NULL, address ST
35                 sqlite_cmd.ExecuteNonQuery();
36                 sqlite_cmd.CommandText = "select * from User;";
37                 sqlite_datareader = sqlite_cmd.ExecuteReader();
38                 while (sqlite_datareader.Read())
39                 {
40                     RowCount++;
41                 }
42             }
43         }
44     }
45 }
```

```

35         sqlite_cmd.ExecuteNonQuery();
36         sqlite_cmd.CommandText = "select * from User;";
37         sqlite_datareader = sqlite_cmd.ExecuteReader();
38         while (sqlite_datareader.Read())
39         {
40             RowCount++;
41         }
42         MessageBox.Show("Database Successfully Generated" + RowCount);
43         sqlite_conn.Close();
44     }
45 }
46
47
48
49 1 reference | 0 changes | 0 authors, 0 changes
50 public void adduser(string username, string password)
51 {
52     sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
53     sqlite_conn.Open();
54     sqlite_cmd = sqlite_conn.CreateCommand();
55     sqlite_cmd.CommandText = "SELECT * FROM User";
56
57     sqlite_datareader = sqlite_cmd.ExecuteReader();
58     while (sqlite_datareader.Read())
59     {
60         RowCount = int.Parse(sqlite_datareader.GetValue(0).ToString());
61     }
62     sqlite_datareader.Close();
63     if (RowCount <= 0)
64     {
65         try
66         {
67             string cmd = "INSERT into User VALUES(" + (RowCount + 1) + ", " + username + ", " + password + ")";
68             sqlite_cmd.CommandText = cmd;
69
70             sqlite_cmd.ExecuteNonQuery();
71             sqlite_conn.Close();
72             MessageBox.Show("Your account is generated");
73         }
74     }

```

```

76     }
77 }
78
79
80 0 references | 0 changes | 0 authors, 0 changes
81 public int rowcounter()
82 {
83     sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
84     sqlite_conn.Open();
85     sqlite_cmd = sqlite_conn.CreateCommand();
86     sqlite_cmd.CommandText = "SELECT * FROM User";
87
88     sqlite_datareader = sqlite_cmd.ExecuteReader();
89     while (sqlite_datareader.Read())
90     {
91         RowCount++;
92     }
93     sqlite_datareader.Close();
94     sqlite_conn.Close();
95     return RowCount;
96 }
97
98 0 references | 0 changes | 0 authors, 0 changes
99 public int cridentals(string username, string password)
100 {
101     sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
102     sqlite_conn.Open();
103     sqlite_cmd = sqlite_conn.CreateCommand();
104     sqlite_cmd.CommandText = "SELECT * FROM User where Username='" + username + "' and password='" + password + "'";
105     sqlite_datareader = sqlite_cmd.ExecuteReader();
106     while (sqlite_datareader.Read())
107     {
108         RowCount++;
109     }
110     sqlite_datareader.Close();
111     sqlite_conn.Close();
112     return RowCount;
113 }
114
115 5 references | 0 changes | 0 authors, 0 changes
116 public SQLiteDataReader dewdata(string statement)

```

Output

```
110         sqlite_conn.Close();
111         return RowCount;
112     }
113 }
114 public SQLiteDataReader dgwdata(string statement)
115 {
116     sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
117     sqlite_conn.Open();
118     sqlite_cmd = sqlite_conn.CreateCommand();
119     sqlite_cmd.CommandText = statement;
120     sqlite_datareader = sqlite_cmd.ExecuteReader();
121     return sqlite_datareader;
122 }
123 }
124 }
125 internal void savecontacts(string statement)
126 {
127     try
128     {
129         sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
130         sqlite_conn.Open();
131         sqlite_cmd = sqlite_conn.CreateCommand();
132         sqlite_cmd.CommandText = statement;
133         sqlite_cmd.ExecuteNonQuery();
134         sqlite_conn.Close();
135         MessageBox.Show("Contact is stored inside database");
136     }
137     catch (SQLiteException e)
138     {
139         MessageBox.Show(e.ToString());
140     }
141 }
142 public void commandexecutor(string cmd)
143 {
144     try
145     {
146         sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
147         sqlite_conn.Open();
148         sqlite_cmd = sqlite_conn.CreateCommand();
```

