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

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

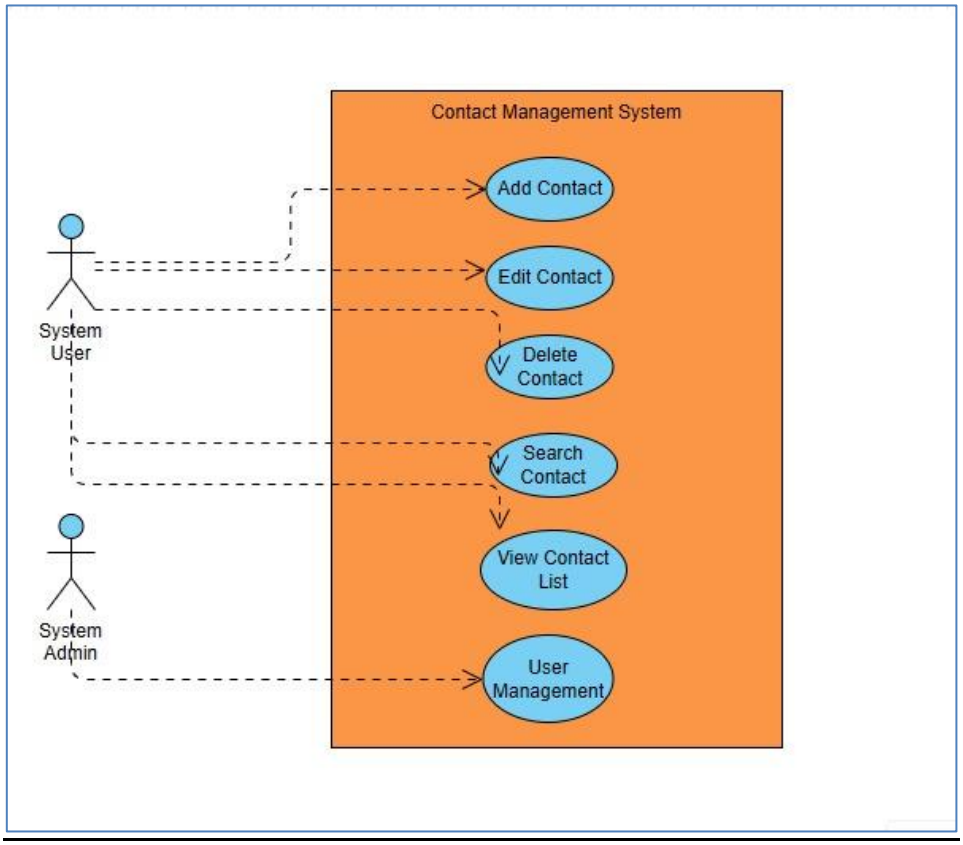
CHAPTER # 3

Design & Implementation

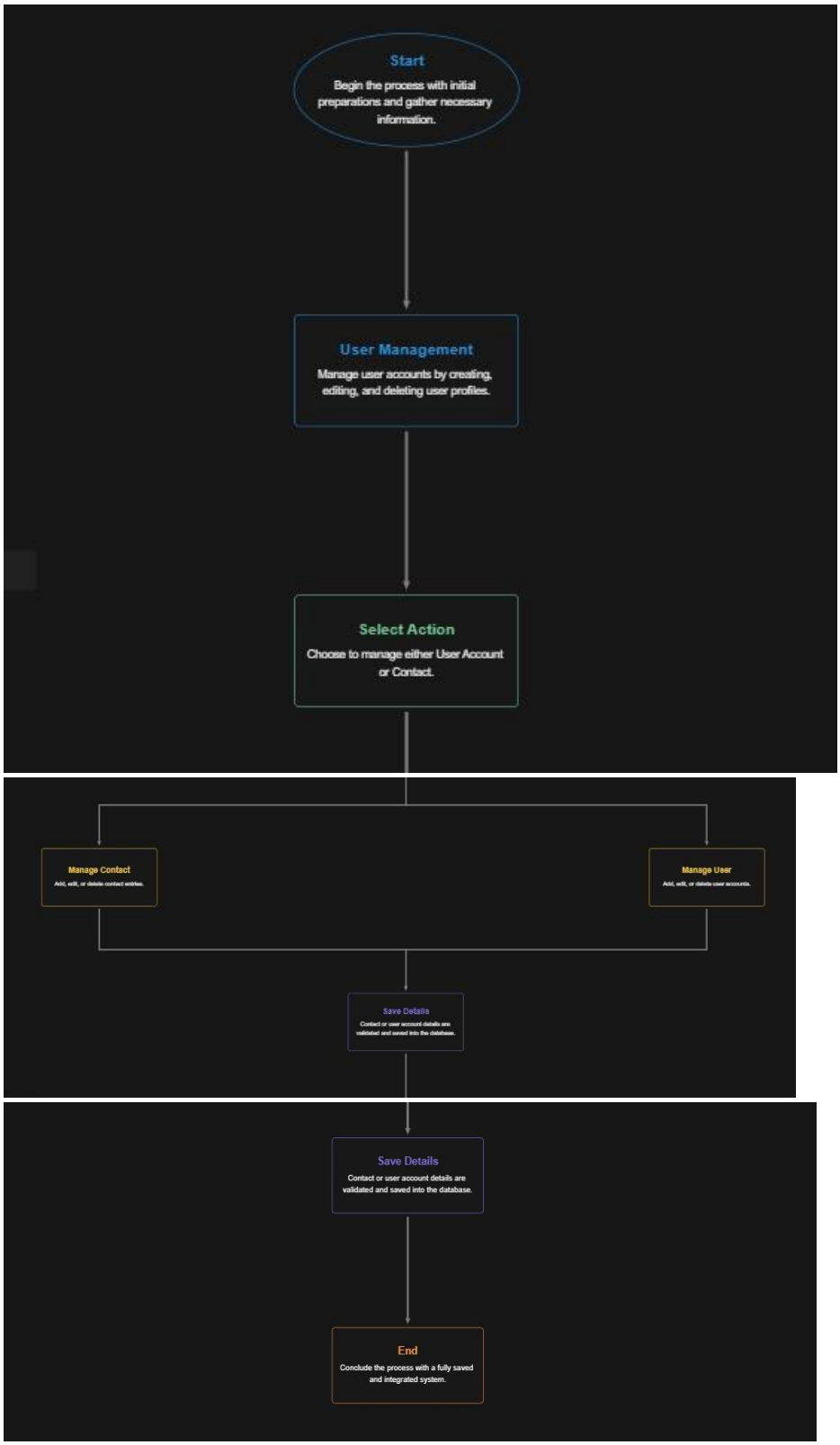
3 DESIGN & IMPLEMENTATION

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:

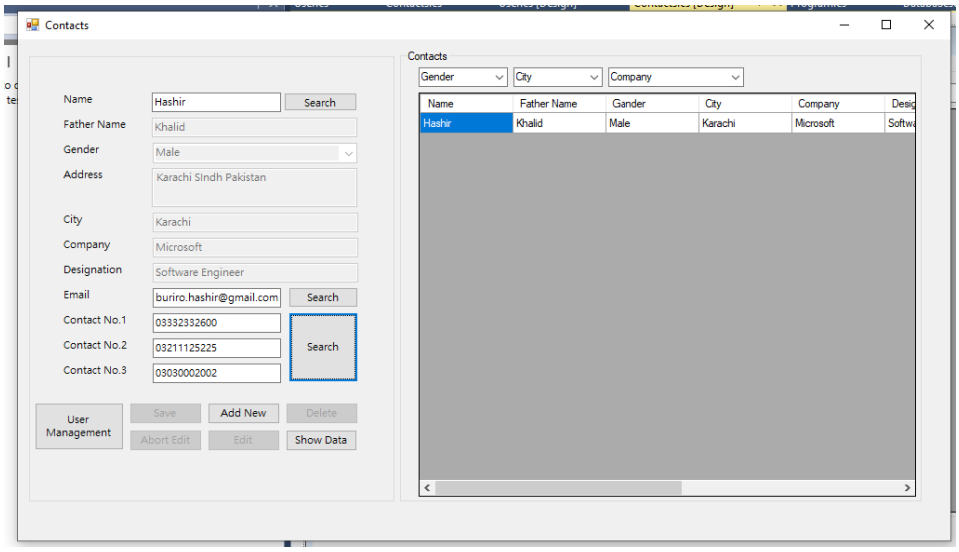
- **Functional Testing:**

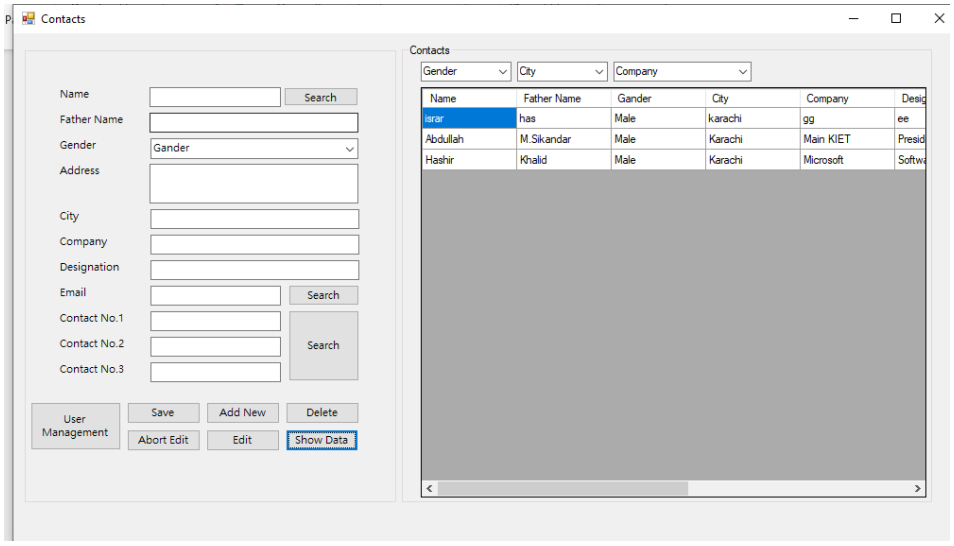
Test Case	Expected Result	Actual Result
Add a new contact	Contact saved successfully	Pass
Search contact by name	Relevant contact displayed	Pass
Edit contact details	Updated information saved	Pass

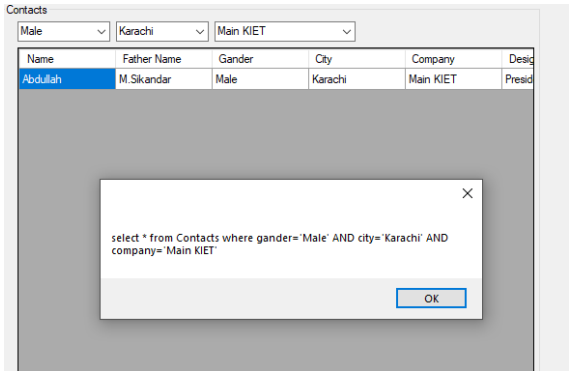
- **Boundary Testing:**

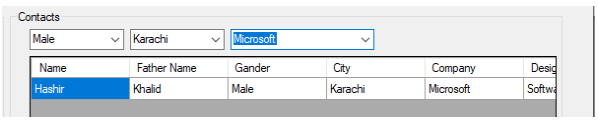
Test Case	Input	Expected Result	Actual Result
Add duplicate contact	Same Contact No.	Error: Duplicate entry	Pass
Leave required fields blank	Blank Name/Email	Error: Fields cannot be empty	Pass

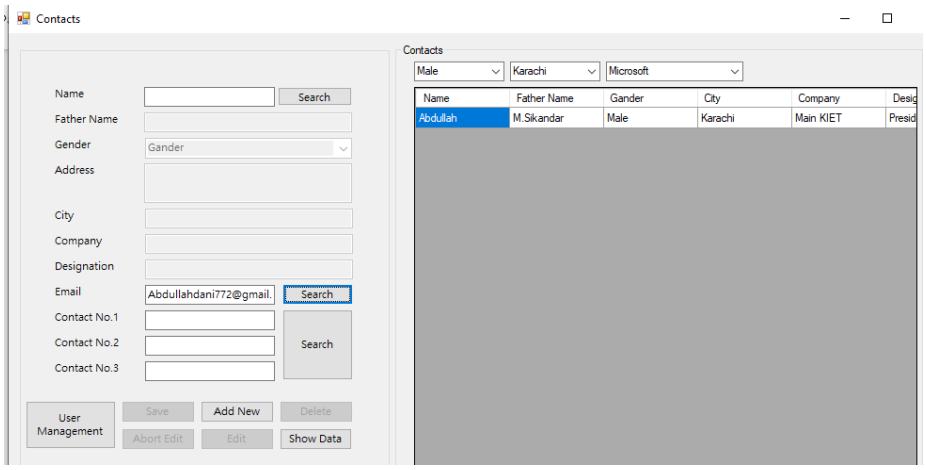
4.2 Results

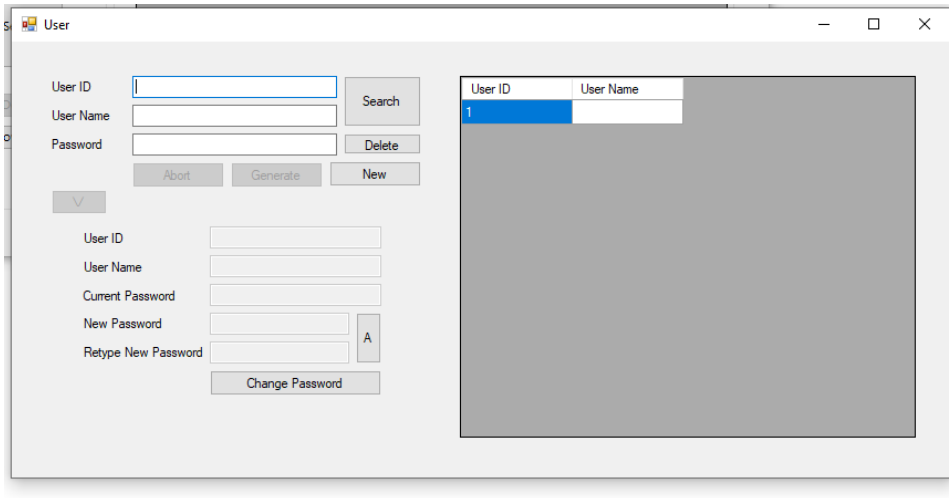

Contacts
Name
Hashir
Search
Father Name
Khalid
Gender
Male
Address
Karachi SIndh Pakistan
City
Karachi
Company
Microsoft
Designation
Software Engineer
Email
buriro.hashir@gmail.com
Search
Contact No.1
03332332600
Contact No.2
03211125225
Contact No.3
03030002002
Search
User Management
Save
Add New
Delete
Abort Edit
Edit
Show Data
Contacts
Gender
City
Company
Name
Father Name
Gander
City
Company
Hashir
Khalid
Male
Karachi
Microsoft