```
142     public void commandexecutor(string cmd)
143     {
144         try
145         {
146             sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
147             sqlite_conn.Open();
148             sqlite_cmd = sqlite_conn.CreateCommand();
149             sqlite_cmd.CommandText = cmd;
150             sqlite_cmd.ExecuteNonQuery();
151             sqlite_conn.Close();
152         }
153         catch (SQLiteException e)
154         {
155             MessageBox.Show(e.ToString());
156         }
157     }
158 }
159 }
```

TEST:

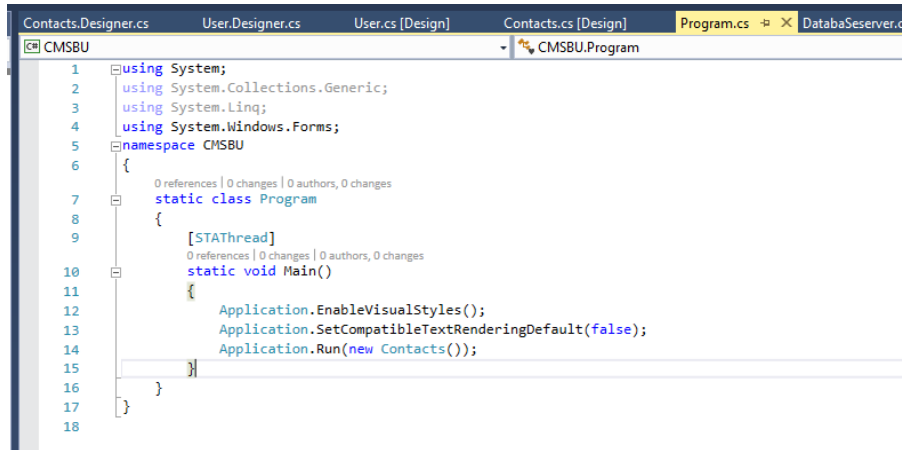
```
SQLiteDataReader.cs  DatabaSeServerTests.cs*  UserTests.cs  User.cs  ContactsTests.cs  Contacts.cs  User.cs [Design]  Program.cs
CMSBU.Tests  CMSBU.Tests.DatabaSeServerTests  Cleanup()

1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3  using System.IO;
4  //using Finisar.SQLite;
5  using System.Data;
6
7  namespace CMSBU.Tests
8  {
9      [TestClass]
10     // 0 references | 0 changes | 0 authors, 0 changes
11     public class DatabaSeServerTests
12     {
13         private const string TestDatabasePath = "test_database.db";
14         private DatabaSeServer _databaseServer;
15
16         [TestInitialize]
17         // 0 references | 0 changes | 0 authors, 0 changes
18         public void Setup()
19         {
20             if (File.Exists(TestDatabasePath))
21             {
22                 File.Delete(TestDatabasePath);
23             }
24             _databaseServer = new DatabaSeServer();
25         }
26
27         [TestMethod]
28         // 0 references | 0 changes | 0 authors, 0 changes
29         public void TestDatabaseCreation()
30         {
31             Assert.IsTrue(File.Exists(TestDatabasePath), "Database file was not created.");
32         }
33
34         [TestMethod]
35         // 0 references | 0 changes | 0 authors, 0 changes
36         public void TestAddUser()
37         {
38             string username = "testuser";
39             string password = "password123";
40
41             _databaseServer.adduser(username, password);
42             int userCount = _databaseServer.rowcounter();
43             Assert.AreEqual(1, userCount, "User was not added correctly.");
44         }
45
46         [TestMethod]
47         // 0 references | 0 changes | 0 authors, 0 changes
48         public void TestRowCounter()
49         {
50             _databaseServer.adduser("user1", "password1");
51             _databaseServer.adduser("user2", "password2");
52         }
53     }
54 }