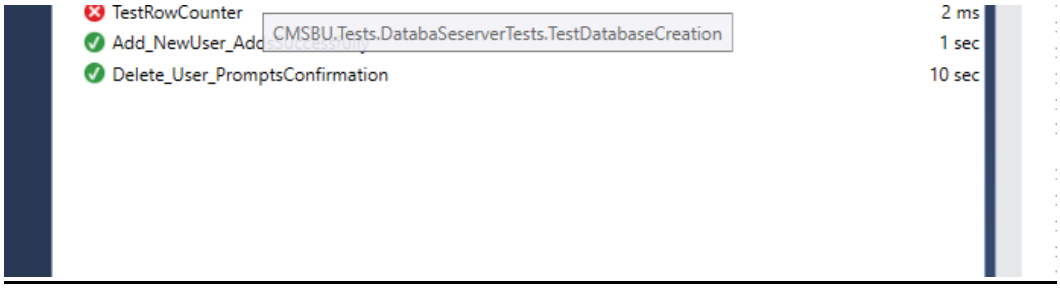

Contacts
Name
Search
Father Name
Gender
Gander
Address
City
Company
Designation
Email
Search
Contact No.1
Contact No.2
Search
Contact No.3
User Management
Save
Add New
Delete
Abort Edit
Edit
Show Data
Contacts
Gender
City
Company
Name
Father Name
Gander
City
Company
israr
has
Male
karachi
gg
ee
Abdullah
M.Sikandar
Male
Karachi
Main KIET
Hashir
Khalid
Male
Karachi
Microsoft


Contacts
Male
Karachi
Main KIET
Name
Father Name
Gander
City
Company
Abdullah
M.Sikandar
Male
Karachi
Main KIET
select * from Contacts where gander='Male' AND city='Karachi' AND company='Main KIET'
OK


Contacts
Male
Karachi
Microsoft
Name
Father Name
Gander
City
Company
Hashir
Khalid
Male
Karachi
Microsoft


Contacts
Name
Search
Father Name
Gender
Gander
Address
City
Company
Designation
Email
Abdullahdani772@gmail.
Search
Contact No.1
Contact No.2
Contact No.3
Search
User Management
Save
Add New
Delete
Abort Edit
Edit
Show Data
Contacts
Male
Karachi
Microsoft
Name
Father Name
Gander
City
Company
Abdullah
M.Sikandar
Male
Karachi
Main KIET


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
Add_NewUser_Add
CMSBU.Tests.DatabaseServerTests.TestDatabaseCreation
Delete_User_PromptsConfirmation
2 ms
1 sec
10 sec

CHAPTER # 5

Future Enhancements

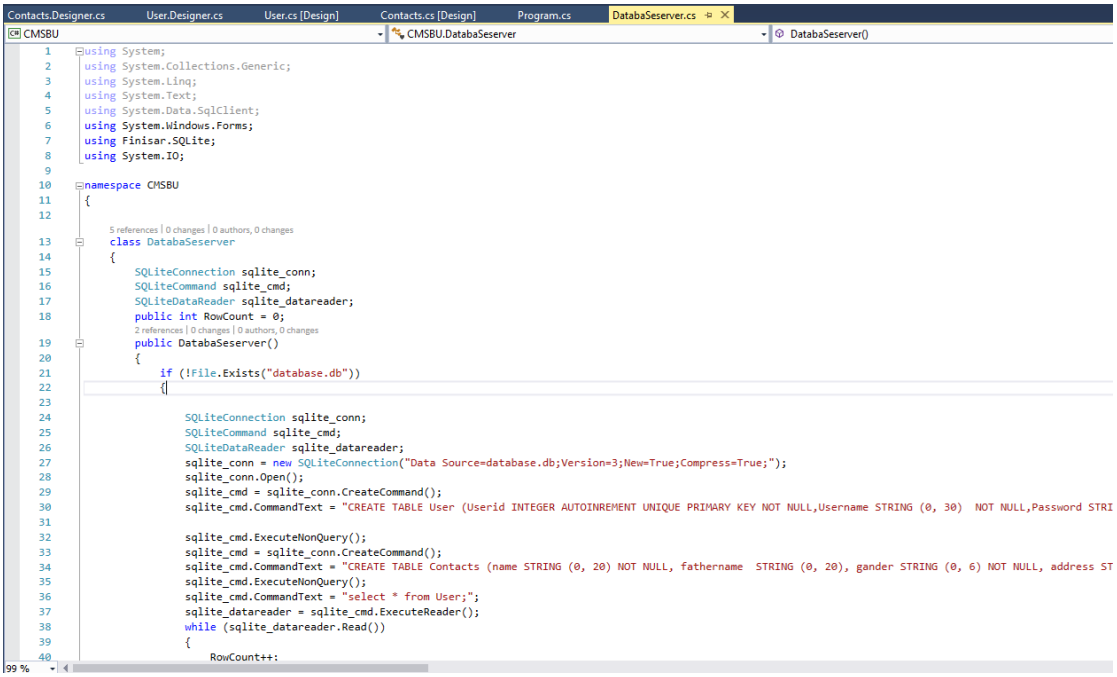
5 Future Enhancements

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:



```
                sqlite_cmd.ExecuteNonQuery();
                sqlite_cmd.CommandText = "select * from User;";
                sqlite_datareader = sqlite_cmd.ExecuteReader();
                while (sqlite_datareader.Read())
                {
                    RowCount++;

                }
                MessageBox.Show("Database Successfully Generated" + RowCount);
                sqlite_conn.Close();

            }

        }
        1 reference | 0 changes | 0 authors, 0 changes
        public void adduser(string username, string password)
        {
            sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
            sqlite_conn.Open();
            sqlite_cmd = sqlite_conn.CreateCommand();
            sqlite_cmd.CommandText = "SELECT * FROM User";

            sqlite_datareader = sqlite_cmd.ExecuteReader();
            while (sqlite_datareader.Read())
            {
                RowCount = int.Parse(sqlite_datareader.GetValue(0).ToString());

            }
            sqlite_datareader.Close();
            if (RowCount <= 0)
            {
            try
            {
                string cmd = "INSERT into User VALUES(" +(RowCount + 1) + ",'" + username + "','" + password + "');";
                sqlite_cmd.CommandText = cmd;

                sqlite_cmd.ExecuteNonQuery();
                sqlite_conn.Close();
                MessageBox.Show("Your account is generated");

            }
```

```
                }

        }

        0 references | 0 changes | 0 authors, 0 changes
        public int rowcounter()
        {
            sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
            sqlite_conn.Open();
            sqlite_cmd = sqlite_conn.CreateCommand();
            sqlite_cmd.CommandText = "SELECT * FROM User";

            sqlite_datareader = sqlite_cmd.ExecuteReader();
            while (sqlite_datareader.Read())
            {
                RowCount++;

            }
            sqlite_datareader.Close();
            sqlite_conn.Close();
            return RowCount;
        }
        0 references | 0 changes | 0 authors, 0 changes
        public int cridentals(string username, string password)
        {
            sqlite_conn = new SQLiteConnection("Data Source=database.db;Version=3;New=False;Compress=False;");
            sqlite_conn.Open();
            sqlite_cmd = sqlite_conn.CreateCommand();
            sqlite_cmd.CommandText = "SELECT * FROM User where Username=='" + username + "' and password=='" + password + "';";
            sqlite_datareader = sqlite_cmd.ExecuteReader();
            while (sqlite_datareader.Read())
            {
                RowCount++;

            }
            sqlite_datareader.Close();
            sqlite_conn.Close();
            return RowCount;

        }
        5 references | 0 changes | 0 authors, 0 changes
        public SQLiteDataReader dewdata(string statementt)
```

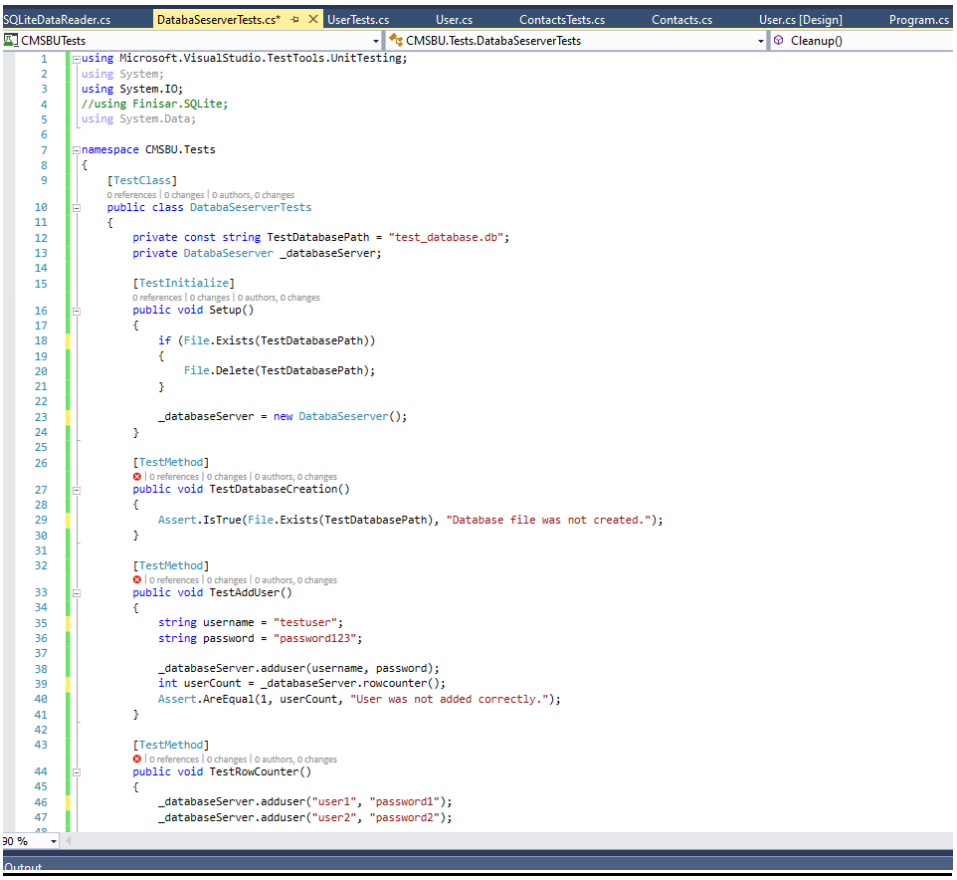
Output

```
110             sqlite_conn.Close();
111             return RowCount;
112             
113         }
        5 references | 0 changes | 0 authors, 0 changes
114         public SQLiteDataReader dgwdata(string statementt)
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 = statementt;
120             sqlite_datareader = sqlite_cmd.ExecuteReader();
121             return sqlite_datareader;
122 
123         }
124 
        1 reference | 0 changes | 0 authors, 0 changes
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         }
        4 references | 0 changes | 0 authors, 0 changes
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();
```

99 %

```
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 }
```

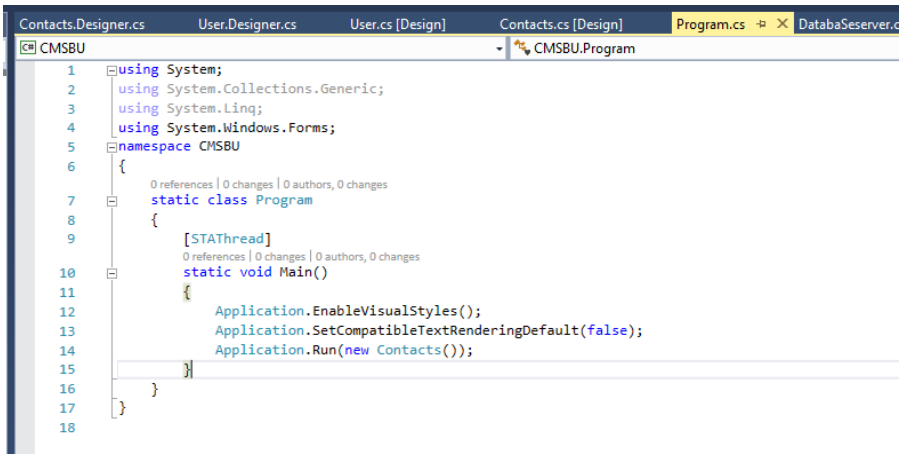
TEST:




```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.IO;
//using Finisar.SQLite;
using System.Data;

namespace CMSBU.Tests
{
    [TestClass]
    0 references | 0 changes | 0 authors, 0 changes
    public class DatabaseServerTests
    {
        private const string TestDatabasePath = "test_database.db";
        private DatabaseServer _databaseServer;

        [TestInitialize]
        0 references | 0 changes | 0 authors, 0 changes
        public void Setup()
        {
            if (File.Exists(TestDatabasePath))
            {
                File.Delete(TestDatabasePath);
            }

            _databaseServer = new DatabaseServer();
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestDatabaseCreation()
        {
            Assert.IsTrue(File.Exists(TestDatabasePath), "Database file was not created.");
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestAddUser()
        {
            string username = "testuser";
            string password = "password123";

            _databaseServer.adduser(username, password);
            int userCount = _databaseServer.rowcounter();
            Assert.AreEqual(1, userCount, "User was not added correctly.");
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestRowCounter()
        {
            _databaseServer.adduser("user1", "password1");
            _databaseServer.adduser("user2", "password2");
```

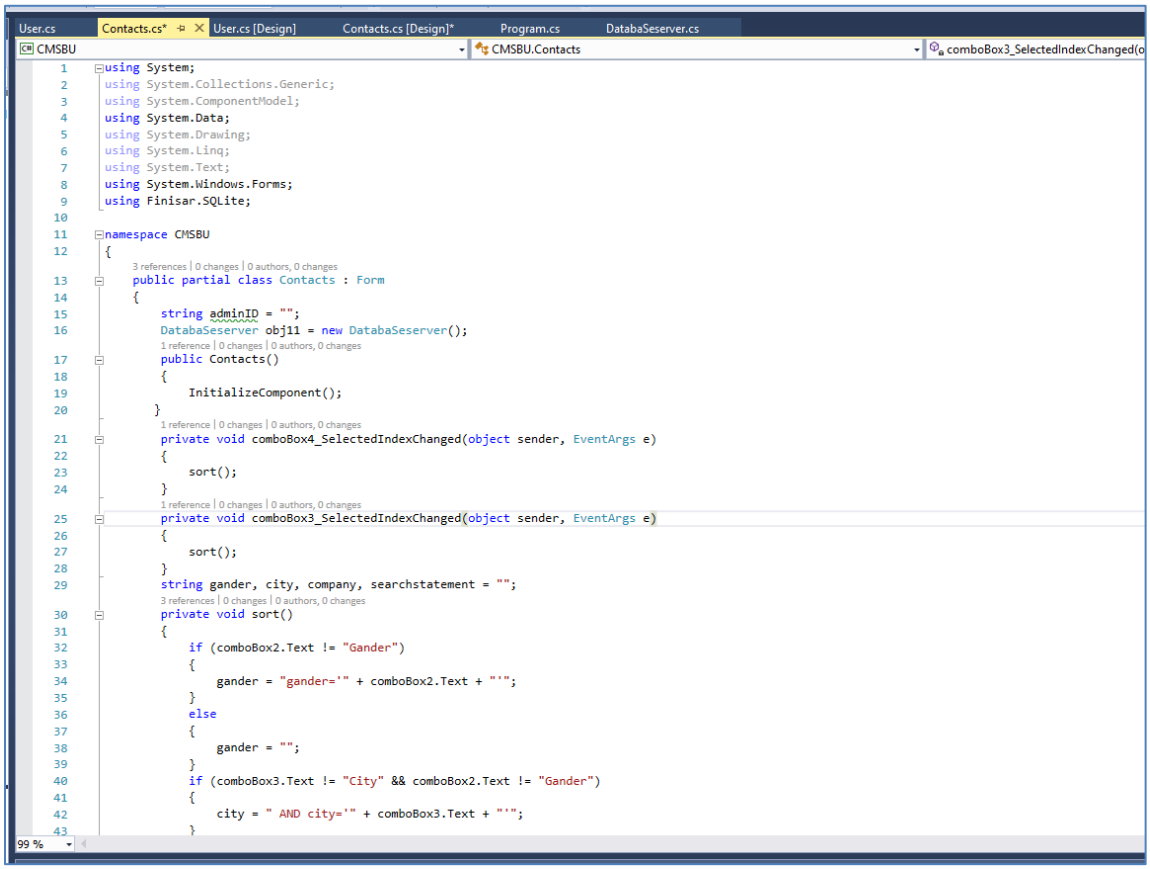
```
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestRowCounter()
        {
            _databaseServer.adduser("user1", "password1");
            _databaseServer.adduser("user2", "password2");

            int rowCount = _databaseServer.rowcounter();
            Assert.AreEqual(2, rowCount, "Row counter is not working correctly.");
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void TestCredentialValidation()
        {
            _databaseServer.adduser("testuser", "password123");

            int validCredentials = _databaseServer.cridentals("testuser", "password123");
            Assert.AreEqual(1, validCredentials, "Credentials validation failed.");

            int invalidCredentials = _databaseServer.cridentals("wronguser", "wrongpassword");
            Assert.AreEqual(0, invalidCredentials, "Invalid credentials passed validation.");
        }



        [TestCleanup]
        0 references | 0 changes | 0 authors, 0 changes
        public void Cleanup()
        {
            if (File.Exists(TestDatabasePath))
            {
                File.Delete(TestDatabasePath);
            }
        }
    }
}
```