30 %
QuitTest
```

```
42
43
44 [TestMethod]
45 // 0 references | 0 changes | 0 authors, 0 changes
46 public void TestRowCounter()
47 {
48     _databaseServer.adduser("user1", "password1");
49     _databaseServer.adduser("user2", "password2");
50
51     int rowCount = _databaseServer.rowcounter();
52     Assert.AreEqual(2, rowCount, "Row counter is not working correctly.");
53 }
54
55 [TestMethod]
56 // 0 references | 0 changes | 0 authors, 0 changes
57 public void TestCredentialValidation()
58 {
59     _databaseServer.adduser("testuser", "password123");
60
61     int validCredentials = _databaseServer.cridentials("testuser", "password123");
62     Assert.AreEqual(1, validCredentials, "Credentials validation failed.");
63
64     int invalidCredentials = _databaseServer.cridentials("wronguser", "wrongpassword");
65     Assert.AreEqual(0, invalidCredentials, "Invalid credentials passed validation.");
66 }
67
68 [TestCleanup]
69 // 0 references | 0 changes | 0 authors, 0 changes
70 public void Cleanup()
71 {
72     if (File.Exists(TestDatabasePath))
73     {
74         File.Delete(TestDatabasePath);
75     }
76 }
77 }
```

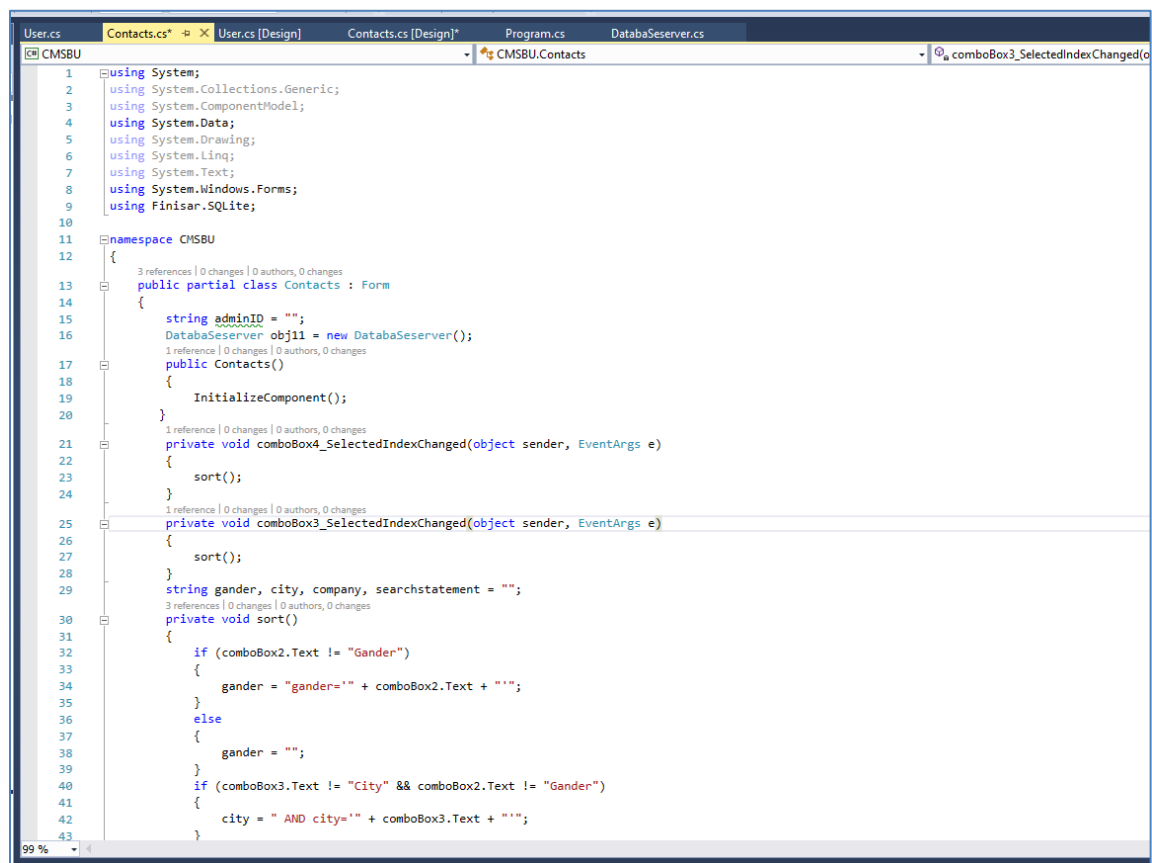
Program.cs:



The screenshot shows the Visual Studio IDE with the 'Program.cs' file open. The file is part of the 'CMSBU' project. It contains the following code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Windows.Forms;
5 namespace CMSBU
6 {
7     0 references | 0 changes | 0 authors, 0 changes
8     static class Program
9     {
10         [STAThread]
11         0 references | 0 changes | 0 authors, 0 changes
12         static void Main()
13         {
14             Application.EnableVisualStyles();
15             Application.SetCompatibleTextRenderingDefault(false);
16             Application.Run(new Contacts());
17         }
18     }
19 }
```

Contacts.cs:



The screenshot shows the Visual Studio IDE with the 'Contacts.cs' file open. The file is part of the 'CMSBU' project. It contains the following code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9 using Finisar.SQLite;
10
11 namespace CMSBU
12 {
13     3 references | 0 changes | 0 authors, 0 changes
14     public partial class Contacts : Form
15     {
16         string adminID = "";
17         DatabaSeServer obj11 = new DatabaSeServer();
18         public Contacts()
19         {
20             InitializeComponent();
21         }
22         1 reference | 0 changes | 0 authors, 0 changes
23         private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
24         {
25             sort();
26         }
27         1 reference | 0 changes | 0 authors, 0 changes
28         private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
29         {
30             sort();
31         }
32         string gander, city, company, searchstatement = "";
33         3 references | 0 changes | 0 authors, 0 changes
34         private void sort()
35         {
36             if (comboBox2.Text != "Gander")
37             {
38                 gander = "gander=" + comboBox2.Text + "&&";
39             }
40             else
41             {
42                 gander = "";
43             }
44             if (comboBox3.Text != "City" && comboBox2.Text != "Gander")
45             {
46                 city = " AND city=" + comboBox3.Text + "&&";
47             }
48         }
49     }
50 }
```

```

40         if (comboBox3.Text != "City" && comboBox2.Text != "Gender")
41         {
42             city = " AND city='" + comboBox3.Text + "'";
43         }
44         else if (comboBox3.Text != "City" && comboBox2.Text == "Gender")
45         {
46             city = "city='" + comboBox3.Text + "'";
47         }
48         else
49         {
50             city = "";
51         }
52         if (comboBox4.Text != "Company" && (comboBox2.Text != "Gender" || comboBox3.Text != "City"))
53         {
54             company = " AND company='" + comboBox4.Text + "'";
55         }
56         else if (comboBox4.Text != "Company" && (comboBox2.Text == "Gender" && comboBox3.Text == "City"))
57         {
58             company = "company='" + comboBox4.Text + "'";
59         }
60         else
61         {
62             company = "";
63         }
64         if (gander != "" || city != "" || company != "")
65         {
66             searchstatement = "select * from Contacts where " + gander + city + company;
67         }
68         else
69         {
70             searchstatement = "select * from Contacts";
71         }
72         MessageBox.Show(searchstatement);
73         loaddataintogrid(searchstatement);
74     }
75     1 reference | 0 changes | 0 authors, 0 changes
76     private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
77     {
78         sort();
79     }
80     1 reference | 0 changes | 0 authors, 0 changes
81     private void button1_Click(object sender, EventArgs e)
82     {
83         obj11.savecontacts("INSERT INTO contacts VALUES('" + txtname.Text + "','" +
84             + txtfather.Text + "','" + gender.Text + "','" + txtaddress.Text + "','" + txtcity.Text
85             + "','" + txtcompany.Text + "','" + txtdesignation.Text + "','" + txtemail.Text + "','" +
86             + txtcontact1.Text + "','" + txtcontact2.Text + "','" + txtcontact3.Text + "')");

```

```

80     {
81         obj11.savecontacts("INSERT INTO contacts VALUES('" + txtname.Text + "','" +
82             + txtfather.Text + "','" + gender.Text + "','" + txtaddress.Text + "','" + txtcity.Text
83             + "','" + txtcompany.Text + "','" + txtdesignation.Text + "','" + txtemail.Text + "','" +
84             + txtcontact1.Text + "','" + txtcontact2.Text + "','" + txtcontact3.Text + "')");
85         loaddataintogrid("Select * from Contacts");
86         addcombodata();
87         contentdisable();
88     }
89     7 references | 0 changes | 0 authors, 0 changes
90     private void loaddataintogrid(string statementt)
91     {
92         SQLiteDataReader sdr = obj11.dgwdata(statementt);
93         DataTable dt = new DataTable();
94         dt.Columns.Add(new DataColumn("Name", typeof(string)));
95         dt.Columns.Add(new DataColumn("Father Name", typeof(string)));
96         dt.Columns.Add(new DataColumn("Gender", typeof(string)));
97         dt.Columns.Add(new DataColumn("City", typeof(string)));
98         dt.Columns.Add(new DataColumn("Company", typeof(string)));
99         dt.Columns.Add(new DataColumn("Designation", typeof(string)));
100        dt.Columns.Add(new DataColumn("Email", typeof(string)));
101        dt.Columns.Add(new DataColumn("Contact1", typeof(string)));
102        dt.Columns.Add(new DataColumn("Contact2", typeof(string)));
103        dt.Columns.Add(new DataColumn("Contact3", typeof(string)));
104        using (sdr)
105        {
106            while (sdr.Read())
107            {
108                dt.Rows.Add( sdr.GetValue(0),sdr.GetValue(1),sdr.GetValue(2),sdr.GetValue(4),sdr.GetValue(5),
109                    sdr.GetValue(6),sdr.GetValue(7),sdr.GetValue(8),sdr.GetValue(9),sdr.GetValue(10) );
110            }
111            dataGridView1.DataSource = dt;
112        }
113        1 reference | 0 changes | 0 authors, 0 changes
114        private void button5_Click(object sender, EventArgs e)
115        {
116            loaddataintogrid("select * from Contacts");
117        }
118        1 reference | 0 changes | 0 authors, 0 changes
119        private void addcombodata()
120        {
121            SQLiteDataReader sdr = obj11.dgwdata("SELECT city,company FROM contacts GROUP BY city, company");
122            using (sdr)
123            {
124                while (sdr.Read())

```



```

120         using (sdr)
121         {
122             while (sdr.Read())
123             {
124                 comboBox3.Items.Add(sdr.GetValue(0));
125                 comboBox4.Items.Add(sdr.GetValue(1));
126             }
127         }
128     }
129     1 reference | 0 changes | 0 authors, 0 changes
130     private void button6_Click(object sender, EventArgs e)
131     {
132         loaddataintogrid("Select * from contacts where upper(name) like '%" + txtname.Text.ToUpper()
133         + "%' OR upper(name) like '%" + txtname.Text.ToUpper() + "' OR upper(name) like '%"
134         + txtname.Text.ToUpper() + "%' OR upper(name)='" + txtname.Text.ToUpper() + "%'");
135     }
136     1 reference | 0 changes | 0 authors, 0 changes
137     private void button7_Click(object sender, EventArgs e)
138     {
139         loaddataintogrid("Select * from contacts where contact1='" + txtcontact1.Text
140         + "' OR contact2='" + txtcontact2.Text + "' OR contact3='" + txtcontact3.Text + "'");
141     }
142     1 reference | 0 changes | 0 authors, 0 changes
143     private void button8_Click(object sender, EventArgs e)
144     {
145         loaddataintogrid("Select * from contacts where upper(email)='" + txtemail.Text.ToUpper() + "'");
146     }
147     1 reference | 0 changes | 0 authors, 0 changes
148     private void button2_Click(object sender, EventArgs e)
149     {
150         contentenable();
151         button9.Enabled = true;
152     }
153     2 references | 0 changes | 0 authors, 0 changes
154     private void contentdisable()
155     {
156         txtaddress.Enabled = false;
157         txtcity.Enabled = false;
158         txtcompany.Enabled = false;
159         txtdesignation.Enabled = false;
160         txtfather.Enabled = false;
161         gender.Enabled = false;
162         button1.Enabled = false;
163         button3.Enabled = false;
164         button4.Enabled = false;
165         button9.Enabled = false;
166     }
167     2 references | 0 changes | 0 authors, 0 changes

```