Program.cs:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
namespace CMSBU
{
    0 references | 0 changes | 0 authors, 0 changes
    static class Program
    {
        [STAThread]
        0 references | 0 changes | 0 authors, 0 changes
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Contacts());
        }
    }
}
```

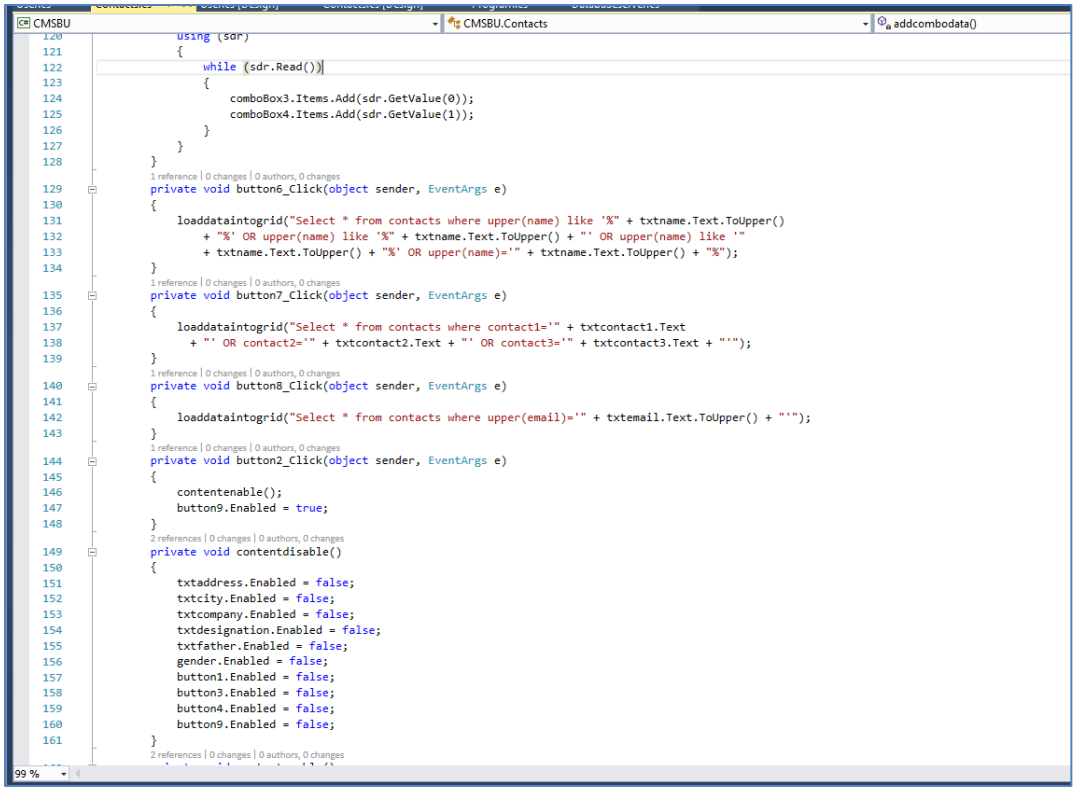
Contacts.cs:



```
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 Finisar.SQLite;

namespace CMSBU
{
    3 references | 0 changes | 0 authors, 0 changes
    public partial class Contacts : Form
    {
        string adminID = "";
        DatabaSeserver obj11 = new DatabaSeserver();
        1 reference | 0 changes | 0 authors, 0 changes
        public Contacts()
        {
            InitializeComponent();
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
        {
            sort();
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            sort();
        }
        string gander, city, company, searchstatement = "";
        3 references | 0 changes | 0 authors, 0 changes
        private void sort()
        {
            if (comboBox2.Text != "Gander")
            {
                gander = "gander='" + comboBox2.Text + "'";
            }
            else
            {
                gander = "";
            }
            if (comboBox3.Text != "City" && comboBox2.Text != "Gander")
            {
                city = " AND city='" + comboBox3.Text + "'";
            }
```

```
            if (comboBox3.Text != "City" && comboBox2.Text != "Gander")
            {
                city = " AND city='" + comboBox3.Text + "'";
            }
            else if (comboBox3.Text != "City" && comboBox2.Text == "Gander")
            {
                city = "city='" + comboBox3.Text + "'";
            }
            else
            {
                city = "";
            }
            if (comboBox4.Text != "Company" && (comboBox2.Text != "Gander" || comboBox3.Text != "City"))
            {
                company = " AND company='" + comboBox4.Text + "'";
            }
            else if (comboBox4.Text != "Company" && (comboBox2.Text == "Gander" && comboBox3.Text == "City"))
            {
                company = "company='" + comboBox4.Text + "'";
            }
            else
            {
                company = "";
            }
            if (gander != "" || city != "" || company != "")
            {
                searchstatement = "select * from Contacts where " + gander + city + company;
            }
            else
            {
                searchstatement = "select * from Contacts";
            }
            MessageBox.Show(searchstatement);
            loaddataintogrid(searchstatement);
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            sort();
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button1_Click(object sender, EventArgs e)
        {
            obj11.savecontacts("INSERT INTO contacts VALUES('" + txtname.Text + "','"
                + txtfather.Text + "','" + gender.Text + "','" + txtaddress.Text + "','" + txtcity.Text
                + "','" + txtcompany.Text + "','" + txtdesignation.Text + "','" + txtemail.Text + "','"
                + txtcontact1.Text + "','" + txtcontact2.Text + "','" + txtcontact3.Text + "');");
```