```

161     }
162     2 references | 0 changes | 0 authors, 0 changes
163     private void contentenable()
164     {
165         txtaddress.Enabled = true;
166         txtcity.Enabled = true;
167         txtcompany.Enabled = true;
168         txtdesignation.Enabled = true;
169         txtfather.Enabled = true;
170         gender.Enabled = true;
171         button1.Enabled = true;
172         button3.Enabled = true;
173         button4.Enabled = true;
174         txtcity.Text = "";
175         txtaddress.Text = "";
176         txtcompany.Text = "";
177         txtcontact1.Text = "";
178         txtcontact2.Text = "";
179         txtcontact3.Text = "";
180         txtdesignation.Text = "";
181         txtemail.Text = "";
182         txtemail.Text = "";
183         txtfather.Text = "";
184         txtname.Text = "";
185         gender.Text = "Gander";
186     }
187     1 reference | 0 changes | 0 authors, 0 changes
188     private void button3_Click(object sender, EventArgs e)
189     {
190         if (MessageBox.Show("Are u sure to delete this contact with name "
191         + txtname.Text, "Warning", MessageBoxButtons.YesNo) == DialogResult.Yes)
192         {
193             objj1.commandexecutor("delete from contacts where upper(name)='"
194             + txtname.Text.ToUpper() + "' OR contact1='" + txtcontact1.Text + "' OR contact2='"
195             + txtcontact2.Text + "' OR contact3='" + txtcontact3.Text + "'");
196             MessageBox.Show("Record is deleted");
197         }
198         else
199         {
200         }
201     }
202     1 reference | 0 changes | 0 authors, 0 changes
203     private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)

```

```

CMSBU.Contacts
dataGridView1_CellCon

{
    string keytxt = txtcontact1.Text.Trim();
    int rowcount = 0;
    if (button4.Text == "Edit" && txtcontact1.Text.Length != 0)
    {
        SQLiteDataReader sdr = obj11.dgwddata("SELECT * from contacts where contact1='"
        + txtcontact1.Text.Trim() + "' OR contact2 like '%" + txtcontact1.Text.Trim()
        + "%' OR contact3 like '%" + txtcontact1.Text.Trim() + "%'");
        using (sdr)
        {
            while (sdr.Read())
            {
                txtname.Text = sdr.GetValue(0).ToString();
                txtfather.Text = sdr.GetValue(1).ToString();
                gender.Text = sdr.GetValue(2).ToString();
                txtaddress.Text = sdr.GetValue(3).ToString();
                txtcity.Text = sdr.GetValue(4).ToString();
                txtcompany.Text = sdr.GetValue(5).ToString();
                txtdesignation.Text = sdr.GetValue(6).ToString();
                txtemail.Text = sdr.GetValue(7).ToString();
                txtcontact1.Text = sdr.GetValue(8).ToString();
                txtcontact2.Text = sdr.GetValue(9).ToString();
                txtcontact3.Text = sdr.GetValue(10).ToString();
                rowcount++;
            }
            if (rowcount < 1)
            {
                MessageBox.Show("No any record found of this contact \nPlease provide correct contact");
                button4.Text = "Edit";
                txtcontact1.Focus();
            }
        }
        button1.Enabled = false;
        button4.Text = "Update";
    }
    else if (button4.Text == "Update" && txtcontact1.Text.Length != 0)
    {
        obj11.commandexecutor("update contacts set name='" + txtname.Text.Trim()
        + "', fathername='" + txtfather.Text.Trim() + "', gander='" + gender.Text
        + "', address='" + txtaddress.Text.Trim() + "', city='" + txtcity.Text
        + "', company='" + txtcompany.Text.Trim() + "', designation='" + txtdesignation.Text.Trim()
        + "', email='" + txtemail.Text + "', contact1='" + txtcontact1.Text.Trim()
        + "', contact2='" + txtcontact2.Text.Trim() + "', contact3='" + txtcontact3.Text.Trim()
        + "' where contact1='" + keytxt + "'");
        loaddataintogrid("select * from Contacts");
        MessageBox.Show("Record is Updated");
    }
}

```