```
        {
            obj11.savecontacts("INSERT INTO contacts VALUES('" + txtname.Text + "','"
                + txtfather.Text + "','" + gender.Text + "','" + txtaddress.Text + "','" + txtcity.Text
                + "','" + txtcompany.Text + "','" + txtdesignation.Text + "','" + txtemail.Text + "','"
                + txtcontact1.Text + "','" + txtcontact2.Text + "','" + txtcontact3.Text + "');");
            loaddataintogrid("Select * from Contacts");
            addcombodata();
            contentdisable();
        }
        7 references | 0 changes | 0 authors, 0 changes
        private void loaddataintogrid(string statementt)
        {
            SQLiteDataReader sdr = obj11.dgwdata(statementt);
            DataTable dt = new DataTable();
            dt.Columns.Add(new DataColumn("Name", typeof(string)));
            dt.Columns.Add(new DataColumn("Father Name", typeof(string)));
            dt.Columns.Add(new DataColumn("Gander", typeof(string)));
            dt.Columns.Add(new DataColumn("City", typeof(string)));
            dt.Columns.Add(new DataColumn("Company", typeof(string)));
            dt.Columns.Add(new DataColumn("Designation", typeof(string)));
            dt.Columns.Add(new DataColumn("Email", typeof(string)));
            dt.Columns.Add(new DataColumn("Contact1", typeof(string)));
            dt.Columns.Add(new DataColumn("Contact2", typeof(string)));
            dt.Columns.Add(new DataColumn("Contact3", typeof(string)));
            using (sdr)
            {
                while (sdr.Read())
                {
                    dt.Rows.Add( sdr.GetValue(0),sdr.GetValue(1),sdr.GetValue(2),sdr.GetValue(4),sdr.GetValue(5),
                      sdr.GetValue(6),sdr.GetValue(7),sdr.GetValue(8),sdr.GetValue(9),sdr.GetValue(10) );
                }
            }
            dataGridView1.DataSource = dt;
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button5_Click(object sender, EventArgs e)
        {
            loaddataintogrid("select * from Contacts");
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void addcombodata()
        {
            SQLiteDataReader sdr = obj11.dgwdata("SELECT city,company FROM contacts GROUP BY city, company");
            using (sdr)
            {
                while (sdr.Read())
                {
```



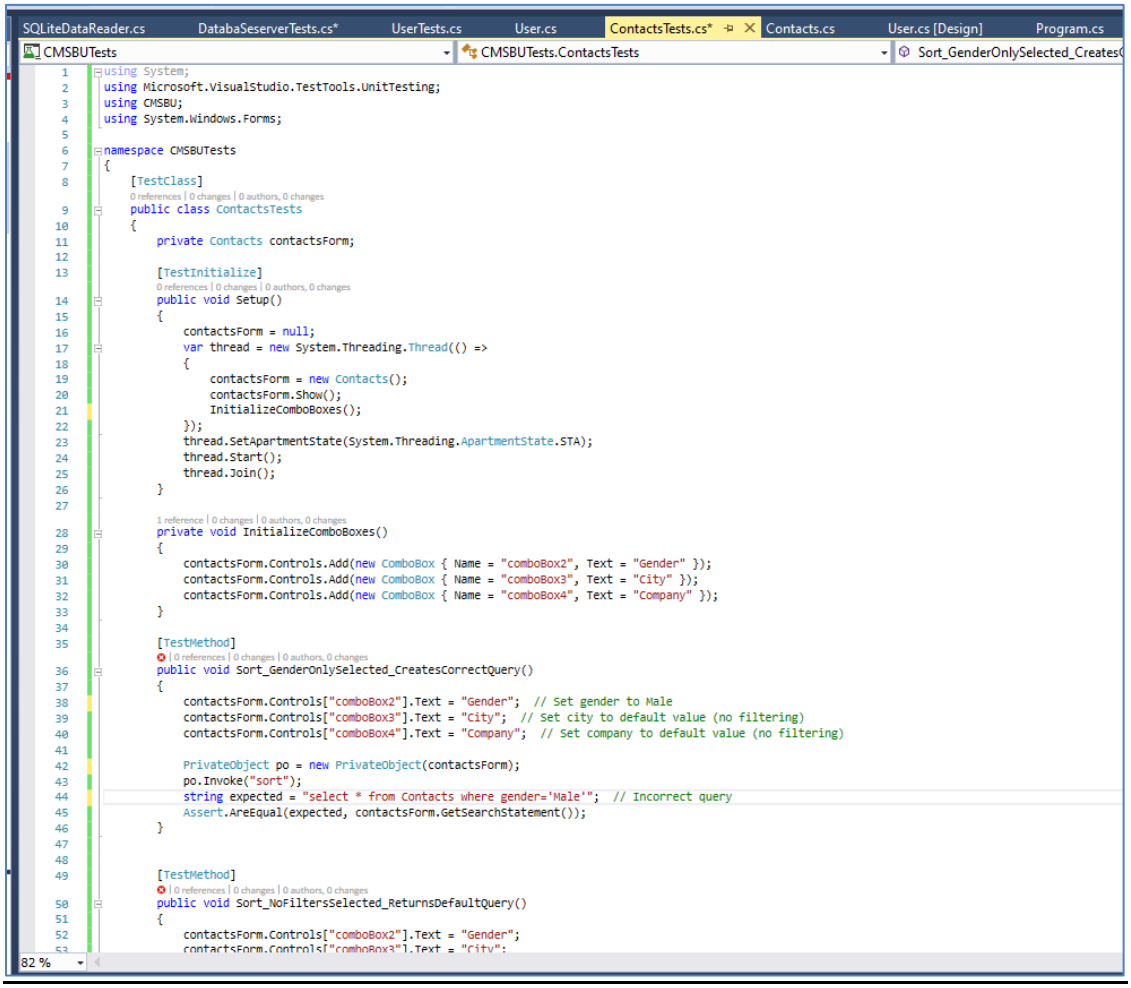
```
CMSBU                                   CMSBU.Contacts                          addcombodata()
            using (sdr)
            {
                while (sdr.Read())
                {
                    comboBox3.Items.Add(sdr.GetValue(0));
                    comboBox4.Items.Add(sdr.GetValue(1));
                }
            }
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button6_Click(object sender, EventArgs e)
        {
            loaddataintogrid("Select * from contacts where upper(name) like '%" + txtname.Text.ToUpper()
                + "%' OR upper(name) like '%" + txtname.Text.ToUpper() + "' OR upper(name) like '"
                + txtname.Text.ToUpper() + "%' OR upper(name)='" + txtname.Text.ToUpper() + "%");
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button7_Click(object sender, EventArgs e)
        {
            loaddataintogrid("Select * from contacts where contact1='" + txtcontact1.Text
              + "' OR contact2='" + txtcontact2.Text + "' OR contact3='" + txtcontact3.Text + "'");
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button8_Click(object sender, EventArgs e)
        {
            loaddataintogrid("Select * from contacts where upper(email)='" + txtemail.Text.ToUpper() + "'");
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button2_Click(object sender, EventArgs e)
        {
            contentenable();
            button9.Enabled = true;
        }
        2 references | 0 changes | 0 authors, 0 changes
        private void contentdisable()
        {
            txtaddress.Enabled = false;
            txtcity.Enabled = false;
            txtcompany.Enabled = false;
            txtdesignation.Enabled = false;
            txtfather.Enabled = false;
            gender.Enabled = false;
            button1.Enabled = false;
            button3.Enabled = false;
            button4.Enabled = false;
            button9.Enabled = false;
        }
        2 references | 0 changes | 0 authors, 0 changes
99 %
```

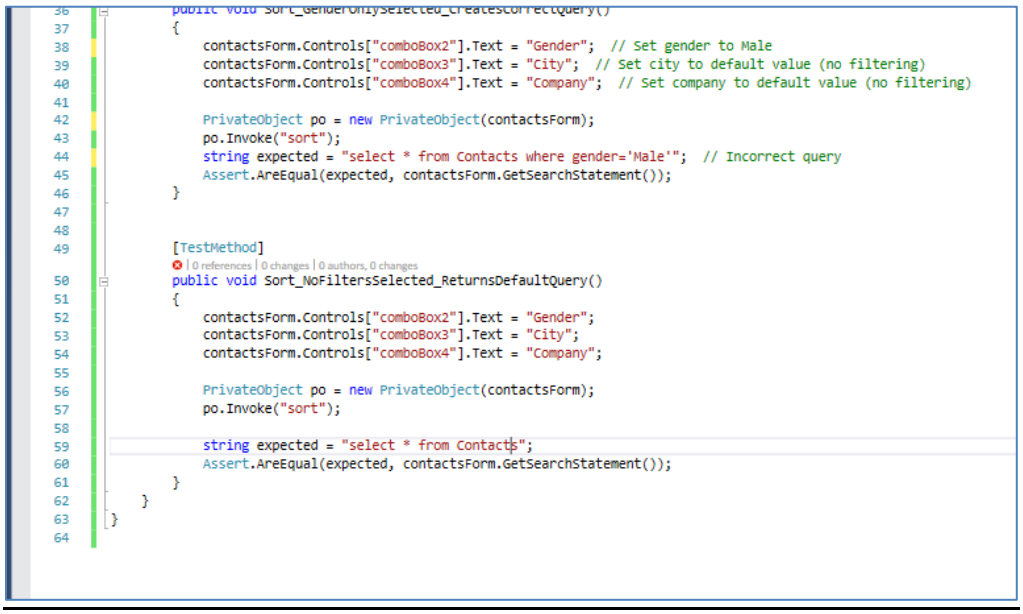
```
        }
        2 references | 0 changes | 0 authors, 0 changes
        private void contentenable()
        {
            txtaddress.Enabled = true;
            txtcity.Enabled = true;
            txtcompany.Enabled = true;
            txtdesignation.Enabled = true;
            txtfather.Enabled = true;
            gender.Enabled = true;
            button1.Enabled = true;
            button3.Enabled = true;
            button4.Enabled = true;
            txtcity.Text = "";
            txtaddress.Text = "";
            txtcompany.Text = "";
            txtcontact1.Text = "";
            txtcontact2.Text = "";
            txtcontact3.Text = "";
            txtdesignation.Text = "";
            txtemail.Text = "";
            txtemail.Text = "";
            txtfather.Text = "";
            txtname.Text = "";
            gender.Text = "Gander";
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button3_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are u sure to delete this contact with name "
                + txtname.Text, "Warning", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                obj11.commandexecutor("delete from contacts where upper(name)='"
                    + txtname.Text.ToUpper() + "' OR contact1='" + txtcontact1.Text + "' OR contact2='"
                    + txtcontact2.Text + "' OR contact3='" + txtcontact3.Text + "'");
                MessageBox.Show("Record is deleted");
            }
            else
            {
            }
        }