```

244         button1.Enabled = false;
245         button4.Text = "Update";
246     }
247     else if (button4.Text == "Update" && txtcontact1.Text.Length != 0)
248     {
249         obj11.commandexecutor("update contacts set name='" + txtname.Text.Trim()
250         + "', fathername='" + txtfather.Text.Trim() + "', gander='" + gender.Text
251         + "', address='" + txtaddress.Text.Trim() + "', city='" + txtcity.Text
252         + "', company='" + txtcompany.Text.Trim() + "', designation='" + txtdesignation.Text.Trim()
253         + "', email='" + txtemail.Text + "', contact1='" + txtcontact1.Text.Trim()
254         + "', contact2='" + txtcontact2.Text.Trim() + "', contact3='" + txtcontact3.Text.Trim()
255         + "' where contact1='" + keytxt + "'");
256         loaddataintogrid("select * from Contacts");
257         MessageBox.Show("Record is Updated");
258         button4.Text = "Edit";
259     }
260     else
261     {
262         MessageBox.Show("Please provide the primary contact to search and update");
263         txtcontact1.Focus();
264         button4.Text = "Edit";
265     }
266 }
267 1 reference | 0 changes | 0 authors, 0 changes
268 private void button9_Click(object sender, EventArgs e)
269 {
270     button4.Text = "Edit";
271     contentenable();
272     contentdisable();
273     button9.Enabled = false;
274 }
275 1 reference | 0 changes | 0 authors, 0 changes
276 private void button10_Click(object sender, EventArgs e)
277 {
278     User User = new User();
279     User.Show();
280 }

```

ContactTest.cs:

```
SQLiteDatabase.cs  DatabaseServerTests.cs*  UserTests.cs  User.cs  ContactsTests.cs*  Contacts.cs  User.cs [Design]  Program.cs
CMSBUTests
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using CMSBU;
using System.Windows.Forms;

namespace CMSBUTests
{
    [TestClass]
    public class ContactsTests
    {
        private Contacts contactsForm;

        [TestInitialize]
        public void Setup()
        {
            contactsForm = null;
            var thread = new System.Threading.Thread(() =>
            {
                contactsForm = new Contacts();
                contactsForm.Show();
                InitializeComboBoxes();
            });
            thread.SetApartmentState(System.Threading.ApartmentState.STA);
            thread.Start();
            thread.Join();
        }

        private void InitializeComboBoxes()
        {
            contactsForm.Controls.Add(new ComboBox { Name = "comboBox2", Text = "Gender" });
            contactsForm.Controls.Add(new ComboBox { Name = "comboBox3", Text = "City" });
            contactsForm.Controls.Add(new ComboBox { Name = "comboBox4", Text = "Company" });
        }

        [TestMethod]
        public void Sort_GenderOnlySelected_CreatesCorrectQuery()
        {
            contactsForm.Controls["comboBox2"].Text = "Gender"; // Set gender to Male
            contactsForm.Controls["comboBox3"].Text = "City"; // Set city to default value (no filtering)
            contactsForm.Controls["comboBox4"].Text = "Company"; // Set company to default value (no filtering)

            PrivateObject po = new PrivateObject(contactsForm);
            po.Invoke("sort");
            string expected = "select * from Contacts where gender='Male'"; // Incorrect query
            Assert.AreEqual(expected, contactsForm.GetSearchStatement());
        }

        [TestMethod]
        public void Sort_NoFiltersSelected_ReturnsDefaultQuery()
        {
            contactsForm.Controls["comboBox2"].Text = "Gender";
            contactsForm.Controls["comboBox3"].Text = "City";
        }
    }
}
```

```
public void Sort_GenderOnlySelected_CreatesCorrectQuery()
{
    contactsForm.Controls["comboBox2"].Text = "Gender"; // Set gender to Male
    contactsForm.Controls["comboBox3"].Text = "City"; // Set city to default value (no filtering)
    contactsForm.Controls["comboBox4"].Text = "Company"; // Set company to default value (no filtering)

    PrivateObject po = new PrivateObject(contactsForm);
    po.Invoke("sort");
    string expected = "select * from Contacts where gender='Male'"; // Incorrect query
    Assert.AreEqual(expected, contactsForm.GetSearchStatement());
}

[TestMethod]
public void Sort_NoFiltersSelected_ReturnsDefaultQuery()
{
    contactsForm.Controls["comboBox2"].Text = "Gender";
    contactsForm.Controls["comboBox3"].Text = "City";
    contactsForm.Controls["comboBox4"].Text = "Company";

    PrivateObject po = new PrivateObject(contactsForm);
    po.Invoke("sort");

    string expected = "select * from Contacts";
    Assert.AreEqual(expected, contactsForm.GetSearchStatement());
}
}
```

ContactDesign.cs:

The screenshot shows a Windows application window titled 'Contacts'. The window is divided into two main panes. The left pane contains a form for managing contacts with the following fields: Name, Father Name, Gender (dropdown), Address, City, Company, Designation, Email, Contact No. 1, Contact No. 2, and Contact No. 3. There are 'Search' buttons next to the Name and Email fields. The right pane is a large empty area, likely for displaying a list of contacts. At the bottom of the window, there are several buttons: 'User Management', 'Save', 'Add New', 'Delete', 'Abort Edit', 'Edit', and 'Show Data'. The window also has a title bar with standard Windows controls and a menu bar with options like 'File', 'Edit', 'View', and 'Help'.

User.cs:

```
1  using Finisar.SQLite;
2  namespace CMSBU
3  {
4      4 references | 0 changes | 0 authors, 0 changes
5      public partial class User : Form
6      {
7          1 reference | 0 changes | 0 authors, 0 changes
8          public User()
9          {
10             InitializeComponent();
11         }
12         DatabaSeServer obj11 = new DatabaSeServer();
13         1 reference | 0 changes | 0 authors, 0 changes
14         private void button1_Click(object sender, EventArgs e)
15         {
16             int rowcount = 0;
17             SQLiteDataReader sdr = obj11.dgudata("SELECT * from user where userid='"
18             + textBox1.Text.Trim() + "' OR username='" + textBox2.Text.Trim() + "'");
19             using (sdr)
20             {
21                 while (sdr.Read())
22                 {
23                     textBox1.Text = sdr.GetValue(0).ToString();
24                     textBox2.Text = sdr.GetValue(1).ToString();
25                     rowcount++;
26                 }
27             }
28             button7.Enabled = true;
29             if (rowcount <= 0)
30             {
31                 MessageBox.Show("No User available on provided identifications.");
32                 button7.Enabled = false;
33             }
34             sdr.Close();
35         }
36         1 reference | 0 changes | 0 authors, 0 changes
37         private void button2_Click(object sender, EventArgs e)
38         {
39             if (MessageBox.Show("Are u sure to delete this User ID with name "
40             + textBox1.Text, "Warning", MessageBoxButtons.YesNo) == DialogResult.Yes)
41             {
42                 obj11.commandexecutor("delete from user where upper(userid)='" + textBox1.Text.ToUpper() +
43                 "' AND upper(username)='" + textBox2.Text.ToUpper() + "' AND userid='1'");
44                 loaddataintogrid();
45                 MessageBox.Show("User id Deleted Successfully,\n This function can not delete admin user ID");
46             }
47             else { }
48         }
49     }
50 }
```

```

49     MessageBox.Show("User id Deleted Successfully,\n This function can not delete admin user ID");
50 }
51 else { }
52
53 private void button3_Click(object sender, EventArgs e)
54 {
55     textBox1.Text = "";
56     textBox2.Text = "";
57     textBox3.Text = "";
58     button4.Enabled = true;
59     button5.Enabled = true;
60 }
61
62 private void USer_Load(object sender, EventArgs e)
63 {
64     button4.Enabled = false;
65     button5.Enabled = false;
66     loaddataintogrid();
67 }
68
69 private void button5_Click(object sender, EventArgs e)
70 {
71     textBox1.Text = "";
72     textBox2.Text = "";
73     textBox3.Text = "";
74     button4.Enabled = false;
75     button5.Enabled = false;
76 }
77
78 private void button4_Click(object sender, EventArgs e)
79 {
80     obj11.adduser(textBox2.Text, textBox3.Text);
81     textBox1.Text = "";
82     textBox2.Text = "";
83     textBox3.Text = "";
84     button4.Enabled = false;
85     button5.Enabled = false;
86     loaddataintogrid();
87 }
88
89 private void button6_Click(object sender, EventArgs e)
90 {
91     if (textBox7.Text == textBox8.Text)
92     {
93         obj11.commandexecutor("update user set username='"
94             + textBox5.Text.Trim() + "', password='" + textBox8.Text.Trim() + "'");
95         MessageBox.Show("Password Chaneeed");
96     }
97 }

```

```

87     if (textBox7.Text == textBox8.Text)
88     {
89         obj11.commandexecutor("update user set username='"
90             + textBox5.Text.Trim() + "', password='" + textBox8.Text.Trim() + "'");
91         MessageBox.Show("Password Changed");
92         textBox6.Text = "";
93         textBox5.Text = "";
94         textBox4.Text = "";
95         textBox7.Text = "";
96         textBox8.Text = "";
97         textBox4.Enabled = false;
98         textBox7.Enabled = false;
99         textBox8.Enabled = false;
100         button6.Enabled = false;
101         button7.Enabled = false;
102     }
103     else
104     {
105         MessageBox.Show("Passwords are not matching");
106     }
107 }
108
109 private void button7_Click(object sender, EventArgs e)
110 {
111     textBox6.Text = textBox1.Text;
112     textBox5.Text = textBox2.Text;
113     textBox4.Enabled = true;
114     textBox7.Enabled = true;
115     textBox8.Enabled = true;
116     button6.Enabled = true;
117 }
118
119 private void button8_Click(object sender, EventArgs e)
120 {
121     MessageBox.Show("Pass 1=" + textBox7.Text + " \n Pass 2=" + textBox8.Text);
122 }
123
124 private void loaddataintogrid()
125 {
126     SQLiteDataReader sdr = obj11.dgwddata("Select * from user");
127     DataTable dt = new DataTable();
128     dt.Columns.Add(new DataColumn("User ID", typeof(string)));
129     dt.Columns.Add(new DataColumn("User Name", typeof(string)));
130     using (sdr)
131     {
132         while (sdr.Read())
133         {
134             // Add data to the DataTable
135         }
136     }
137 }

```

UserTest.cs:

```
SQLiteDataReader.cs  DatabaSeServerTests.cs  UserTests.cs  User.cs  ContactsTests.cs  Con
CMSBUTests
1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3  using CMSBU;
4  using System.Windows.Forms;
5
6  namespace CMSBUTests
7  {
8      [TestClass]
9      0 references | 0 changes | 0 authors, 0 changes
10     public class UserTests
11     {
12         private User userForm;
13
14         [TestInitialize]
15         0 references | 0 changes | 0 authors, 0 changes
16         public void Setup()
17         {
18             userForm = null;
19             var thread = new System.Threading.Thread(() =>
20             {
21                 userForm = new User();
22                 userForm.Show();
23                 InitializeControls();
24             });
25             thread.SetApartmentState(System.Threading.ApartmentState.STA);
26             thread.Start();
27             thread.Join();
28
29             1 reference | 0 changes | 0 authors, 0 changes
30             private void InitializeControls()
31             {
32                 userForm.Controls.Add(new TextBox { Name = "textBox1" });
33                 userForm.Controls.Add(new TextBox { Name = "textBox2" });
34                 userForm.Controls.Add(new TextBox { Name = "textBox3" });
35                 userForm.Controls.Add(new Button { Name = "button1" });
36                 userForm.Controls.Add(new Button { Name = "button2" });
37             }
38
39             [TestMethod]
40             0 | 0 references | 0 changes | 0 authors, 0 changes
41             public void Delete_User_PromptsConfirmation()
42             {
43                 userForm.Controls["textBox1"].Text = "User123";
44                 userForm.Controls["textBox2"].Text = "JohnDoe";
45
46                 var button = (Button)userForm.Controls["button2"];
47                 button.PerformClick();
48
49                 // Simulate Yes to confirm deletion
50                 Assert.AreEqual("", userForm.Controls["textBox1"].Text);
51                 Assert.AreEqual("", userForm.Controls["textBox2"].Text);
52             }
53         }
54     }
55 }
```

```
44         button.PerformClick();
45
46         // Simulate Yes to confirm deletion
47         Assert.AreEqual("", userForm.Controls["textBox1"].Text);
48         Assert.AreEqual("", userForm.Controls["textBox2"].Text);
49     }
50
51     [TestMethod]
52     0 | 0 references | 0 changes | 0 authors, 0 changes
53     public void Add_NewUser_AddsSuccessfully()
54     {
55         userForm.Controls["textBox2"].Text = "NewUser";
56         userForm.Controls["textBox3"].Text = "Password123";
57
58         var addButton = (Button)userForm.Controls["button4"];
59         addButton.PerformClick();
60
61         Assert.AreEqual("", userForm.Controls["textBox2"].Text);
62         Assert.AreEqual("", userForm.Controls["textBox3"].Text);
63     }
64 }
65 }
```

UserDesign.cs:

The screenshot shows a Windows application window titled "User" with a standard Windows XP-style title bar (minimize, maximize, close buttons). The application has a menu bar with the following items: "User.cs", "Contacts.cs*", "User.cs [Design]" (highlighted in yellow), "Contacts.cs [Design]*", "Program.cs", and "DatabaSeServer.cs".

The main content area of the "User" window contains a form with the following elements:

- On the left side, there are three text input fields labeled "User ID", "User Name", and "Password". To the right of these fields are three buttons: "Search", "Delete", and "New".
- Below the "Password" field, there are three buttons: "Abort", "Generate", and "New".
- Below the "Generate" button, there is a small button with a downward-pointing arrow (v).
- Below the "v" button, there are four text input fields labeled "User ID", "User Name", "Current Password", and "New Password". To the right of the "New Password" field is a small button labeled "A".
- Below the "New Password" field, there is a text input field labeled "Retype New Password".
- Below the "Retype New Password" field, there is a button labeled "Change Password".
- On the right side of the form, there is a large, empty rectangular area, possibly a placeholder for a list or a detailed view.

Appendix A: Bibliography / References

Appendix B: Index

GIT HUB REPOSITORY LINK:

<https://github.com/Hashir2022/Contact-Management-System-CMS-Software-Construction-Lab-Project>

Hashir2022 / Contact-Management-System-CMS-Software-Construction-Lab-Project

Type [] to search

code issues pull requests actions projects 1 wiki security insights settings

Contact-Management-System-CMS-Software-Construction-Lab-Pr... Public Pin Unwatch 1

main 1 Branch 0 Tags

Go to file t Add file <> Code

Hashir2022 Add files via upload e3801fd · 2 days ago 3 Commits

.gitignore	Initial commit	2 days ago
Contact Management System (CMS) Software ...	Project Report	2 days ago
Contact Managment System Project Proposal...	Add files via upload	2 days ago
Contact Managment System Software Constru...	Add files via upload	2 days ago
README.md	Initial commit	2 days ago

README

Contact-Management-System-CMS-Software-Construction-Lab-Project