        1 reference | 0 changes | 0 authors, 0 changes
        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.dgwdata("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 \nPlese 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");
```

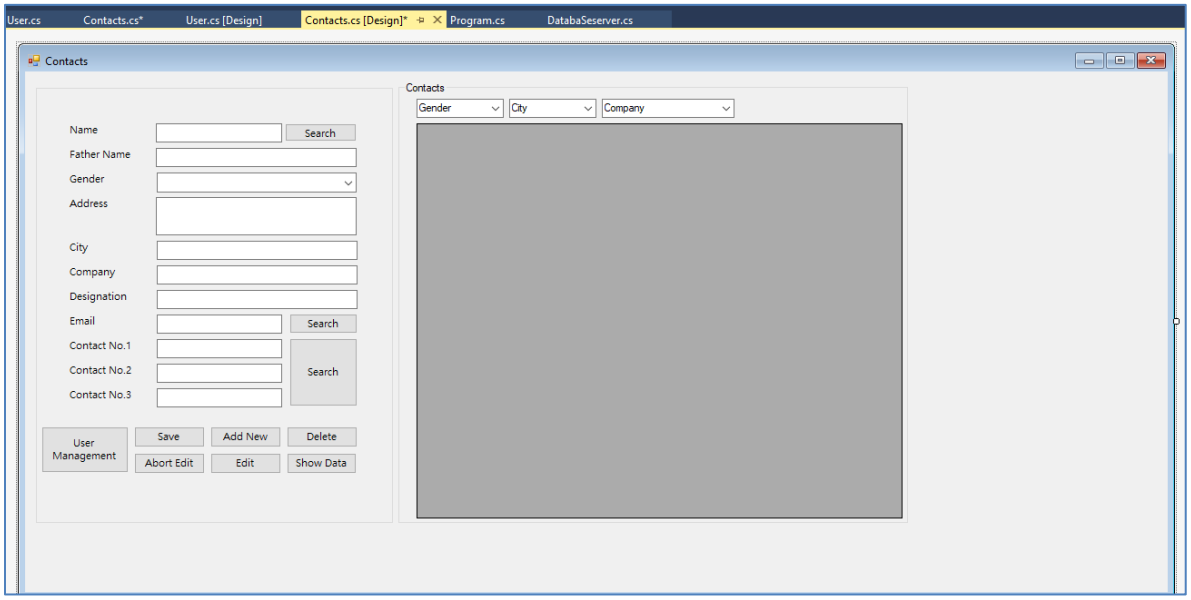
```
244                 button1.Enabled = false;
245                 button4.Text = "Update";
246             }
247             else if (button4.Text == "Update" && txtcontact1.Text.Length != 0)
248             {
249                 obj11.commandexecutor("update contacts set name='" + txtname.Text.Trim()
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         }
            1 reference | 0 changes | 0 authors, 0 changes
267         private void button9_Click(object sender, EventArgs e)
268         {
269             button4.Text = "Edit";
270             contentenable();
271             contentdisable();
272             button9.Enabled = false;
273         }
            1 reference | 0 changes | 0 authors, 0 changes
274         private void button10_Click(object sender, EventArgs e)
275         {
276             User User = new User();
277             User.Show();
278         }
279     }
280 }
```

ContactTest.cs:

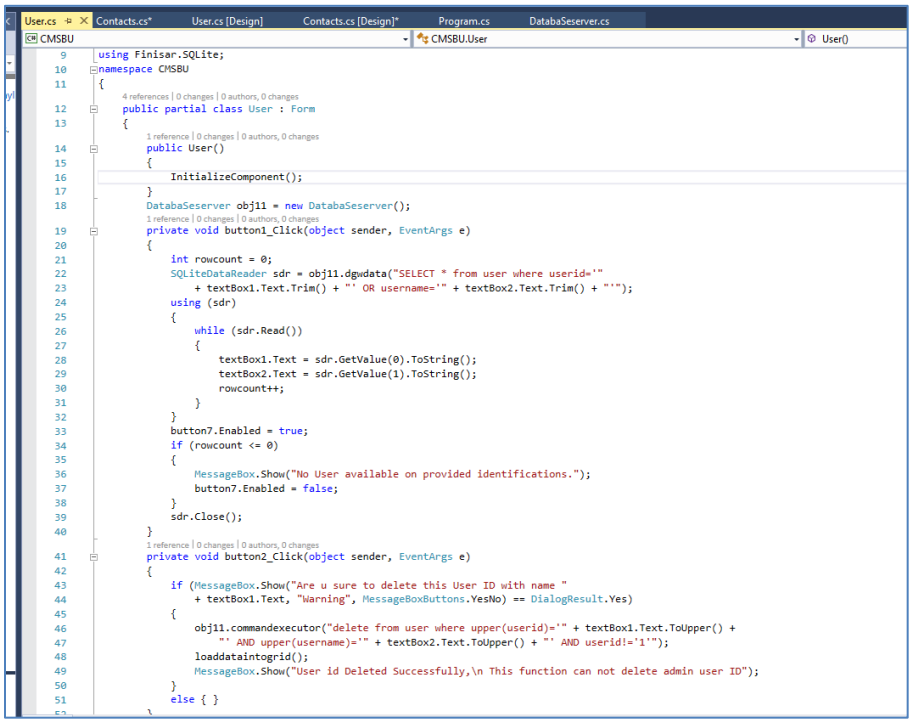




ContactDesign.cs:



User.cs:

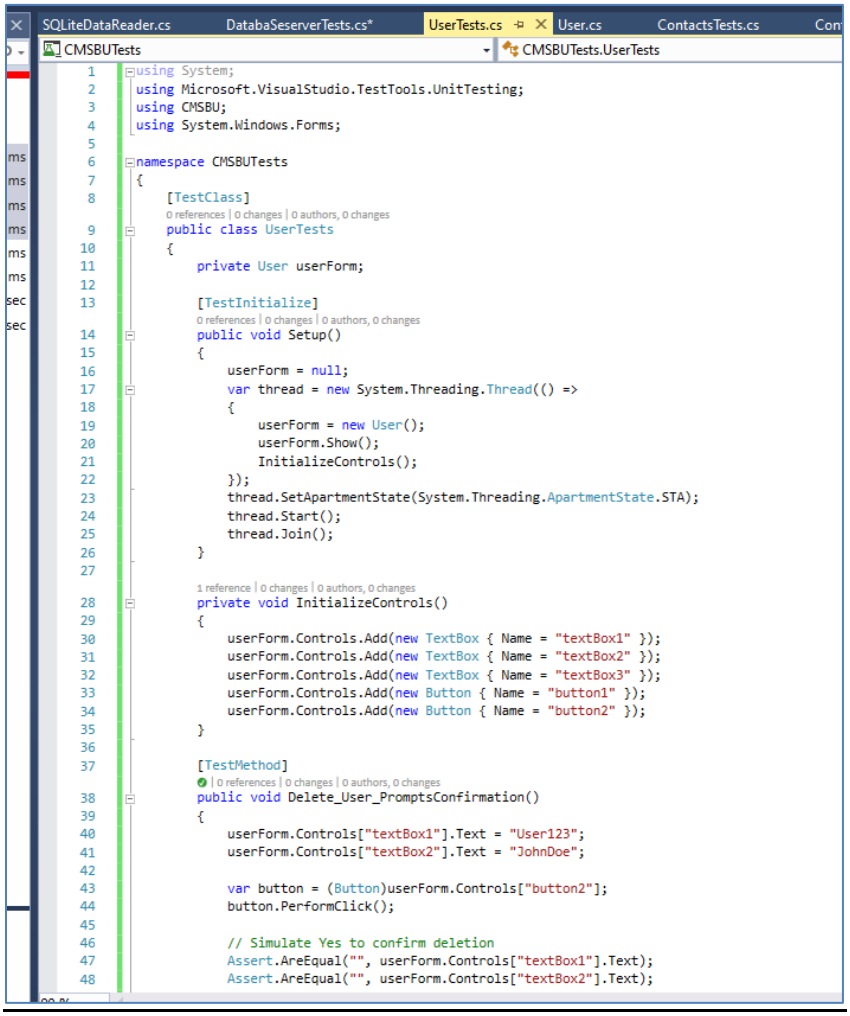


```
using Finisar.SQLite;
namespace CMSBU
{
    public partial class User : Form
    {
        public User()
        {
            InitializeComponent();
        }
        DatabaseServer obj11 = new DatabaseServer();
        private void button1_Click(object sender, EventArgs e)
        {
            int rowcount = 0;
            SQLiteDataReader sdr = obj11.dgwdata("SELECT * from user where userid='"
                + textBox1.Text.Trim() + "' OR username='" + textBox2.Text.Trim() + "'");
            using (sdr)
            {
                while (sdr.Read())
                {
                    textBox1.Text = sdr.GetValue(0).ToString();
                    textBox2.Text = sdr.GetValue(1).ToString();
                    rowcount++;
                }
            }
            button7.Enabled = true;
            if (rowcount <= 0)
            {
                MessageBox.Show("No User available on provided identifications.");
                button7.Enabled = false;
            }
            sdr.Close();
        }
        private void button2_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are u sure to delete this User ID with name "
                + textBox1.Text, "Warning", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                obj11.commandexecutor("delete from user where upper(userid)='" + textBox1.Text.ToUpper() +
                    "' AND upper(username)='" + textBox2.Text.ToUpper() + "' AND userid!='1'");
                loaddataintogrid();
                MessageBox.Show("User id Deleted Successfully,\n This function can not delete admin user ID");
            }
            else { }
```

```
                MessageBox.Show("User id Deleted Successfully,\n This function can not delete admin user ID");
            }
            else { }
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button3_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            button4.Enabled = true;
            button5.Enabled = true;
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void USer_Load(object sender, EventArgs e)
        {
            button4.Enabled = false;
            button5.Enabled = false;
            loaddataintogrid();
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button5_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            button4.Enabled = false;
            button5.Enabled = false;
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button4_Click(object sender, EventArgs e)
        {
            obj11.adduser(textBox2.Text, textBox3.Text);
            textBox1.Text = "";
            textBox2.Text = "";
            textBox3.Text = "";
            button4.Enabled = false;
            button5.Enabled = false;
            loaddataintogrid();
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button6_Click(object sender, EventArgs e)
        {
            if (textBox7.Text == textBox8.Text)
            {
                obj11.commandexecutor("update user set username='"
                    + textBox5.Text.Trim() + "', password='" + textBox8.Text.Trim() + "'");
                MessageBox.Show("Password Changed");
```

```
            if (textBox7.Text == textBox8.Text)
            {
                obj11.commandexecutor("update user set username='"
                    + textBox5.Text.Trim() + "', password='" + textBox8.Text.Trim() + "'");
                MessageBox.Show("Password Changed");
                textBox6.Text = "";
                textBox5.Text = "";
                textBox4.Text = "";
                textBox7.Text = "";
                textBox8.Text = "";
                textBox4.Enabled = false;
                textBox7.Enabled = false;
                textBox8.Enabled = false;
                button6.Enabled = false;
                button7.Enabled = false;
            }
            else
            {
                MessageBox.Show("Passwords are not matching");
            }
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button7_Click(object sender, EventArgs e)
        {
            textBox6.Text = textBox1.Text;
            textBox5.Text = textBox2.Text;
            textBox4.Enabled = true;
            textBox7.Enabled = true;
            textBox8.Enabled = true;
            button6.Enabled = true;
        }
        1 reference | 0 changes | 0 authors, 0 changes
        private void button8_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Pass 1=" + textBox7.Text + " \n Pass 2=" + textBox8.Text);
        }
        3 references | 0 changes | 0 authors, 0 changes
        private void loaddataintogrid()
        {
            SQLiteDataReader sdr = obj11.dgwdata("Select * from user");
            DataTable dt = new DataTable();
            dt.Columns.Add(new DataColumn("User ID", typeof(string)));
            dt.Columns.Add(new DataColumn("User Name", typeof(string)));
            using (sdr)
            {
                while (sdr.Read())
                {
```

UserTest.cs:




```cs
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using CMSBU;
using System.Windows.Forms;

namespace CMSBUTests
{
    [TestClass]
    0 references | 0 changes | 0 authors, 0 changes
    public class UserTests
    {
        private User userForm;

        [TestInitialize]
        0 references | 0 changes | 0 authors, 0 changes
        public void Setup()
        {
            userForm = null;
            var thread = new System.Threading.Thread(() =>
            {
                userForm = new User();
                userForm.Show();
                InitializeControls();
            });
            thread.SetApartmentState(System.Threading.ApartmentState.STA);
            thread.Start();
            thread.Join();
        }

        1 reference | 0 changes | 0 authors, 0 changes
        private void InitializeControls()
        {
            userForm.Controls.Add(new TextBox { Name = "textBox1" });
            userForm.Controls.Add(new TextBox { Name = "textBox2" });
            userForm.Controls.Add(new TextBox { Name = "textBox3" });
            userForm.Controls.Add(new Button { Name = "button1" });
            userForm.Controls.Add(new Button { Name = "button2" });
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void Delete_User_PromptsConfirmation()
        {
            userForm.Controls["textBox1"].Text = "User123";
            userForm.Controls["textBox2"].Text = "JohnDoe";

            var button = (Button)userForm.Controls["button2"];
            button.PerformClick();

            // Simulate Yes to confirm deletion
            Assert.AreEqual("", userForm.Controls["textBox1"].Text);
            Assert.AreEqual("", userForm.Controls["textBox2"].Text);
```

```cs
            button.PerformClick();

            // Simulate Yes to confirm deletion
            Assert.AreEqual("", userForm.Controls["textBox1"].Text);
            Assert.AreEqual("", userForm.Controls["textBox2"].Text);
        }

        [TestMethod]
        0 references | 0 changes | 0 authors, 0 changes
        public void Add_NewUser_AddsSuccessfully()
        {
            userForm.Controls["textBox2"].Text = "NewUser";
            userForm.Controls["textBox3"].Text = "Password123";

            var addButton = (Button)userForm.Controls["button4"];
            addButton.PerformClick();

            Assert.AreEqual("", userForm.Controls["textBox2"].Text);
            Assert.AreEqual("", userForm.Controls["textBox3"].Text);
        }
    }
}
```

UserDesign.cs: