ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Программная инженерия»

СОГЛАСОВАНО
профессор департамента программной инженерии
факультета компьютерных наук
канд. техн. наук

_____ В.В. Шилов
«___» _____ 2020 г

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»
профессор департамента программной инженерии, канд. техн. наук

_____ В.В. Шилов
«___» _____ 2020 г.

Десктопное приложение для анализа и прогнозирования потребности СТО в запасных деталях автомобилей

Текст программы

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.04.09-01 12 01-1-ЛУ

Исполнитель
студент группы __БПИ197_
_____ / Я.Янал /
«____»_____ 2020 г.

Москва 2020

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.09-01 12 01-1-ЛУ				

УТВЕРЖДЕН
RU.17701729.04.09-01 12 01-1

Десктопное приложение для анализа и прогнозирования потребности СТО в запасных деталях автомобилей

Текст программы

RU.17701729.04.09-01 12 01-1

Листов 157

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.04.09 -01 12 01-1				

МОСКВА 2020

Содержание

ТЕКСТ ПРОГРАММЫ

Программа состоит из трёх проектов: «Car_Service», «CarServiceLibrary», «DataBasesLibrary». Car_Service – был создан автоматически Windows.Forms и дополнялся в процессе разработки. CarServiceLibrary и DataBasesLibrary – библиотеки классов, созданн в процессе разработки. Далее будет приведён весь исходный код на языке C#.

1. БИБЛИОТЕКА CarServiceLibrary

1.1 DeleteFormDatabase.cs

```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;

namespace CarServiceLibrary
{

    public class DeleteFromDatabase
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//to save the conection text to database
        static SqlConnection conn;
        static SqlCommand cmd;

        /// <summary>
        /// constrocter
        /// </summary>
        public DeleteFromDatabase()
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
        }

        /// <summary>
        /// delete specefic register from database
        /// </summary>
```

```
        /// <param name="nameDatabase"> the name of database</param>
        /// <param name="Id"> id register in database</param>
        public void DeleteRow(string nameDatabase, string Id)
        {
            try
            {
                cmd = new SqlCommand($"delete from {nameDatabase} where Id='" + Id + "'");
                conn.Open();
                cmd.Connection = conn;
                cmd.ExecuteNonQuery();
                conn.Close();
            }
            catch (SqlException)
            {
                MessageBox.Show($"couldn't delete this data", "Unsuccessful operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            catch (ArgumentNullException ex)
            {
                MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }


    }
}
```

1.2 JSONserialize.cs

```
using System;
using System.Web.Script.Serialization;

namespace CarServiceLibrary
{
    /// <summary>
    /// seialization class
    /// </summary>
    public static class JSONserialize
    {
        /// <summary>
        /// serialize method
        /// </summary>
        /// <param name="tools">the array which will be sirialized</param>
        /// <returns>the string form of the array</returns>
        public static string Serialize(Tool[] tools)
        {
            JavaScriptSerializer dataContract = new JavaScriptSerializer();
            string serializedDataInStringFormat = dataContract.Serialize(tools);
            return serializedDataInStringFormat;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="serializedDataInStringFormat">the string form or the array  or Tool objects</param>
        /// <returns>the array of Tool objects which was saved at string form</returns>
        public static Tool[] Deserialize(string serializedDataInStringFormat)
        {
            JavaScriptSerializer dataContract = new JavaScriptSerializer();
            Tool[] tools = dataContract.Deserialize<Tool[]>(serializedDataInStringFormat);
            return tools;
```

```
        }


    }
}
```

1.3 ReadClass.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace CarServiceLibrary
{
    public class ReadClass
    {

        static SqlConnection Sql;
        readonly DataTable table = new DataTable();
        static FileInfo file;//help to get the path of database
        static string constring;//to save the conection text to database
        public ReadClass(string dataBaseName)
        {
            /*
            * initialize the connection to database
            */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            Sql = new SqlConnection(constring);
            GetData(dataBaseName);
        }

        /// <summary>
        /// get all data from specefic database
```

```
/// </summary>
/// <param name="dataBaseName"> name of database</param>
private void GetData(string dataBaseName)
{
    Sql.Open();
    SqlDataAdapter ad = new SqlDataAdapter($"select * from {dataBaseName}", Sql);
    ad.Fill(table);
    Sql.Close();

}
/// <summary>
/// help to get the database useing object of this class
/// </summary>
/// <returns>the data which was downloaded from specefic database</returns>
public DataTable GetDataTable() => table;

/// <summary>
/// find tools data which have shortage in their quantity and return it
/// </summary>
public DataTable CheckTools()
{
    DataTable badTools = new DataTable();
    badTools.Columns.Add("Tool_Name", typeof(string));
    badTools.Columns.Add("Min_Quan", typeof(string));
    badTools.Columns.Add("Max_Quan", typeof(string));
    badTools.Columns.Add("Current_Quan", typeof(string));
    object[] val = null;
    for (int i = 1; i < table.Rows.Count; i++)
    {
        val = table.Rows[i].ItemArray;

        if ((int)val[5] < (int)val[7])
        {
            badTools.Rows.Add(val[1].ToString(), val[7].ToString(), val[6].ToString(), val[5].ToString());
```

```
        }


    }
    return badTools;
}
/// <summary>
/// return tools data which have shortage in their quantity but in this method
/// will be returned all preporties of the these tools
/// </summary>
/// <returns></returns>
public DataTable GetBadTools()
{
    DataTable temp = table.Copy();
    object[] val;
    DataRow row = temp.NewRow();
    for (int i = 0; i < table.Rows.Count; i++)
    {
        val = temp.Rows[i].ItemArray;
        if ((int)val[5] >= (int)val[7])
        {
            temp.Rows[i].Delete();
        }
    }
    return temp;
}


/// <summary>
/// creat dictioary of companies where keys this dictionary is the names and
/// the values is email addresses of companies and return this dictionary
/// </summary>
public Dictionary<string, string> GetCompaniesNames()
{
    Dictionary<string, string> names = new Dictionary<string, string>();
    object[] val = null;
```

```
            for (int i = 0; i < table.Rows.Count; i++)
            {
                val = table.Rows[i].ItemArray;
                names.Add(val[1].ToString(), val[5].ToString());
            }

            return names;
        }

    }
}
```

1.4 SendEmail.cs

```
using System;
using System.Text;
using System.Net;
using System.Net.Mail;
using System.ComponentModel;
using System.Windows.Forms;



namespace CarServiceLibrary
{
    //this delegate help to callback some methods at SendOrder form.
    public delegate void Del();

    /// <summary>
    /// class sending an email
    /// </summary>
    public class SendEmail
    {
        /*
         * variables help in sending an email
         */
```

```
Del del;
DataGridView order;
NetworkCredential login;
SmtpClient client;
MailMessage msg;
string userName, password, message, receiver, subject, smtp;
int port;
bool ssl;

//constrocter (the names of the variables represent their function)
public SendEmail(string userName, string password, string message, string receiver,
   string subject, string smtp, int port, bool ssl, DataGridView grid)
{
   this.userName = userName;
   this.password = password;
   this.message = message;
   this.receiver = receiver;
   this.subject = subject;
   this.smtp = smtp;
   this.port = port;
   this.ssl = ssl;
   this.order = grid;
}

/// <summary>
/// the main process of sending an email
/// </summary>
public void Send(Del del)
{
   this.del = del;
   try
   {
      /*
       * you can know the function of every class or property by turning the mouse cursour over the code.
```

```
            */
            login = new NetworkCredential(userName, password);//get login and password
            client = new SmtpClient(smtp, port);//initialization for SmtpClient class which sends email (smtp: smtp server)
            client.EnableSsl = ssl;
            client.Credentials = login;
            msg = new MailMessage { From = new MailAddress(userName + smtp.Replace("smtp.", "@"), "Car Service",
Encoding.UTF8) };
            msg.To.Add(new MailAddress(receiver));
            msg.Body = message + "\n\n" + GetTabaleOrders();
            msg.Subject = subject;
            msg.BodyEncoding = Encoding.UTF8;
            msg.IsBodyHtml = true;
            msg.Priority = MailPriority.Normal;
            msg.DeliveryNotificationOptions = DeliveryNotificationOptions.OnFailure;
            client.SendCompleted += new SendCompletedEventHandler(SendCompletedCallback);
            string userstate = "sending....";
            client.SendAsync(msg, userstate);
        }
        catch (NullReferenceException)
        {
            MessageBox.Show($"Sorry you have forgot to fill some information please check it!!", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        /// <summary>
        /// creating a order table to send it in readable form.
        /// </summary>
        string GetTabaleOrders()
        {
            string tableOrders = "<table width='100%' style='border:Solid 1px Black;'>";
            tableOrders += "<tr>" + "<td stlye='color:blue;'>" + "Tool Name" + "</td>" + "<td stlye='color:blue;'>" +
                "Tool Unit" + "</td>" + "<td stlye='color:blue;'>" + "Quantity" + "</td>" + "<td stlye='color:blue;'>"
                + "Note" + "</td>" + "<tr>";
            foreach (DataGridViewRow row in order.Rows)
```

```
            {
                tableOrders += "<tr>";
                for (int i = 0; i < 5; i++)
                {
                    if (i != 2)
                        tableOrders += "<td stlye='color:blue;'>" + row.Cells[i].Value + "</td>";
                }
                tableOrders += "</tr>";
            }
            tableOrders += "</table>";
            return tableOrders;
        }

        //gives information about the result of sending an email.
        void SendCompletedCallback(object sender, AsyncCompletedEventArgs e)
        {
            if (e.Cancelled)
                MessageBox.Show($"{e.UserState} send cancelled", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            if (e.Error != null)
                MessageBox.Show($"Incorrect User Name or Password", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            else
            {
                MessageBox.Show($"Your message has been seccessfully sent.", "Message", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                this.del?.Invoke();
            }

        }

    }

}
```

```
}
```

1.5 Tool.cs

```
using System.Windows.Forms;

namespace CarServiceLibrary
{
    /// <summary>
    /// class help to creat tools array from bouth and saled tools in order to serialize it
    /// </summary>
    public class Tool
    {

        /*
         * required variables to save the tools
         */
        string toolName;
        string toolUnit;
        double purchasePrice;
        uint quantity;
        string note;
        public string ToolName { get => toolName; set => toolName = value; }
        public string ToolUnit { get => toolUnit; set => toolUnit = value; }
        public double PurchasePrice { get => purchasePrice; set => purchasePrice = value; }
        public uint Quantity { get => quantity; set => quantity = value; }
        public string Note { get => note; set => note = value; }

        /// <summary>
        /// constrocter
        /// </summary>
        /// <param name="row"> row of some register from datagridview</param>
        public Tool(DataGridViewRow row)
        {
```

```
            this.ToolName = row.Cells[0].Value.ToString();
            this.ToolUnit = row.Cells[1].Value.ToString();
            this.PurchasePrice = double.Parse(row.Cells[2].Value.ToString());
            this.Quantity = uint.Parse(row.Cells[3].Value.ToString());
            if (row.Cells[4].Value != null)
                this.Note = row.Cells[4].Value.ToString();

        }
        //becuase of serialization
        public Tool() { }
    }
}
```

2. БИБЛИОТЕКА DataBasesLibrary

2.1 CarDeliveryDatabase.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
    public class CarDeliveryDatabase
    {
        static FileInfo file;//help to get the path of database
        static string constring;//to save the conection text to database
        //help in adding register to database
        static readonly string carDeliveryData = $"INSERT INTO CarDelivery (First_Name, Second_Name, Car_Number," +
            $" Car_Type, Car_Color, Car_Model, Phone_Number, Adress, Entry_Date, Exit_Date, Identification_Number, Note,
Bill)" +
            $" VALUES (@First_Name, @Second_Name, @Car_Number, @Car_Type, @Car_Color, @Car_Model,
@Phone_Number, @Adress, @Entry_Date," +
            $" @Exit_Date, @Identification_Number, @Note, @Bill)";
```

```
        readonly SqlConnection conn;
        SqlCommand cmd;
        DataTable dt = new DataTable();
        SqlDataAdapter adapt;

        /// <summary>
        /// constrocter
        /// </summary>
        public CarDeliveryDatabase()
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
        }

        /// <summary>
        /// adding a data into database
        /// </summary>
        /// <param name="inputrow">get data in list form</param>
        public void AddData(List<string> inputrow)
        {
            cmd = new SqlCommand(carDeliveryData)
            {
                Connection = conn
            };
            conn.Open();
            cmd.Parameters.AddWithValue("@First_Name", inputrow[0]);
            cmd.Parameters.AddWithValue("@Second_Name", inputrow[1]);
            cmd.Parameters.AddWithValue("@Car_Number", inputrow[2]);
            cmd.Parameters.AddWithValue("@Car_Type", inputrow[3]);
```

```
    cmd.Parameters.AddWithValue("@Car_Color", inputrow[4]);
    cmd.Parameters.AddWithValue("@Car_Model", inputrow[5]);
    cmd.Parameters.AddWithValue("@Phone_Number", inputrow[6]);
    cmd.Parameters.AddWithValue("@Adress", inputrow[7]);
    cmd.Parameters.AddWithValue("@Entry_Date", inputrow[8]);
    cmd.Parameters.AddWithValue("@Exit_Date", inputrow[9]);
    cmd.Parameters.AddWithValue("@Identification_Number", inputrow[10]);
    cmd.Parameters.AddWithValue("@Note", inputrow[11]);
    cmd.Parameters.AddWithValue("@Bill", Encoding.ASCII.GetBytes(inputrow[12]));
    cmd.ExecuteNonQuery();
    conn.Dispose();
}

/// <summary>
/// get serialized bill of specefic register from CarDelivery database by knowing Id
/// </summary>
/// <param name="id"> id of specefic register </param>
/// <returns></returns>
public string GetBillRow(int id)
{
    string SQL = $"select * from CarDelivery where Id='" + id + "'";
    string temp = null;
    conn.Open();
    using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
    {
        while (reader.Read())
        {
            for (int i = 0; i < reader.FieldCount; i++)
            {
                if (i == 13)
                {
                    temp = Encoding.ASCII.GetString((byte[])reader[i]);
                }
```

```
            }
          }
        }
        return temp;
    }

    /// <summary>
    /// get data which was added to CarDelivery database between two specefic periods
    /// </summary>
    /// <param name="searchClient">list of client's information</param>
    /// <param name="from">the first period</param>
    /// <param name="to">the second period</param>
    /// <returns></returns>
    public DataTable SearchFor(List<string> searchClient, DateTime from, DateTime to)
    {
        conn.Open();
        adapt = new SqlDataAdapter("select * from CarDelivery where First_Name like '" + searchClient[0] + "%' and
Second_Name like '" + searchClient[1] + "%' and " +
            "Car_Number like '" + searchClient[2] + "%' and Car_Type like '" + searchClient[3] + "%' and Car_Color like '" +
searchClient[4] + "%' and Car_Model like '" + searchClient[5] + "%' and Entry_Date between '" + from + "' and '" + to + "'",
conn);
        dt = new DataTable();
        adapt.Fill(dt);
        conn.Close();
        return dt;
    }

    /// <summary>
    /// get column built from bills of cars which was delivered in some selected period
    /// </summary>
    /// <param name="from">the first period</param>
    /// <param name="to">the second period</param>
    /// <returns></returns>
    public List<string> GetforReport(DateTime from, DateTime to)
```

```
        {
            List<string> dates = new List<string>();
            using (SqlCommand cmd = new SqlCommand("SELECT Bill FROM CarDelivery where Exit_Date between '" + from +
"' and '" + to + "'", conn))
            using (SqlDataReader rdr = cmd.ExecuteReader())
            {
                while (rdr.Read())
                {
                    dates.Add(Encoding.ASCII.GetString((byte[])rdr.GetValue(0)));
                }
            }
            return dates;
        }
    }
}
```

2.2 CarRegisterDatabase.cs

```
 using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;
using System.IO;

namespace DataBasesLibrary
{
    public class CarRegisterDatabase
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//to save the conection text to database
        //help in adding new register to database
        static readonly string carRegisterData = $"INSERT INTO CarRegister (First_Name, Second_Name," +
            $" Car_Number,Car_Type, Car_Color, Car_Model, Phone_Number, Adress, Entry_Date, Identification_Number, Note)
```

```
VALUES (@First_Name, " +
        $"@Second_Name, @Car_Number, @Car_Type, @Car_Color, @Car_Model, @Phone_Number, @Adress,
@Entry_Date, @Identification_Number, @Note)";


    /// <summary>
    /// constrocter
    /// </summary>
    public CarRegisterDatabase()
    {
       /*
        * initialize the connection to database
        */
       file = new FileInfo(@"../../CarsDatabase.mdf");
       constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
    }
    /// <summary>
    /// adding new register into CarRegister database
    /// </summary>
    /// <param name="inputrow">get data in list form</param>
    public void AddData(List<string> inputrow)
    {
       SqlConnection conn = new SqlConnection(constring);
       SqlCommand cmd;
       cmd = new SqlCommand(carRegisterData)
       {
          Connection = conn
       };
       conn.Open();
       cmd.Parameters.AddWithValue("@First_Name", inputrow[0]);
       cmd.Parameters.AddWithValue("@Second_Name", inputrow[1]);
       cmd.Parameters.AddWithValue("@Car_Number", inputrow[2]);
       cmd.Parameters.AddWithValue("@Car_Type", inputrow[3]);
       cmd.Parameters.AddWithValue("@Car_Color", inputrow[4]);
```

```
            cmd.Parameters.AddWithValue("@Car_Model", inputrow[5]);
            cmd.Parameters.AddWithValue("@Phone_Number", inputrow[6]);
            cmd.Parameters.AddWithValue("@Adress", inputrow[7]);
            cmd.Parameters.AddWithValue("@Entry_Date", inputrow[8]);
            cmd.Parameters.AddWithValue("@Identification_Number", inputrow[9]);
            cmd.Parameters.AddWithValue("@Note", inputrow[10]);
            cmd.ExecuteNonQuery();
            conn.Dispose();
        }
    }
}
```

2.3 CompaniesDataBase.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
    public class CompaniesDataBase
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//to save the conection text to database
        //help in adding register to database
        static readonly string companiesData = $"INSERT INTO Companies (Company_Name, Company_Owner,
Phone_Number, Company_Adress, " +
            $"Company_Email, Date, Note) VALUES (@Company_Name, @Company_Owner, @Phone_Number,
@Company_Adress, @Company_Email, @Date, @Note)";
        readonly SqlConnection conn;
        SqlCommand cmd;
        DataTable dt = new DataTable();
        SqlDataAdapter adapt;

        /// <summary>
```

```
        /// constrocter
        /// </summary>
        public CompaniesDataBase()
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
        }



        /// <summary>
        /// adding a data into database
        /// </summary>
        /// <param name="inputrow">get data in list form</param>
        public void AddData(List<string> inputrow = null)
        {
            #region Add data to Companies Database
            cmd = new SqlCommand(companiesData)
            {
                Connection = conn
            };
            conn.Open();
            cmd.Parameters.AddWithValue("@Company_Name", inputrow[0]);
            cmd.Parameters.AddWithValue("@Company_Owner", inputrow[1]);
            cmd.Parameters.AddWithValue("@Phone_Number", inputrow[2]);
            cmd.Parameters.AddWithValue("@Company_Adress", inputrow[3]);
            cmd.Parameters.AddWithValue("@Company_Email", inputrow[4]);
            cmd.Parameters.AddWithValue("@Date", inputrow[5]);
            cmd.Parameters.AddWithValue("@Note", inputrow[6]);
            cmd.ExecuteNonQuery();
```

```
            conn.Dispose();
            #endregion
        }
        /// <summary>
        /// edit data of specific register in Companies database
        /// </summary>
        /// <param name="row">the renewed data of specific register</param>
        public void EditRow(List<string> row)
        {
            cmd = new SqlCommand("update Companies set Company_Name='" + row[1] + "', Company_Owner='" + row[2] + "',"
+
                " Phone_Number='" + row[3] + "', Company_Adress='" + row[4] + "', Company_Email='" + row[5] + "'," +
                " Date='" + row[6] + "', Note='" + row[7] + "' where Id='" + row[0] + "'");
            conn.Open();
            cmd.Connection = conn;
            cmd.ExecuteNonQuery();
            conn.Close();
        }


        /// <summary>
        /// get all data from Companies database which have similar company name and company's owner name
        /// </summary>
        /// <param name="name">company name</param>
        /// <param name="owner">company's owner name</param>
        /// <returns></returns>
        public DataTable SearchFor(string name, string owner)
        {
            conn.Open();
            adapt = new SqlDataAdapter("select * from Companies where Company_name like '" + name + "%' and
Company_Owner like '" + owner + "%'", conn);
            dt = new DataTable();
            adapt.Fill(dt);
            conn.Close();
```

```
            return dt;
        }

        /// <summary>
        /// get specific register from Companies data base by knowing company name and company's  owner name
        /// </summary>
        /// <param name="name">company name</param>
        /// <param name="owner">company's owner name</param>
        /// <returns></returns>
        public List<string> GetSpecificRow(string name, string owner)
        {
            List<string> row = new List<string>();
            using (SqlConnection conn = new SqlConnection(constring))
            {
                string SQL = $"select * from Companies where Company_name='" + name + "' and Company_Owner='" + owner +
"'";
                conn.Open();
                using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
                {
                    while (reader.Read())
                    {
                        for (int i = 1; i < reader.FieldCount - 1; i++)
                        {
                            row.Add(reader[i].ToString());


                        }
                    }
                }
            }
            return row;
        }
    }
}
```

2.4 EmployeesDatabase.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
    public class EmployeesDatabase
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//help to get the path of database
        //help in adding new register to database
        static readonly string staffsData = $"INSERT INTO Staffs (Name, Adresse, Phone_Number, Career_Type, Salary, " +
            $"Registration_Time, Note) VALUES (@Name, @Adresse, @Phone_Number, @Career_Type, @Salary,
@Registration_Time, @Note)";
        readonly SqlConnection conn;
        SqlCommand cmd;
        DataTable dt = new DataTable();
        SqlDataAdapter adapt;

        /// <summary>
        /// constrocter
        /// </summary>
        public EmployeesDatabase()
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
```

```
}


/// <summary>
/// adding a new register into Staffs database
/// </summary>
/// <param name="inputrow">get data in list form</param>
public void AddData(List<string> inputrow)
{

   cmd = new SqlCommand(staffsData)
   {
      Connection = conn
   };
   conn.Open();
   cmd.Parameters.AddWithValue("@Name", inputrow[0]);
   cmd.Parameters.AddWithValue("@Adresse", inputrow[1]);
   cmd.Parameters.AddWithValue("@Phone_Number", inputrow[2]);
   cmd.Parameters.AddWithValue("@Career_Type", inputrow[3]);
   cmd.Parameters.AddWithValue("@Salary", inputrow[4]);
   cmd.Parameters.AddWithValue("@Registration_Time", inputrow[5]);
   cmd.Parameters.AddWithValue("@Note", inputrow[6]);
   cmd.ExecuteNonQuery();
   conn.Dispose();
}

/// <summary>
/// edit data of specific register in Staffs database
/// </summary>
/// <param name="row">the renewed data of specific register</param>
public void EditRow(List<string> row)
{
   cmd = new SqlCommand("update Staffs set Name='" + row[1] + "', Adresse='" + row[2] + "'," +
      " Phone_Number='" + row[3] + "', Career_Type='" + row[4] + "', Salary='" + row[5] + "'," +
```

```
        " Registration_Time='" + row[6] + "', Note='" + row[7] + "' where Id='" + row[0] + "'");
    conn.Open();
    cmd.Connection = conn;
    cmd.ExecuteNonQuery();
    conn.Close();
}

/// <summary>
/// get all data from Staffs database which have similar employee name
/// </summary>
/// <param name="name">employee name</param>
/// <returns></returns>
public DataTable SearchFor(string name)
{
    conn.Open();
    adapt = new SqlDataAdapter("select * from Staffs where Name like '" + name + "%'", conn);
    dt = new DataTable();
    adapt.Fill(dt);
    conn.Close();
    return dt;
}

/// <summary>
/// getting the sum of salaries of all employees
/// </summary>
/// <returns></returns>
public double GetStaffWages()
{
    conn.Open();
    SqlCommand query = new SqlCommand("select SUM(Salary) from Staffs", conn);
    double staffWages = double.Parse(query.ExecuteScalar().ToString());
    conn.Close();
    return staffWages;
}
```

```
        /// <summary>
        /// get specific register from Staffs data base by knowing employee name
        /// </summary>
        /// <param name="name">employee name</param>
        /// <returns></returns>
        public List<string> GetSpecificRow(string name)
        {
            List<string> row = new List<string>();
            using (SqlConnection conn = new SqlConnection(constring))
            {
                string SQL = $"select * from Staffs where Name='" + name + "'";
                conn.Open();
                using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
                {
                    while (reader.Read())
                    {
                        for (int i = 1; i < reader.FieldCount - 1; i++)
                        {
                            row.Add(reader[i].ToString());


                        }
                    }
                }
            }
            return row;
        }
    }
}
```

2.5 Invoice.cs

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

```
using System.Data.SqlClient;
using System.IO;

namespace DataBasesLibrary
{
    public class Invoice
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//help to get the path of database
        readonly string invoiceData;
        SqlConnection conn;
        SqlCommand cmd;
        SqlDataAdapter adapt;
        readonly DataTable tempDT = new DataTable();

        /// <summary>
        /// constrocter
        /// </summary>
        /// <param name="dataBaseName"> name of database</param>
        public Invoice(string dataBaseName) : this()
        {
            //help in adding new register to database
            invoiceData = $"INSERT INTO {dataBaseName} (Tool_Name, Tool_Unit, Quantity, Total_Amount, Date) " +
                $"VALUES (@Tool_Name, @Tool_Unit, @Quantity, @Total_Amount, @Date)";

        }
        /// <summary>
        /// constrocter
        /// </summary>
        public Invoice()
        {
            /*
             * initialize the connection to database
             */
```

```
        file = new FileInfo(@"../../CarsDatabase.mdf");
        constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
    }
    /// <summary>
    /// adding a new register into Invoices database (there is two)
    /// </summary>
    /// <param name="inputrow">get data in list form</param>
    public void AddData(List<string> inputrow)
    {
        conn = new SqlConnection(constring);
        cmd = new SqlCommand(invoiceData)
        {
            Connection = conn
        };
        conn.Open();
        cmd.Parameters.AddWithValue("@Tool_Name", inputrow[0]);
        cmd.Parameters.AddWithValue("@Tool_Unit", inputrow[1]);
        cmd.Parameters.AddWithValue("@Quantity", inputrow[2]);
        cmd.Parameters.AddWithValue("@Total_Amount", inputrow[3]);
        cmd.Parameters.AddWithValue("@Date", inputrow[4]);
        cmd.ExecuteNonQuery();
        conn.Dispose();
    }



    /// <summary>
    /// get registers of tools between two specific periods
    /// </summary>
    /// <param name="from">the first period</param>
    /// <param name="to">the secod period</param>
    /// <param name="namedata">name of database</param>
    /// <returns></returns>
    public DataTable GetforReport(DateTime from, DateTime to, string namedata)
```

```
        {
            tempDT.Columns.Add("Id", typeof(int));
            tempDT.Columns.Add("Tool_Name", typeof(string));
            tempDT.Columns.Add("Tool_Unit", typeof(string));
            tempDT.Columns.Add("Quantity", typeof(int));
            tempDT.Columns.Add("Total_Amount", typeof(double));
            tempDT.Columns.Add("Date", typeof(DateTime));
            conn = new SqlConnection(constring);
            conn.Open();
            adapt = new SqlDataAdapter($"SELECT * FROM {namedata} where Date between '" + from + "' and '" + to + "'",
conn);
            adapt.Fill(tempDT);
            conn.Close();
            return tempDT;
        }
    }
}
```

2.6 ShoppingDatabase.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
    public class ShoppingDatabase
    {
        readonly FileInfo file;//help to get the path of database
        static string constring;//help to get the path of database
        //help in adding new register to database
        static readonly string shoppingData = $"INSERT INTO Shopping (Company_Name, Company_Owner, Phone_Number,
Adress, " +
```

```
        $"Email, Date, Bill) VALUES (@Company_Name, @Company_Owner, @Phone_Number, @Adress, @Email, @Date,
@Bill)";
    SqlDataAdapter adapt;
    readonly SqlConnection conn;
    SqlCommand cmd;
    DataTable dt;

    /// <summary>
    /// constrocter
    /// </summary>
    public ShoppingDatabase()
    {
        /*
         * initialize the connection to database
         */
        file = new FileInfo(@"../../CarsDatabase.mdf");
        constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
        conn = new SqlConnection(constring);
    }



    /// <summary>
    /// adding a new register into ShoppingDatabase database
    /// </summary>
    /// <param name="tableName">name of database</param>
    /// <param name="inputrow">list of data which will be added to database</param>
    public void AddData(string tableName, List<string> inputrow = null)
    {
        #region Add data to Shopping Database
        cmd = new SqlCommand(shoppingData)
        {
            Connection = conn
```

```
    };
    conn.Open();
    cmd.Parameters.AddWithValue("@Company_Name", inputrow[0]);
    cmd.Parameters.AddWithValue("@Company_Owner", inputrow[1]);
    cmd.Parameters.AddWithValue("@Phone_Number", inputrow[2]);
    cmd.Parameters.AddWithValue("@Adress", inputrow[3]);
    cmd.Parameters.AddWithValue("@Email", inputrow[4]);
    cmd.Parameters.AddWithValue("@Date", inputrow[5]);
    cmd.Parameters.AddWithValue("@Bill", Encoding.ASCII.GetBytes(inputrow[6]));
    cmd.ExecuteNonQuery();
    conn.Dispose();
    #endregion
}

/// <summary>
/// get specific register from Shoppingdatabase or from CompaniesDatabase where name of company equal
/// received name's value
/// </summary>
/// <param name="name">company name </param>
/// <param name="tableName">name of database</param>
/// <returns></returns>
public List<string> GetSpecificRow(string name, string tableName)
{
    List<string> row = new List<string>();
    using (SqlConnection conn = new SqlConnection(constring))
    {
        string SQL = $"select * from {tableName} where Company_name='" + name + "'";
        conn.Open();
        using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
        {
            while (reader.Read())
            {
                for (int i = 1; i < reader.FieldCount - 1; i++)
                {
```

```
                row.Add(reader[i].ToString());


            }
        }
    }
    return row;

}

/// <summary>
/// get specific register from Shopping database knowing id's  register
/// </summary>
/// <param name="id">id of data register</param>
/// <returns></returns>
public string GetBillRow(int id)
{
    string SQL = $"select * from Shopping where Id='" + id + "'";
    string temp = null;
    conn.Open();
    using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
    {
        while (reader.Read())
        {
            for (int i = 0; i < reader.FieldCount; i++)
            {
                if (i == 7)
                {
                    temp = Encoding.ASCII.GetString((byte[])reader[i]);
                }


            }
        }
    }
```

```
            return temp;



        }

        /// <summary>
        /// get data from Shopping database which has register time between two periods of time, similar company's name and similar company's owner name
        /// </summary>
        /// <param name="name">company's name</param>
        /// <param name="owner"> company's owner name</param>
        /// <param name="from">first period</param>
        /// <param name="to">second period</param>
        /// <returns>datatabe of found data</returns>
        public DataTable SearchFor(string name, string owner, DateTime from, DateTime to)
        {
            conn.Open();
            adapt = new SqlDataAdapter("select * from Shopping where Company_Name like '" + name + "%' and Company_Owner like '" + owner + "%' and Date between '" + from + "' and '" + to + "' ", conn);
            dt = new DataTable();
            adapt.Fill(dt);
            conn.Close();
            return dt;
        }

        /// <summary>
        /// get list of bills values which was added in specific period
        /// </summary>
        /// <param name="from">start period</param>
        /// <param name="to">end period</param>
        /// <returns></returns>
        public List<string> GetforReport(DateTime from, DateTime to)
        {
            List<string> dates = new List<string>();
```

```
        using (SqlCommand cmd = new SqlCommand("SELECT Bill FROM Shopping where Date between '" + from + "' and '"
+ to + "'", conn))
        using (SqlDataReader rdr = cmd.ExecuteReader())
        {
            while (rdr.Read())
            {
                dates.Add(Encoding.ASCII.GetString((byte[])rdr.GetValue(0)));
            }
        }
        return dates;
    }
  }
}
```

2.7 ToolsDatabase.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
  public class ToolsDatabase
  {
    readonly FileInfo file;//help to get the path of database
    static string constring;//help to get the path of database
    //help in adding new register to database
    static readonly string ToolsData = $"INSERT INTO Tools (Name_Tool, Tool_Unit, Sell_Price, Purchase_Price," +
      $" Current_Quantity, Max_Quantity, Min_Quantity, Note) VALUES (@Name_Tool, @Tool_Unit, @Sell_Price," +
      $" @Purchase_Price, @Current_Quantity, @Max_Quantity, @Min_Quantity, @Note)";
    SqlConnection conn;
    readonly SqlCommand cmd = new SqlCommand();
    SqlDataAdapter adapt;
```

```
        DataTable dt;

        /// <summary>
        /// constrocter
        /// </summary>
        public ToolsDatabase()
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
        }

        /// <summary>
        /// adding a new register into Tools database
        /// </summary>
        /// <param name="inputrow">get data in list form</param>
        public void AddData(List<string> inputrow)
        {
            cmd.Connection = conn;
            cmd.CommandText = ToolsData;
            conn.Open();
            cmd.Parameters.AddWithValue("@Name_Tool", inputrow[0]);
            cmd.Parameters.AddWithValue("@Tool_Unit", inputrow[1]);
            cmd.Parameters.AddWithValue("@Sell_Price", inputrow[2]);
            cmd.Parameters.AddWithValue("@Purchase_Price", inputrow[3]);
            cmd.Parameters.AddWithValue("@Current_Quantity", inputrow[4]);
            cmd.Parameters.AddWithValue("@Max_Quantity", inputrow[5]);
            cmd.Parameters.AddWithValue("@Min_Quantity", inputrow[6]);
            cmd.Parameters.AddWithValue("@Note", inputrow[7]);
            cmd.ExecuteNonQuery();
```

```
            conn.Dispose();
        }

        /// <summary>
        /// edit specific register by knowing Id's register
        /// </summary>
        /// <param name="row">renewed data register</param>
        public void EditRow(List<string> row)
        {
            conn.Open();
            cmd.Connection = conn;
            cmd.CommandText = "update Tools set Name_Tool='" + row[1] + "', Tool_Unit='" + row[2] + "'," +
                " Sell_Price='" + row[3] + "', Purchase_Price='" + row[4] + "', Current_Quantity='" + row[5] + "'," +
                " Max_Quantity='" + row[6] + "', Min_Quantity='" + row[7] + "', Note='" + row[8] + "' where Id='" + row[0] + "'";
            cmd.ExecuteNonQuery();
            conn.Close();
        }

        /// <summary>
        /// update data of specific register
        /// </summary>
        /// <param name="name">tool's name </param>
        /// <param name="unit">tool's unit</param>
        /// <param name="quantity">tool's quantity</param>
        public void EditQuantity(string name, string unit, int quantity)
        {
            List<string> temp = GetSpecificRow(name, unit);
            string note = temp.Count > 7 ? temp[7] : null;
            conn.Open();
            cmd.Connection = conn;
            cmd.CommandText = "update Tools set Name_Tool='" + temp[0] + "', Tool_Unit='" + temp[1] + "', Sell_Price='" +
temp[2] + "', " +
                "Purchase_Price='" + temp[3] + "', Current_Quantity='" + (int.Parse(temp[4]) + quantity).ToString() + "'," +
                " Max_Quantity='" + temp[5] + "', Min_Quantity='" + temp[6] + "', Note='" + note + "' " +
```

```
        "where Name_Tool='" + name + "' and Tool_Unit='" + unit + "'";
    cmd.ExecuteNonQuery();
    conn.Close();
}

/// <summary>
/// get specific register from Tools database where tool's name equal
/// received name's value  and tools's unit = unit
/// </summary>
/// <param name="name">tool's name </param>
/// <param name="unit">tool's unit</param>
/// <returns></returns>
public List<string> GetSpecificRow(string name, string unit)
{
    List<string> row = new List<string>();
    using (SqlConnection conn = new SqlConnection(constring))
    {
        string SQL = $"select * from Tools where Name_Tool='" + name + "' and Tool_Unit='" + unit + "'";
        conn.Open();
        using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
        {
            while (reader.Read())
            {
                for (int i = 1; i < reader.FieldCount - 1; i++)
                {
                    row.Add(reader[i].ToString());


                }
            }
        }
    }
    return row;
}
```

```
        /// <summary>
        ///  get all data from Tools database which have similar tool's name  and tools's unit
        /// </summary>
        /// <param name="name">tool's name</param>
        /// <param name="unit">tool's unit</param>
        /// <returns></returns>
        public DataTable SearchFor(string name, string unit)
        {
            conn.Open();
            adapt = new SqlDataAdapter("select * from Tools where Name_Tool like '" + name + "%' and Tool_Unit like '" + unit +
"%'", conn);
            dt = new DataTable();
            adapt.Fill(dt);
            conn.Close();
            return dt;
        }

        /// <summary>
        /// get purchases price of specific register knowing tool's name and tool's unit
        /// </summary>
        /// <param name="name">tool's name</param>
        /// <param name="unit">tool's unit</param>
        /// <returns></returns>
        public double GetPurchasesPrice(string name, string unit)
        {
            double price = 0;
            conn = new SqlConnection(constring);
            conn.Open();
            using (SqlCommand cmd = new SqlCommand("select Purchase_Price from Tools where Name_Tool='" + name + "' and
" +
                "Tool_Unit ='" + unit + "'", conn))
            using (SqlDataReader rdr = cmd.ExecuteReader())
            {
                while (rdr.Read())
```

```
            {
                price = double.Parse(rdr.GetValue(0).ToString());
            }
        }
        return price;

    }
  }
}
```

2.8 UserDatabase.cs

```
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.IO;

namespace DataBasesLibrary
{
   public class UserDatabase
   {
      readonly FileInfo file;//help to get the path of database
      static string constring;//help to get the path of database
      //help in adding new register to database
      static readonly string usersData = $"INSERT INTO users (User_Name, Password, Is_Admain, " +
         $"Note) VALUES (@User_Name, @Password, @Is_Admain, @Note)";
      readonly SqlConnection conn;
      DataTable dt = new DataTable();
      SqlDataAdapter adapt;
      SqlCommand cmd;

      /// <summary>
      /// constrocter
      /// </summary>
      public UserDatabase()
```

```
        {
            /*
             * initialize the connection to database
             */
            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            conn = new SqlConnection(constring);
        }

        /// <summary>
        /// adding a new register into users database
        /// </summary>
        /// <param name="inputrow">get data in list form</param>
        public void AddData(List<string> inputrow)
        {
            cmd = new SqlCommand(usersData)
            {
                Connection = conn
            };
            conn.Open();
            cmd.Parameters.AddWithValue("@User_Name", inputrow[0]);
            cmd.Parameters.AddWithValue("@Password", inputrow[1]);
            cmd.Parameters.AddWithValue("@Is_Admain", inputrow[2]);
            cmd.Parameters.AddWithValue("@Note", inputrow[3]);
            cmd.ExecuteNonQuery();
            conn.Dispose();
        }

        /// <summary>
        /// edit specific register by knowing Id's register
        /// </summary>
        /// <param name="row">renewed data register</param>
        public void EditRow(List<string> row)
```

```
{
    cmd = new SqlCommand("update users set User_Name='" + row[1] + "', Password='" + row[2] + "'," +
        " Is_Admain='" + row[3] + "', Note='" + row[4] + "' where Id='" + row[0] + "'");
    conn.Open();
    cmd.Connection = conn;
    cmd.ExecuteNonQuery();
    conn.Close();
}

/// <summary>
///  get all data from users database which have similar name
/// </summary>
/// <param name="name">user name</param>
public DataTable SearchFor(string name)
{
    conn.Open();
    adapt = new SqlDataAdapter("select * from users where User_Name like '" + name + "%'", conn);
    dt = new DataTable();
    adapt.Fill(dt);
    conn.Close();
    return dt;
}

/// <summary>
///  get specific register from Tools database where user name equal
///  received name's value  and user's password = password
/// </summary>
/// <param name="name">user name </param>
/// <param name="password">user password</param>
/// <returns></returns>
public List<string> GetSpecificRow(string name, string password)
{
    List<string> row = new List<string>();
    using (SqlConnection conn = new SqlConnection(constring))
```

```
        {
            string SQL = $"select * from users where User_Name='" + name + "' and Password='" + password + "'";
            conn.Open();
            using (SqlDataReader reader = new SqlCommand(SQL, conn).ExecuteReader())
            {
                while (reader.Read())
                {
                    for (int i = 1; i < reader.FieldCount - 1; i++)
                    {
                        row.Add(reader[i].ToString());


                    }
                }
            }
        }
        return row;
    }
  }
}
```

3. ПРОЕКТ Car_Service

3.1 Add.cs

```
using System;
using System.Windows.Forms;

namespace Car_Service
{
    /// <summary>
    /// inherited class from Windows.Forms.UserControl class was Shared in config initializeuring interface form
    /// </summary>
    public partial class Add : UserControl
    {
        bool isAdmain; //determine if the user is admain to make some spicial property for him
        public bool IsAdmain { get => isAdmain; set => isAdmain = value; } //isAdmain's property
```

```
public Add()
{
   InitializeComponent();
}

/// <summary>
/// event move user to AddUser form when click on pic
/// </summary>
private void AddUser_pictureBox1_Click(object sender, EventArgs e)
{
   //accses there only for admains
   if (IsAdmain)
   {
      try
      {
         AddUser addUser = new AddUser();
         addUser.ShowDialog();
      }
      catch (InvalidOperationException ex)
      {
         MessageBox.Show($"{ex.Message}",
        "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
      }
   }
   else
      MessageBox.Show($"You are not an admain, so you can't add a user!!",
         "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

/// <summary>
/// event move user to AddCompanie form when click on pic
/// </summary>
private void AddCompany_pictureBox3_Click(object sender, EventArgs e)
```

```
{
   try
   {
      AddCompanie companie = new AddCompanie();
      companie.ShowDialog();
   }
   catch (InvalidOperationException ex)
   {
      MessageBox.Show($"{ex.Message}",
         "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
   }
}

/// <summary>
/// event move user to AddTool form when click on pic
/// </summary>
private void AddTool_pictureBox2_Click(object sender, EventArgs e)
{
   try
   {
      AddTool tool = new AddTool();
      tool.ShowDialog();
   }
   catch (InvalidOperationException ex)
   {
      MessageBox.Show($"{ex.Message}",
         "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
   }
}

/// <summary>
/// event move user to AddEmployees form when click on pic
/// </summary>
private void AddEmployee_pictureBox4_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                AddEmployees employees = new AddEmployees();
                employees.ShowDialog();
            }
            catch (InvalidOperationException ex)
            {
                MessageBox.Show($"{ex.Message}",
                    "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
    }
}
```

3.2 AddCarRegister.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using DataBasesLibrary;
using System.Data.SqlClient;

namespace Car_Service
{
    public partial class AddCarRegister : Form
    {
        //constroctor
        public AddCarRegister()
        {
            InitializeComponent(); //F12 to see the functions of this method
            entryDate_dateTimePicker.Value = DateTime.Now;//determine the time of adding car register
        }
```

```
/// <summary>
/// button click event to add car register to datebase
/// </summary>
private void Add_CarRegister_button_Click(object sender, EventArgs e)
{
   /*
    * testing the input data
    */
   bool isFullData = ISFullData();
   if (isFullData && !HasQuotationChar() && uint.TryParse(phoneNumber_textBox1.Text, out _))
   {
      try
      {
         //creat list from user input
         List<string> registerCar = new List<string>() { fName_textBox.Text,sName_textBox.Text,
            carNumber_textBox.Text,type_textBox1.Text,color_comboBox1.SelectedItem.ToString(),
            model_comboBox2.SelectedItem.ToString(), phoneNumber_textBox1.Text,adress_textBox.Text,
            entryDate_dateTimePicker.Value.ToString(),identif_textBox.Text,note_textBox.Text};
         CarRegisterDatabase car = new CarRegisterDatabase();
         //send list to CarRegisterDatabase.AddData method in order to add this data of carregister to database
         car.AddData(registerCar);
         MessageBox.Show($"The car was added successfully.", "Successful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
         CleanBoxes();//clean the boxes from previous input

      }
      catch (SqlException)
      {
         MessageBox.Show($"Could not get the DataBase!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);

      }

   }
```

```
    else if (isFullData)
    {
        MessageBox.Show($"Phone Number box can't contain a char or negative number!!",
            "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

/// <summary>
/// make texts of boxes empty
/// </summary>
private void CleanBoxes()
{
    fName_textBox.Text = "";
    sName_textBox.Text = "";
    phoneNumber_textBox1.Text = "";
    carNumber_textBox.Text = "";
    type_textBox1.Text = "";
    color_comboBox1.Text = "";
    model_comboBox2.Text = "";
    adress_textBox.Text = "";
    note_textBox.Text = "";
    identif_textBox.Text = "";
    entryDate_dateTimePicker.Value = DateTime.Now;
}

/// <summary>
/// check if boxes contain input
/// </summary>
/// <returns>true for full data, and false for not</returns>
private bool ISFullData()
{
    try
    {
        if (!string.IsNullOrEmpty(fName_textBox.Text.Trim()) && !string.IsNullOrEmpty(sName_textBox.Text.Trim()) &&
```

```
            !string.IsNullOrEmpty(type_textBox1.Text.Trim()) && !string.IsNullOrEmpty(adress_textBox.Text.Trim()) &&
            !string.IsNullOrEmpty(carNumber_textBox.Text.Trim()) && color_comboBox1.SelectedItem.ToString() != "" &&
             model_comboBox2.SelectedItem.ToString() != "" &&
            !string.IsNullOrEmpty(identif_textBox.Text.Trim()))
            return true;


    }
    catch (NullReferenceException)
    {
        MessageBox.Show("Incorrect input", "Erorr Inputing", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// check if input has quotation mark
/// </summary>
/// <returns></returns>
private bool HasQuotationChar()
{
    if (fName_textBox.Text.Contains("'") || sName_textBox.Text.Contains("'") || carNumber_textBox.Text.Contains("'") ||
        type_textBox1.Text.Contains("'") || adress_textBox.Text.Contains("'") || identif_textBox.Text.Contains("'")
        || note_textBox.Text.Contains("'"))
    {
        MessageBox.Show($"Unclosed quotation mark after the character ( ' )!!", "Unsuccessful Operation",
              MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
```

```
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
        private void FName_textBox_Validating(object sender, CancelEventArgs e)
        {
            fName_textBox.Text = fName_textBox.Text.Trim();
            sName_textBox.Text = sName_textBox.Text.Trim();
            phoneNumber_textBox1.Text = phoneNumber_textBox1.Text.Trim();
            carNumber_textBox.Text = carNumber_textBox.Text.Trim();
            type_textBox1.Text = type_textBox1.Text.Trim();
            adress_textBox.Text = adress_textBox.Text.Trim();
            note_textBox.Text = note_textBox.Text.Trim();
            identif_textBox.Text = identif_textBox.Text.Trim();
        }



        /*
         * these two events help to make good Effects when cursor mouse enter or leave  Add_CarRegister_button boundaries
         */
        private void Add_CarRegister_button_MouseEnter(object sender, EventArgs e)
        {

            Add_CarRegister_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
            Add_CarRegister_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
        }
        private void Add_CarRegister_button_MouseLeave(object sender, EventArgs e)
        {
            Add_CarRegister_button.FlatAppearance.BorderColor = System.Drawing.Color.LightGray;
            Add_CarRegister_button.ForeColor = System.Drawing.Color.LightGray;
        }
    }
}
```

3.3 AddCompanie.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data.SqlClient;
using DataBasesLibrary;



namespace Car_Service
{
    public partial class AddCompanie : Form
    {
        /// <summary>
        /// constrocter
        /// </summary>
        public AddCompanie()
        {
            InitializeComponent();//F12 to see the functions of this mehtod
        }


        /// <summary>
        /// button click event to add company to datebase
        /// </summary>
        private void Add_Company_button_Click(object sender, EventArgs e)
        {
            /*
             * testing the input data
             */
            if (ISFullData() && !HasQuotationChar() && CheckPhoneNumber() && IsValidEmail() && !Exist())
            {
                try
                {
```

```
            //creat list from user input
            List<string> companyPro = new List<string>() {companyName_textBox.Text,companyOwner_textBox.Text,
                phoneNumber_textBox.Text,adress_textBox.Text,email_textBox1.Text,
                entryDate_dateTimePicker.Value.ToString(),note_textBox.Text };
            //send list to CompaniesDataBase.AddData method in order to add this company's data to database
            CompaniesDataBase companies = new CompaniesDataBase();
            companies.AddData(companyPro);
            MessageBox.Show($"The company was added successfully.", "Successful Operation",
              MessageBoxButtons.OK, MessageBoxIcon.Information);
            CleanBoxes();//clean the boxes from previous input


        }
        catch (SqlException)
        {
            MessageBox.Show($"Could not get the DataBase!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);


        }
    }
}

/// <summary>
/// check if input has quotation mark
/// </summary>
/// <returns></returns>
private bool HasQuotationChar()
{
    if (companyName_textBox.Text.Contains("'") || companyOwner_textBox.Text.Contains("'") ||
        adress_textBox.Text.Contains("'") || note_textBox.Text.Contains("'"))
    {
        MessageBox.Show($"Unclosed quotation mark after the character ( ' )!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
```

```
    return false;
}

/// <summary>
/// check if company's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
    CompaniesDataBase companies = new CompaniesDataBase();
    if (companies.GetSpecificRow(companyName_textBox.Text, companyOwner_textBox.Text).Count > 0)
    {

        MessageBox.Show($"This Company already exist in you database!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// make sure if phonenumber box contain only possitive numbers
/// </summary>
/// <returns>false if phonenumber box contain char or negative number</returns>
private bool CheckPhoneNumber()
{
    if (uint.TryParse(phoneNumber_textBox.Text, out _))
        return true;
    MessageBox.Show($"Phone Number box can't contain a char or negative number!!",
            "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}
```

```
/// <summary>
/// check if email address input is valid or not
/// </summary>
bool IsValidEmail()
{
    bool methodresult = false;
    try
    {

        var addr = new System.Net.Mail.MailAddress(email_textBox1.Text);
        methodresult = addr.Address == email_textBox1.Text;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}, please enter another one!!",
            "Invalid Email Adress", MessageBoxButtons.OK, MessageBoxIcon.Warning);


    }
    return methodresult;
}

/// <summary>
/// make texts of boxes empty
/// </summary>
private void CleanBoxes()
{
    companyName_textBox.Text = "";
    companyOwner_textBox.Text = "";
    phoneNumber_textBox.Text = "";
    adress_textBox.Text = "";
    email_textBox1.Text = "";
    note_textBox.Text = "";
    entryDate_dateTimePicker.Value = DateTime.Now;
```

```
        }

        /// <summary>
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool ISFullData()
        {
            if (!string.IsNullOrEmpty(companyName_textBox.Text.Trim()) &&
!string.IsNullOrEmpty(companyOwner_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(adress_textBox.Text.Trim()) && !string.IsNullOrEmpty(email_textBox1.Text.Trim()))
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
        private void FName_textBox_Validating(object sender, CancelEventArgs e)
        {
            companyName_textBox.Text = companyName_textBox.Text.Trim();
            companyOwner_textBox.Text = companyOwner_textBox.Text.Trim();
            phoneNumber_textBox.Text = phoneNumber_textBox.Text.Trim();
            adress_textBox.Text = adress_textBox.Text.Trim();
            email_textBox1.Text = email_textBox1.Text.Trim();
            note_textBox.Text = note_textBox.Text.Trim();
        }

        /*
         * these two events help to make good Effects when cursor mouse enter or leave  Add_Company_button boundaries
         */
        private void Add_Company_button_MouseEnter(object sender, EventArgs e)
```

```
        {
            Add_Company_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
            Add_Company_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
        }
        private void Add_Company_button_MouseLeave(object sender, EventArgs e)
        {
            Add_Company_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Add_Company_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.4 AddEmployees.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using DataBasesLibrary;
using System.Data.SqlClient;



namespace Car_Service
{
    public partial class AddEmployees : Form
    {
        //constrocter
        public AddEmployees()
        {
            InitializeComponent();//F12 to see the functions of this method
            register_dateTimePicker.Value = DateTime.Now;//determine the time of adding car register
        }


        /// <summary>
        /// button click event to add employee to datebase
        /// </summary>
```

```
private void Add_Employee_button_Click(object sender, EventArgs e)
{
   /*
   * testing the input data
   */
   if (ISFullData() && !HasQuotationChar() && CheckPhoneNumber() && !Exist())
   {
      //creat list from user input
      List<string> employee = new List<string>
       {
          employeeName_textBox.Text,
          adress_textBox.Text,
          phoneNumber_textBox.Text,
          career_textBox.Text,
          salary_textBox.Text,
          register_dateTimePicker.Value.ToString(),
          note_textBox.Text
       };
      try
      {
         EmployeesDatabase employees = new EmployeesDatabase();
         //send list to EmployeesDatabase.AddData method in order to add this employee's data to database
         employees.AddData(employee);
         MessageBox.Show($"The employee was added successfully.", "Successful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
         CleanBoxes();//clean the boxes from previous input
      }
      catch (SqlException)
      {
         MessageBox.Show($"Salary box can't contain a char!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);

      }
   }
```

```
}

/// <summary>
/// make sure if phonenumber box contain only possitive numbers
/// </summary>
/// <returns>false if phonenumber box contain char or negative number</returns>
private bool CheckPhoneNumber()
{
    if (uint.TryParse(phoneNumber_textBox.Text, out _))
        return true;
    MessageBox.Show($"Phone Number box can't contain a char or negative number!!",
          "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// make texts of boxes empty
/// </summary>
private void CleanBoxes()
{
    employeeName_textBox.Text = "";
    adress_textBox.Text = "";
    phoneNumber_textBox.Text = "";
    career_textBox.Text = "";
    salary_textBox.Text = "";
    note_textBox.Text = "";
    register_dateTimePicker.Value = DateTime.Now;
}

/// <summary>
/// check if employee's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
```

```
        {
            EmployeesDatabase employees = new EmployeesDatabase();
            if (employees.GetSpecificRow(employeeName_textBox.Text).Count > 0)
            {
                MessageBox.Show($"This Employee's name already exist in you database, Please choose another one!!",
"Unsuccessful Operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return true;
            }
            return false;
        }

        /// <summary>
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool ISFullData()
        {
            if (!string.IsNullOrEmpty(employeeName_textBox.Text.Trim()) &&
!string.IsNullOrEmpty(phoneNumber_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(adress_textBox.Text.Trim()) && !string.IsNullOrEmpty(career_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(salary_textBox.Text.Trim()))
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// check if input has quotation mark
        /// </summary>
        /// <returns></returns>
        private bool HasQuotationChar()
        {
```

```
    if (employeeName_textBox.Text.Contains("'") || adress_textBox.Text.Contains("'") ||
        career_textBox.Text.Contains("'") || note_textBox.Text.Contains("'"))
    {
        MessageBox.Show($"Unclosed quotation mark after the character ( ' )!!", "Unsuccessful Operation",
              MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}




/// <summary>
/// event fire after each change on boxes to delete extra space from first and end the input
/// </summary>
private void EmployeeName_textBox_Validating(object sender, CancelEventArgs e)
{
    employeeName_textBox.Text = employeeName_textBox.Text.Trim();
    adress_textBox.Text = adress_textBox.Text.Trim();
    phoneNumber_textBox.Text = phoneNumber_textBox.Text.Trim();
    career_textBox.Text = career_textBox.Text.Trim();
    salary_textBox.Text = salary_textBox.Text.Trim();
    note_textBox.Text = note_textBox.Text.Trim();
}


/*
 * these two events help to make good Effects when cursor mouse enter or leave  Add_Employee_button boundaries
 */
private void Add_Employee_button_MouseEnter(object sender, EventArgs e)
{
    Add_Employee_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Add_Employee_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
}
private void Add_Employee_button_MouseLeave(object sender, EventArgs e)
```

```
        {
            Add_Employee_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Add_Employee_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.5 AddTool.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data.SqlClient;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class AddTool : Form
    {
        //constrocter
        public AddTool()
        {
            InitializeComponent();//F12 to see the functions of this method
        }

        /// <summary>
        /// button click event to add tool to datebase
        /// </summary>
        private void AddTool_button_Click(object sender, EventArgs e)
        {

            /*
            * testing the input data
            */
            if (IsFullData() && !HasQuotationChar() && CheckQuantity() && CheckPrices() && !Exist())
```

```
    {
        try
        {
            //creat list from user input
            List<string> tool = new List<string>() { name_textBox.Text, unit_textBox.Text,
                sellPrice_textBox.Text, purchasePrice_textBox.Text, current_numericUpDown.Value.ToString(),
                max_numericUpDown.Value.ToString(),min_numericUpDown.Value.ToString(),note_textBox.Text};
            ToolsDatabase toolsData = new ToolsDatabase();
            //send list to ToolsDatabase.AddData method in order to add this tool's data to database
            toolsData.AddData(tool);
            MessageBox.Show($"The tool was added successfully.", "Successful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
            CleanBoxes();//clean the boxes from previous input

        }
        catch (SqlException)
        {
            MessageBox.Show($"Could not get the DataBase!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

    }

}

/// <summary>
/// check if employee's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
    try
    {
        ToolsDatabase toolsData = new ToolsDatabase();
```

```
        if (toolsData.GetSpecificRow(name_textBox.Text, unit_textBox.Text).Count > 0)
        {
            MessageBox.Show($"This tool already exist in you database!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return true;
        }
    }
    catch (SqlException)
    {
        MessageBox.Show($"Could not get the DataBase!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }

    return false;
}

/// <summary>
/// check if input has quotation mark
/// </summary>
/// <returns></returns>
private bool HasQuotationChar()
{
    if (name_textBox.Text.Contains("'") || unit_textBox.Text.Contains("'") ||
        note_textBox.Text.Contains("'"))
    {
        MessageBox.Show($"Unclosed quotation mark after the character ( ' )!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// check the type and the logical of entering prices
```

```
        /// </summary>
        /// <returns></returns>
        private bool CheckPrices()
        {
            if (double.TryParse(sellPrice_textBox.Text, out double sell) && double.TryParse(purchasePrice_textBox.Text, out
double purchase))
            {
                if (sell >= purchase)
                    return true;
                //if selling price lower than  purchase the user will be asked about that (it might be a typo)
                else if (DialogResult.Yes == MessageBox.Show("Are you sure that you will sell this tool at a lower price than
purchase price ?",
                    "Strange Input", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
                    return true;
                return false;
            }
            else
            {
                MessageBox.Show($"Prices boxes can't contain a char!!", "Unsuccessful Operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
        }

        /// <summary>
        /// check the logical of tool's quantity input
        /// </summary>
        /// <returns>true for logical input, fasle for not</returns>
        private bool CheckQuantity()
        {
            if (current_numericUpDown.Value <= max_numericUpDown.Value && min_numericUpDown.Value <=
max_numericUpDown.Value)
                return true;
            MessageBox.Show($"The order of quantity is not logical!!", "Unsuccessful Operation",
```

```
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// check if boxes contain input
/// </summary>
/// <returns>true for full data, and false for not</returns>
private bool IsFullData()
{
    if (!string.IsNullOrEmpty(name_textBox.Text.Trim()) && !string.IsNullOrEmpty(unit_textBox.Text.Trim()))
        return true;
    MessageBox.Show($"You have forgot to fill some information, please check your input!!",
        "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// make texts of boxes empty
/// </summary>
private void CleanBoxes()
{
    name_textBox.Text = "";
    unit_textBox.Text = "";
    sellPrice_textBox.Text = "";
    purchasePrice_textBox.Text = "";
    current_numericUpDown.Value = 0;
    min_numericUpDown.Value = 0;
    max_numericUpDown.Value = 0;
    note_textBox.Text = "";
}

/// <summary>
/// event fire after each change on boxes to delete extra space from first and end the input
```

```
        /// </summary>
        private void Name_textBox_Validating(object sender, CancelEventArgs e)
        {
            name_textBox.Text = name_textBox.Text.Trim();
            unit_textBox.Text = unit_textBox.Text.Trim();
            sellPrice_textBox.Text = sellPrice_textBox.Text.Trim();
            purchasePrice_textBox.Text = purchasePrice_textBox.Text.Trim();
            note_textBox.Text = note_textBox.Text.Trim();
        }

        /*
         * these two events help to make good Effects when cursor mouse enter or leave  Add_Employee_button boundaries
         */
        private void AddTool_button_MouseEnter(object sender, EventArgs e)
        {
            AddTool_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
            AddTool_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
        }
        private void AddTool_button_MouseLeave(object sender, EventArgs e)
        {
            AddTool_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            AddTool_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.6 AddUser.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data.SqlClient;
using DataBasesLibrary;

namespace Car_Service
```

```
{
    public partial class AddUser : Form
    {

        //constrocter
        public AddUser()
        {
            InitializeComponent();//F12 to see the functions of this method
        }

        /// <summary>
        /// button click event to add user to datebase
        /// </summary>
        private void Add_User_button_Click(object sender, EventArgs e)
        {
            /*
             * testing input's data
             */
            if (ISFullData() && !HasQuotationChar() && !Exist())
            {
                //creat list from user input
                List<string> userpro = new List<string>() { userName_textBox.Text, password_textBox.Text,
                    isAdmain_comboBox.Text, note_textBox2.Text };
                try
                {
                    UserDatabase _user = new UserDatabase();
                    //send list to UserDatabase.AddData method in order to add this user's data to database
                    _user.AddData(userpro);
                    MessageBox.Show($"The User was added successfully.", "Successful Operation",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
                    CleanBoxes();//clean the boxes from previous input
                }
                catch (SqlException)
                {
```

```
            MessageBox.Show($"Could not get the DataBase!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

    }
}

/// <summary>
/// check if user's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
    UserDatabase _user = new UserDatabase();
    if (_user.GetSpecificRow(userName_textBox.Text, password_textBox.Text).Count > 0)
    {
        MessageBox.Show($"This user already exist in your database!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// make texts of boxes empty
/// </summary>
private void CleanBoxes()
{
    userName_textBox.Text = "";
    password_textBox.Text = "";
    isAdmain_comboBox.Text = "";
    note_textBox2.Text = "";
}
```

```
        /// <summary>
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool ISFullData()
        {
            if (!string.IsNullOrEmpty(userName_textBox.Text.Trim()) && !string.IsNullOrEmpty(password_textBox.Text.Trim())
&&
                isAdmain_comboBox.SelectedItem != null)
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!",
                "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// check if input has quotation mark
        /// </summary>
        /// <returns></returns>
        private bool HasQuotationChar()
        {
            if (userName_textBox.Text.Contains("'") || password_textBox.Text.Contains("'") ||
                note_textBox2.Text.Contains("'"))
            {
                MessageBox.Show($"Unclosed quotation mark after the character ( ' )!!", "Unsuccessful Operation",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return true;
            }
            return false;
        }

        /// <summary>
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
```

```
        private void AddUser_Validating(object sender, CancelEventArgs e)
        {
            userName_textBox.Text = userName_textBox.Text.Trim();
            password_textBox.Text = password_textBox.Text.Trim();
            isAdmain_comboBox.Text = isAdmain_comboBox.Text.Trim();
            note_textBox2.Text = note_textBox2.Text.Trim();
        }

        /*
         * these two events help to make good Effects when cursor mouse enter or leave  Add_Employee_button boundaries
         */
        private void Add_Employee_button_MouseEnter(object sender, EventArgs e)
        {
            Add_User_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
            Add_User_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
        }
        private void Add_Employee_button_MouseLeave(object sender, EventArgs e)
        {
            Add_User_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Add_User_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.7 BillServicingCar.cs

```
using System;
using System.Windows.Forms;
using CarServiceLibrary;

namespace Car_Service
{
    public partial class BillServicingCar : Form
    {
        readonly DataGridViewRow row;//to save the received row (client's data)
        readonly string billSer;//text JSON serialization of tools array (tools which was used for this client)
```

```
public BillServicingCar(DataGridViewRow row, string billSer)
{
    InitializeComponent();//F12 to see the functions of this method
    this.billSer = billSer;
    this.row = row;
    FillData();
    FillDataGridView();
}

/// <summary>
/// deserialization tools's array and fill data into dataGridView (table)
/// </summary>
private void FillDataGridView()
{
    double totalAmount = 0;
    Tool[] tools = JSONserialize.Deserialize(billSer);

    for (int i = 0; i < tools.Length; i++)
    {
        totalAmount += tools[i].PurchasePrice * tools[i].Quantity;
        bill_list_dataGridView.Rows.Add(tools[i].ToolName.ToString(), tools[i].ToolUnit.ToString(),
            tools[i].PurchasePrice.ToString(), tools[i].Quantity.ToString());

    }
    total_textBox2.Text = $"{totalAmount:0.00}";
}

/// <summary>
/// fill client's information in labels
/// </summary>
private void FillData()
{
    id_label2.Text += $"  {row.Cells[0].Value}";
```

```
            fName_label.Text += $"  {row.Cells[1].Value}";
            sName_label3.Text += $"  {row.Cells[2].Value}";
            car_number_label4.Text += $"  {row.Cells[3].Value}";
            CarType_label2.Text += $"  {row.Cells[4].Value}";
            carColor_label3.Text += $"  {row.Cells[5].Value}";
            carModel_label4.Text += $"  {row.Cells[6].Value}";
            phone_label2.Text += $"  {row.Cells[7].Value}";
            adress_label.Text += $"  {row.Cells[8].Value}";
            entry_label.Text += $"  {row.Cells[9].Value}";
            /*
             * these two conditions becuase {row} can be data from CarRegister database or CarDelivery database
             * there is difference in one variabale
             */
            if (row.Cells.Count > 12)
            {
                exit_label7.Text += $"  {row.Cells[10].Value}";
                identification_label.Text += $"  {row.Cells[11].Value}";
                note_textBox1.Text += row.Cells[12].Value.ToString();

            }
            else
            {
                exit_label7.Text += $"  {DateTime.Now}";
                identification_label.Text += $"  {row.Cells[10].Value}";
                note_textBox1.Text += row.Cells[11].Value.ToString();
            }

        }
    }
}
```

3.8 BuyingProcess.cs

```
using System;
using System.Collections;
```

```
using System.Data;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;
using System.Data.SqlClient;

namespace Car_Service
{
    public partial class BuyingProcess : Form
    {
        readonly DataTable purchases;
        readonly DataTable alldata, baddata;//baddata represent tools which have shortage in their quantity.
        //constructer
        public BuyingProcess()
        {
            InitializeComponent();//F12 to see the functions of this method
            try
            {
                ReadClass toolsData = new ReadClass("Tools");
                alldata = toolsData.GetDataTable();//get all tools data
                baddata = toolsData.GetBadTools();//get bad tools data
            }
            catch (SqlException)
            {
                MessageBox.Show($"couldn't get the database",
                    "Unsuccessful operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                Close();
            }
            purchases = alldata.Copy();
            purchases.Rows.Clear();
            dataToolsGridView.DataSource = baddata;
        }

        /// <summary>
```

```
/// click button event to add tool with specific quantity to the list of purchases
/// </summary>
private void AddTool_button_Click(object sender, EventArgs e)
{
   if (quantity_numericUpDown.Value == 0)
      MessageBox.Show($"Please select the quantity of the tool!!",
         "Quantity Is Zero", MessageBoxButtons.OK, MessageBoxIcon.Warning);
   else
   {
      var dataRow = dataToolsGridView.SelectedRows;
      //if user already added this tool
      if (HasBuyingListRow(dataRow))
      {
         MessageBox.Show($"You have just add this tool!", "Already Exist",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
         quantity_numericUpDown.Value = 0;
         return;
      }
      ArrayList row = new ArrayList
      {
         dataRow[0].Cells[1].Value.ToString(),
         dataRow[0].Cells[2].Value.ToString(),
         dataRow[0].Cells[4].Value.ToString(),
         quantity_numericUpDown.Value.ToString()
      };
      //add tool to purchases list
      Buying_list_dataGridView.Rows.Add(row.ToArray());
      FixTotal();
      string[] temp = new string[dataRow[0].Cells.Count];
      for (int i = 0; i < dataRow[0].Cells.Count; i++)
      {
         temp[i] = dataRow[0].Cells[i].Value.ToString();
      }
      purchases.Rows.Add(temp);
```

```
    }
    quantity_numericUpDown.Value = 0;
}

/// <summary>
/// change total amount of tools price  when add(delete) to(from) purchases list
/// </summary>
private void FixTotal()
{
    double total = 0;
    for (int i = 0; i < Buying_list_dataGridView.Rows.Count; i++)
    {
        total += double.Parse(Buying_list_dataGridView.Rows[i].Cells[2].Value.ToString()) *
            double.Parse(Buying_list_dataGridView.Rows[i].Cells[3].Value.ToString());
    }
    total_textBox1.Text = total.ToString();
}

/// <summary>
/// check if user has already added specific tool to purchases list
/// </summary>
/// <param name="dataRow">tool which need to check it</param>
/// <returns>true if exit in purchases list, false if not</returns>
private bool HasBuyingListRow(DataGridViewSelectedRowCollection dataRow)
{
    for (int i = 0; i < Buying_list_dataGridView.Rows.Count; i++)
    {
        if (Buying_list_dataGridView.Rows[i].Cells[0].Value.ToString() == dataRow[0].Cells[1].Value.ToString() &&
            Buying_list_dataGridView.Rows[i].Cells[1].Value.ToString() == dataRow[0].Cells[2].Value.ToString())
            return true;
    }
    return false;
}
```

```
/// <summary>
/// event to delete tool from purchases list when click on Delete me button exists in purchases table
/// </summary>
private void Buying_list_dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
   if (e.ColumnIndex == 5)
   {
      Buying_list_dataGridView.Rows.RemoveAt(e.RowIndex);
      FixTotal();
   }
}

/// <summary>
/// button click event to move to SendOrder form where order will be sent to some company
/// </summary>
private void Buy_button_Click(object sender, EventArgs e)
{
   if (Buying_list_dataGridView.Rows.Count > 0)
   {
      SendOrder order = new SendOrder(Buying_list_dataGridView);
      order.Show();
   }
}

DataTable temp = new DataTable();
readonly ToolsDatabase tools = new ToolsDatabase();

/// <summary>
/// event fire when search about somedata into DataGridView
/// </summary>
private void ToolUnit_textBox1_TextChanged(object sender, EventArgs e)
{
   if (!serName_textBox1.Text.Contains("'") && !toolUnit_textBox1.Text.Contains("'"))
   {
```

```
        temp = tools.SearchFor(serName_textBox1.Text, toolUnit_textBox1.Text);
        dataToolsGridView.DataSource = temp;
    }
}

/*
 * these 4 events help to make good Effects when cursor mouse enter or leave  AddTool_button and Buy_button boundaries
 */
private void AddTool_MouseEnter(object sender, EventArgs e)
{
    AddTool.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    AddTool.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
}
private void AddTool_MouseLeave(object sender, EventArgs e)
{
    AddTool.FlatAppearance.BorderColor = System.Drawing.Color.White;
    AddTool.ForeColor = System.Drawing.Color.White;
}
private void Buy_button_MouseEnter(object sender, EventArgs e)
{
    Buy_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Buy_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);

}
private void Buy_button_MouseLeave(object sender, EventArgs e)
{
    Buy_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Buy_button.ForeColor = System.Drawing.Color.White;
}

/// <summary>
/// event of checkBox tool to show all tools or only tools which have shortage in their quantity
/// </summary>
private void Show_all_tools_checkBox_CheckedChanged(object sender, EventArgs e)
```

```
        {
            if (alldata == baddata)
                MessageBox.Show("true");
            if (Show_all_tools_checkBox.Checked)
                dataToolsGridView.DataSource = alldata;
            else
                dataToolsGridView.DataSource = baddata;
            dataToolsGridView.Refresh();
        }
    }
}
```

3.9 CarRegisters.cs

```
using System;
using System.Data;
using System.Windows.Forms;
using CarServiceLibrary;
using System.Data.SqlClient;

namespace Car_Service
{
    public partial class CarRegisters : Form
    {
        DataTable table = new DataTable();
        //constrocter
        public CarRegisters()
        {
            InitializeComponent();//F12 to see the functions of this method
            DownloadToolsData();
        }
        /// <summary>
        /// download registered cars ino DataGridView( dataToolsGridView)
        /// </summary>
```

```
private void DownloadToolsData()
{
    ReadClass read = new ReadClass("CarRegister");
    table = read.GetDataTable();
    var topLeftHeaderCell = dataToolsGridView.TopLeftHeaderCell;
    if (table != null && topLeftHeaderCell != null)
        dataToolsGridView.DataSource = table;
}

/// <summary>
/// click button event move user to ChoiceUsedTools form to choose
/// the tools which were used in order to repair the car
/// </summary>
private void Repair_button_Click(object sender, EventArgs e)
{
    if (dataToolsGridView.SelectedRows.Count > 0)
    {
        this.Visible = false;
        this.Close();
        ChoiceUsedTools tools = new ChoiceUsedTools(dataToolsGridView.SelectedRows[0]);
        tools.ShowDialog();
    }
}

/// <summary>
/// click button event to delete selected car register
/// </summary>
private void Delete_button1_Click(object sender, EventArgs e)
{
    if (dataToolsGridView.SelectedRows.Count > 0)
        if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this car register?", "Delete",
          MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
        {
            try
```

```
            {
                DeleteFromDatabase registerdelete = new DeleteFromDatabase();
                registerdelete.DeleteRow("CarRegister", dataToolsGridView.SelectedRows[0].Cells[0].Value.ToString());
                dataToolsGridView.Rows.RemoveAt(dataToolsGridView.SelectedRows[0].Index);
            }
            catch (SqlException)
            {
                MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }

        }
    }

    /*
     * these 4 events help to make good Effects when cursor mouse enter or leave Repair_button and Delete_button1
boundaries
     */
    private void Repair_button_MouseEnter(object sender, EventArgs e)
    {
        Repair_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
        Repair_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    }
    private void Repair_button_MouseLeave(object sender, EventArgs e)
    {
        Repair_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
        Repair_button.ForeColor = System.Drawing.Color.White;
    }
    private void Delete_button1_MouseEnter(object sender, EventArgs e)
    {
        Delete_button1.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
        Delete_button1.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    }
    private void Delete_button1_MouseLeave(object sender, EventArgs e)
```

```
        {
            Delete_button1.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Delete_button1.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.10 ChoiceUsedTools.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class ChoiceUsedTools : Form
    {
        DataTable toolsTable = new DataTable();
        readonly DataGridViewRow order;//to save recevide information of car register
        readonly ToolsDatabase toolsDatabase = new ToolsDatabase();

        //constrocter
        public ChoiceUsedTools(DataGridViewRow order)
        {
            InitializeComponent();//F12 to see the functions of this method
            DownloadToolsData();
            this.order = order;
        }
```

```
/// <summary>
/// get tools data from database and show it in toolsDataToolsGridView(Tebale)
/// </summary>
private void DownloadToolsData()
{
    try
    {
        ReadClass read = new ReadClass("Tools");
        toolsTable = read.GetDataTable();
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    var topLeftHeaderCell = toolsDataToolsGridView.TopLeftHeaderCell;
    if (toolsTable != null && topLeftHeaderCell != null)
        toolsDataToolsGridView.DataSource = toolsTable;
}

/// <summary>
/// click button event to add tool with specific quantity to the list of purchases
/// </summary>
private void AddTool_Click(object sender, EventArgs e)
{
    var dataRow = toolsDataToolsGridView.SelectedRows;
    if (quantity_numericUpDown.Value == 0)
        MessageBox.Show($"Please select the quantity of the tool!!", "Quantity Is Zero",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);

    else if (quantity_numericUpDown.Value > (int)dataRow[0].Cells[5].Value)
        MessageBox.Show($"the quantity of this tool is not available!!", "Unsuccessfully Operation",
```

```
                MessageBoxButtons.OK, MessageBoxIcon.Warning);


            else
            {
                //if user already added this tool
                if (HasBuyingListRow(dataRow))
                {

                    MessageBox.Show($"You have just added this tool!", "Already Exist",
                        MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    quantity_numericUpDown.Value = 0;
                    return;

                }
                ArrayList row = new ArrayList
                {
                    dataRow[0].Cells[1].Value.ToString(),
                    dataRow[0].Cells[2].Value.ToString(),
                    dataRow[0].Cells[3].Value.ToString(),
                    quantity_numericUpDown.Value.ToString()
                };
                //add tool to sales table
                selling_list_dataGridView.Rows.Add(row.ToArray());
                toolsDataToolsGridView.SelectedRows[0].Cells[5].Value =
(int)toolsDataToolsGridView.SelectedRows[0].Cells[5].Value
                    - (int)quantity_numericUpDown.Value;
            }
            quantity_numericUpDown.Value = 0;
        }

        /// <summary>
        /// check if user has already added specific tool to sales table
        /// </summary>
        /// <param name="dataRow">tool which need to check it</param>
```

```
/// <returns>true if exit in sales table, false if not</returns>
private bool HasBuyingListRow(DataGridViewSelectedRowCollection dataRow)
{
   for (int i = 0; i < selling_list_dataGridView.Rows.Count; i++)
   {
      if (selling_list_dataGridView.Rows[i].Cells[0].Value.ToString() == dataRow[0].Cells[1].Value.ToString() &&
         selling_list_dataGridView.Rows[i].Cells[1].Value.ToString() == dataRow[0].Cells[2].Value.ToString())
         return true;
   }
   return false;
}

/// <summary>
/// event to delete tool from sales table when click on Delete me button exists in sales table
/// </summary>
private void Selling_list_dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
   if (e.ColumnIndex == 5 && selling_list_dataGridView.SelectedRows.Count != 0)
   {
      FixQuantityInDataTools(e.RowIndex);

      selling_list_dataGridView.Rows.RemoveAt(e.RowIndex);
   }
}

/// <summary>
/// this method help to fix the quantity of tools when add or delete it to(from) sales table
/// </summary>
/// <param name="rowIndex"> index row where quantity was changed</param>
private void FixQuantityInDataTools(int rowIndex)
{
   int index = (from r in toolsDataToolsGridView.Rows.Cast<DataGridViewRow>()
                where r.Cells[1].Value == selling_list_dataGridView.Rows[rowIndex].Cells[0].Value
                where r.Cells[2].Value == selling_list_dataGridView.Rows[rowIndex].Cells[1].Value
```

```
            select r.Index).First();
    int quantityInDataTools = (int)toolsDataToolsGridView.Rows[index].Cells[5].Value;
    int quantityInSellingTools = int.Parse(selling_list_dataGridView.Rows[rowIndex].Cells[3].Value.ToString());
    toolsDataToolsGridView.Rows[index].Cells[5].Value = quantityInDataTools + quantityInSellingTools;
}

/// <summary>
/// save delivery data / delete this register from CarRegister database becuase it saved already
/// in CarDelivery database / move to BillServicingCar form
/// </summary>
private void GetInvoice_button_Click(object sender, EventArgs e)
{
    if (selling_list_dataGridView.Rows.Count > 0)
    {
        string billSerial = GetSerialize();
        List<string> rowDeliveryRegister = new List<string>() {order.Cells[1].Value.ToString(),
            order.Cells[2].Value.ToString(), order.Cells[3].Value.ToString(), order.Cells[4].Value.ToString(),
            order.Cells[5].Value.ToString(), order.Cells[6].Value.ToString(),order.Cells[7].Value.ToString(),
            order.Cells[8].Value.ToString(), order.Cells[9].Value.ToString(),DateTime.Now.ToString(),
            order.Cells[10].Value.ToString(),note_textBox1.Text,billSerial};
        try
        {
            CarDeliveryDatabase delivery = new CarDeliveryDatabase();
            //save delivery data
            delivery.AddData(rowDeliveryRegister);
            DeleteFromDatabase write = new DeleteFromDatabase();
            //delete this register from CarRegister database
            write.DeleteRow("CarRegister", order.Cells[0].Value.ToString());
            this.Close();
            order.Cells[11].Value = note_textBox1.Text;
            //move to BillServicingCar form
            BillServicingCar bill = new BillServicingCar(order, billSer: billSerial);
            bill.ShowDialog();
        }
```

```
        catch (SqlException)
        {
            MessageBox.Show($"couldn't get the database",
           "Unsuccessful operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Close();
        }
    }
    else
        MessageBox.Show($"You should choose at least one tool!!", "Empty List",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
}

/// <summary>
/// get string of tools array which was used for client from JSON serialization
/// </summary>
/// <returns>serialized tools array </returns>
private string GetSerialize()
{
    Tool[] tools = new Tool[selling_list_dataGridView.Rows.Count];
    for (int i = 0; i < selling_list_dataGridView.Rows.Count; i++)
    {
        tools[i] = new Tool(selling_list_dataGridView.Rows[i]);
    }
    SaveToolsIntoInvoice(tools);
    string getStringJSON = JSONserialize.Serialize(tools);
    return getStringJSON;
}



/// <summary>
/// save saled tools in specefic database
/// </summary>
/// <param name="tools"> array of saled tools</param>
private void SaveToolsIntoInvoice(Tool[] tools)
```

```
{
    Invoice invoice = new Invoice("SalesInvoice");
    List<string> row;
    for (int i = 0; i < tools.Length; i++)
    {
        row = new List<string>() { tools[i].ToolName, tools[i].ToolUnit, (tools[i].Quantity).ToString(),
            (tools[i].Quantity * tools[i].PurchasePrice).ToString(), DateTime.Now.ToString() };
        invoice.AddData(row);
        toolsDatabase.EditQuantity(tools[i].ToolName, tools[i].ToolUnit, -(int)tools[i].Quantity);
    }
}

DataTable temp = new DataTable();
/// <summary>
/// event of searching operation
/// </summary>
private void ToolUnit_textBox1_TextChanged(object sender, EventArgs e)
{
    if (!serName_textBox1.Text.Contains("") && !toolUnit_textBox1.Text.Contains(""))
    {
        temp = toolsDatabase.SearchFor(serName_textBox1.Text, toolUnit_textBox1.Text);
        toolsDataToolsGridView.DataSource = temp;
    }
}

/*
 *  these 4 events help to make good Effects when cursor mouse enter or leave
 *  AddTool_button and GetInvoice_button boundaries
 */
private void AddTool_MouseEnter(object sender, EventArgs e)
{
    AddTool.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    AddTool.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
}
```

```
        private void AddTool_MouseLeave(object sender, EventArgs e)
        {
            AddTool.FlatAppearance.BorderColor = System.Drawing.Color.White;
            AddTool.ForeColor = System.Drawing.Color.White;
        }
        private void GetInvoice_button_MouseEnter(object sender, EventArgs e)
        {
            GetInvoice_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
            GetInvoice_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
        }
        private void GetInvoice_button_MouseLeave(object sender, EventArgs e)
        {
            GetInvoice_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            GetInvoice_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.11 History.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class History : Form
    {
        DataTable sales = new DataTable();
```

```
//constroctor
public History()
{
   InitializeComponent();//F12 to see the functions of this method
   DownloadToolsData();
}
/// <summary>
/// get CarDelivery database into salesGridView (table)
/// </summary>
private void DownloadToolsData()
{
   try
   {
      ReadClass read = new ReadClass("CarDelivery");
      sales = read.GetDataTable();
      sales.Columns.RemoveAt(13);
      if (sales != null)
         salesGridView.DataSource = sales;
   }
   catch (SqlException)
   {
      MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
         MessageBoxButtons.OK, MessageBoxIcon.Warning);
      Close();
   }
   catch (ArgumentException ex)
   {
      MessageBox.Show($"{ex.Message}", "Unsuccessfully Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
      Close();
   }
}

/// <summary>
```

```
/// double click event to get the invoice of Car Delivery register
/// </summary>
private void SalesGridView_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)
{
   try
   {
      CarDeliveryDatabase CarDelivery = new CarDeliveryDatabase();
      BillServicingCar bill = new BillServicingCar(salesGridView.SelectedRows[0],
            CarDelivery.GetBillRow(int.Parse(salesGridView.SelectedRows[0].Cells[0].Value.ToString())));
      bill.ShowDialog();
   }
   catch (Exception) { }
}

/// <summary>
/// check Box event to allow searching by date
/// </summary>
private void On_off_checkBox_CheckedChanged(object sender, EventArgs e)
{
   if (On_off_checkBox.Checked)
   {
      from_dateTimePicker1.Enabled = true;
      to_dateTimePicker2.Enabled = true;

   }
   else
   {
      from_dateTimePicker1.Value = DateTime.Parse("1/25/1900 11:59 PM");
      to_dateTimePicker2.Value = DateTime.Parse("12/25/2099 11:59 PM");
      from_dateTimePicker1.Enabled = false;
      to_dateTimePicker2.Enabled = false;
   }
}
```

```
/// <summary>
/// event of searching operation
/// </summary>
private void History_TextChanged(object sender, EventArgs e)
{
   GetData();
}

/// <summary>
/// event fire when user enter data in search boxes
/// </summary>
private void From_dateTimePicker1_ValueChanged(object sender, EventArgs e)
{
   GetData();
}

/// <summary>
/// Display the required data for searching operation
/// </summary>
DataTable temp = new DataTable();
private void GetData()
{
   List<string> clientSearch = new List<string>() {fName_textBox.Text,
      sName_textBox.Text,carNumber_textBox.Text,type_textBox1.Text,color_comboBox1.Text,
      model_comboBox2.Text};
   if (clientSearch.All<string>(c => !c.Contains("")))
   {
      //calling method SearchFor which located in CarDeliveryDatabase class
      try
      {
         CarDeliveryDatabase CarDelivery = new CarDeliveryDatabase();
         temp = CarDelivery.SearchFor(clientSearch, from_dateTimePicker1.Value, to_dateTimePicker2.Value);
         temp.Columns.RemoveAt(13);
         salesGridView.DataSource = temp;
```

```
        }
        catch (SqlException) { }
    }
}

/// <summary>
/// button click event to delete car delivery register from database
/// </summary>
private void Delete_button_Click(object sender, EventArgs e)
{
    if (salesGridView.SelectedRows.Count > 0)
        if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this invoice ?", "Delete",
            MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
        {
            DeleteFromDatabase tooldelete = new DeleteFromDatabase();
            tooldelete.DeleteRow("CarDelivery", salesGridView.SelectedRows[0].Cells[0].Value.ToString());
            salesGridView.Rows.RemoveAt(salesGridView.SelectedRows[0].Index);
        }
}

/*
 * these two events help to make good Effects when cursor mouse enter or leave
 *  Delete_button boundaries
 */
private void Delete_button_MouseEnter(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
    Delete_button.ForeColor = System.Drawing.Color.Red;
}
private void Delete_button_MouseLeave(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Delete_button.ForeColor = System.Drawing.Color.White;
}
```

```
    }
}
```

3.12 HistoryC.cs

```
using System;
using System.Windows.Forms;

namespace Car_Service
{
    //Inherited class from Windows.forms.UserControl class
    public partial class HistoryC : UserControl
    {
        bool isAdmain;

        //constrocter
        public HistoryC()
        {
            InitializeComponent();//F12 to see the functions of this method
        }

        public bool IsAdmain { get => isAdmain; set => isAdmain = value; }

        /// <summary>
        /// click on pic event, to move user to PurchasesHistory form
        /// </summary>
        private void ShowPurchases_pictureBox2_Click(object sender, EventArgs e)
        {
            if (isAdmain)
            {
                PurchasesHistory purchases = new PurchasesHistory();
                purchases.ShowDialog();
            }
            else
```

```
            MessageBox.Show($"You are not an admain, you can't see this!!",
              "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        /// <summary>
        /// click on pic event, to move user to History form
        /// </summary>
        private void ShowSeles_pictureBox1_Click(object sender, EventArgs e)
        {
            History history = new History();
            history.ShowDialog();
        }
    }
}
```

3.13 InterfaceForm.cs

```
using System;
using System.Drawing;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;

namespace Car_Service
{
    public partial class InterfaceForm : Form
    {
        private const bool f = false;

        //if login user is admain
        readonly bool isAdmain = f;
        /*
         * these 5 bool variables help in making efficte on buttons while moving between controles
         */
```

```
bool pressedShowEdit = false;
bool pressedAdd = false;
bool pressedOperation = false;
bool pressedRegister = false;
bool pressedHistory = false;

/// <summary>
/// Constrocter
/// </summary>
/// <param name="isAdmain"> check if login user is admain</param>
public InterfaceForm(bool isAdmain)
{
   this.isAdmain = isAdmain;
   InitializeComponent();//F12 to see the functions of this method
   Warning();
   if (shortage_GridView.Rows.Count > 0)
      panel5.BackColor = Color.Red;
   shortage_GridView.Visible = false;
   Up_pictureBox6.Visible = false;
   showEditDelete1.IsAdmain = isAdmain;
   add1.IsAdmain = isAdmain;
   historyC1.IsAdmain = isAdmain;
   operationsC1.IsAdmain = isAdmain;
}
/// <summary>
/// load form event
/// </summary>
private void InterfaceForm_Load(object sender, EventArgs e)
{
   /*
    * the panels where controles locate
    */
   main_panel5.Show();
   showEditDelete1.Hide();
```

```
    add1.Hide();
    operationsC1.Hide();
    historyC1.Hide();

}

/// <summary>
/// getting tools which have shortage
/// </summary>
private void Warning()
{
    try
    {
        ReadClass check = new ReadClass("Tools");
        shortage_GridView.DataSource = check.CheckTools();
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
}

/// <summary>
/// click on pic event to close the form
/// </summary>
private void PictureBox3_Click(object sender, EventArgs e)
{
    Close();
}

/// <summary>
/// click on pic event to minimize the form
```

```
/// </summary>
private void PictureBox4_Click(object sender, EventArgs e)
{
    WindowState = FormWindowState.Minimized;
}

/// <summary>
/// event move user to AddCarRegister form
/// </summary>
private void RegisterCar_button1_Click(object sender, EventArgs e)
{
    #region help in design
    ChangeBorderColorAndForeColor();
    RegisterCar_button1.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    RegisterCar_button1.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Pressed("register");
    main_panel5.Show();
    showEditDelete1.Hide();
    add1.Hide();
    operationsC1.Hide();
    historyC1.Hide();
    #endregion
    AddCarRegister addCar = new AddCarRegister();
    addCar.ShowDialog();
}

/// <summary>
/// click button event to  show panel were locate ShowEditDelete user control
/// </summary>
private void ShowEditDelete_button_Click(object sender, EventArgs e)
{
    //hide other user controls
    add1.Hide();
    operationsC1.Hide();
```

```
    historyC1.Hide();
    //show current user control
    showEditDelete1.Show();
    showEditDelete1.BringToFront();
    ChangeBorderColorAndForeColor();
    ShowEditDelete_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    ShowEditDelete_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Pressed("show");
}

/// <summary>
/// click button event to show panel were locate Add user control
/// </summary>
private void AddData_button_Click(object sender, EventArgs e)
{
    //hide other user controls
    showEditDelete1.Hide();
    operationsC1.Hide();
    historyC1.Hide();
    //show current user control
    add1.Show();
    add1.BringToFront();
    ChangeBorderColorAndForeColor();
    AddData_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    AddData_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Pressed("add");
}

/// <summary>
/// click button event to show panel were locate OperationC user control
/// </summary>
private void Operations_button_Click(object sender, EventArgs e)
{
```

```
    //hide other user controls
    showEditDelete1.Hide();
    add1.Hide();
    historyC1.Hide();
    //show current user control
    operationsC1.Show();
    operationsC1.BringToFront();
    ChangeBorderColorAndForeColor();
    Operations_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Operations_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Pressed("operation");
}

/// <summary>
/// click button event to show panel were locate HistoryC user control
/// </summary>
private void History_button_Click(object sender, EventArgs e)
{
    //hide other user controls
    showEditDelete1.Hide();
    add1.Hide();
    operationsC1.Hide();
    //show current user control
    historyC1.Show();
    historyC1.BringToFront();
    ChangeBorderColorAndForeColor();
    History_button.FlatAppearance.BorderColor = System.Drawing.Color.FromArgb(229, 126, 49);
    History_button.ForeColor = System.Drawing.Color.FromArgb(229, 126, 49);
    Pressed("history");
}

/// <summary>
/// click pic event to back to main panel
/// </summary>
```

```
private void PictureBox5_Click(object sender, EventArgs e)
{
    main_panel5.Show();
    showEditDelete1.Hide();
    add1.Hide();
    operationsC1.Hide();
    historyC1.Hide();
    ChangeBorderColorAndForeColor();
    pressedShowEdit = pressedRegister = pressedAdd = pressedOperation = pressedHistory = false;
}

/// <summary>
/// this method help to get required design and effects
/// </summary>
private void Pressed(string nameButton)
{
    switch (nameButton)
    {
        case "show":
            pressedShowEdit = true;
            pressedRegister = pressedAdd = pressedOperation = pressedHistory = false;
            break;
        case "add":
            pressedAdd = true;
            pressedRegister = pressedShowEdit = pressedOperation = pressedHistory = false;
            break;
        case "operation":
            pressedOperation = true;
            pressedRegister = pressedShowEdit = pressedAdd = pressedHistory = false;
            break;
        case "register":
            pressedRegister = true;
            pressedAdd = pressedShowEdit = pressedOperation = pressedHistory = false;
            break;
```

```
        case "history":
            pressedHistory = true;
            pressedRegister = pressedShowEdit = pressedOperation = pressedAdd = false;
            break;
        default:
            break;
    }
}

/// <summary>
/// reset the color of buttons
/// </summary>
private void ChangeBorderColorAndForeColor()
{
    History_button.FlatAppearance.BorderColor = Color.White;
    History_button.ForeColor = Color.White;
    RegisterCar_button1.FlatAppearance.BorderColor = Color.White;
    RegisterCar_button1.ForeColor = Color.White;
    Operations_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Operations_button.ForeColor = Color.White;
    AddData_button.FlatAppearance.BorderColor = Color.White;
    AddData_button.ForeColor = Color.White;
    ShowEditDelete_button.FlatAppearance.BorderColor = Color.White;
    ShowEditDelete_button.ForeColor = Color.White;
}

/// <summary>
/// click pic event help to hide list of tools which have shortage
/// </summary>
private void Up_pictureBox6_Click(object sender, EventArgs e)
{
    shortage_GridView.Visible = false;
    Up_pictureBox6.Visible = false;
    Down_pictureBox7.Visible = true;
```

```
}

/// <summary>
/// click pic event help to show list of tools which have shortage
/// </summary>
private void Down_pictureBox7_Click(object sender, EventArgs e)
{
   shortage_GridView.Visible = true;
   Up_pictureBox6.Visible = true;
   Down_pictureBox7.Visible = false;
}



/*
 * these 10 events help to make good Effects when cursor mouse enter or leave buttons boundaries
 */
private void ShowEditDelete_button_MouseEnter(object sender, EventArgs e)
{
   ShowEditDelete_button.FlatAppearance.BorderColor = Color.FromArgb(229, 126, 49);
   ShowEditDelete_button.ForeColor = Color.FromArgb(229, 126, 49);
}
private void ShowEditDelete_button_MouseLeave(object sender, EventArgs e)
{
   if (!pressedShowEdit)
   {
      ShowEditDelete_button.FlatAppearance.BorderColor = Color.White;
      ShowEditDelete_button.ForeColor = Color.White;
   }
}
private void AddData_button_MouseEnter(object sender, EventArgs e)
{
   AddData_button.FlatAppearance.BorderColor = Color.FromArgb(229, 126, 49);
   AddData_button.ForeColor = Color.FromArgb(229, 126, 49);
}
```

```
private void AddData_button_MouseLeave(object sender, EventArgs e)
{
   if (!pressedAdd)
   {
      AddData_button.FlatAppearance.BorderColor = Color.White;
      AddData_button.ForeColor = Color.White;
   }

}
private void Operations_button_MouseEnter(object sender, EventArgs e)
{
   Operations_button.FlatAppearance.BorderColor = Color.FromArgb(229, 126, 49);
   Operations_button.ForeColor = Color.FromArgb(229, 126, 49);
}
private void Operations_button_MouseLeave(object sender, EventArgs e)
{
   if (!pressedOperation)
   {
      Operations_button.FlatAppearance.BorderColor = Color.White;
      Operations_button.ForeColor = Color.White;
   }
}
private void RegisterCar_button1_MouseEnter(object sender, EventArgs e)
{
   RegisterCar_button1.FlatAppearance.BorderColor = Color.FromArgb(229, 126, 49);
   RegisterCar_button1.ForeColor = Color.FromArgb(229, 126, 49);
}
private void RegisterCar_button1_MouseLeave(object sender, EventArgs e)
{
   if (!pressedRegister)
   {
      RegisterCar_button1.FlatAppearance.BorderColor = Color.White;
      RegisterCar_button1.ForeColor = Color.White;
   }
```

```
        }
        private void History_button_MouseEnter(object sender, EventArgs e)
        {
            History_button.FlatAppearance.BorderColor = Color.FromArgb(229, 126, 49);
            History_button.ForeColor = Color.FromArgb(229, 126, 49);
        }
        private void History_button_MouseLeave(object sender, EventArgs e)
        {
            if (!pressedHistory)
            {
                History_button.FlatAppearance.BorderColor = Color.White;
                History_button.ForeColor = Color.White;
            }
        }


    }
}
```

3.14 MainForm.cs

```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.IO;

namespace Car_Service
{
    public partial class MainForm : Form
    {
        readonly FileInfo file;//help in getting path of database
        static string constring;
        readonly SqlConnection Sql;
        /// <summary>
        /// constrocter
```

```
        /// </summary>
        public MainForm()
        {
            InitializeComponent();//F12 to see the functions of this method
        // string[] filePaths = Directory.GetFiles(@"../../");
        // string s = null;
        // for (int i = 0; i < filePaths.Length; i++)
        // {
        //     s += filePaths[i] + "\n";
        // }
        // MessageBox.Show(s);
            if (!File.Exists(@"../../CarsDatabase.mdf"))
            {
                MessageBox.Show(@"Database Was not found, Please copy it here (Course work\Car_Service)!", "No Connection",
              MessageBoxButtons.OK, MessageBoxIcon.Warning);
                Close();
            }

            file = new FileInfo(@"../../CarsDatabase.mdf");
            constring = $@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename={file.FullName};Integrated
Security=True";
            Sql = new SqlConnection(constring);
        }

        /// <summary>
        /// click button event check the input data and determine if
        /// user admain or note if pass and user name was correct
        /// and move user to InterfaceForm
        /// </summary>
        private void Logain_button_Click(object sender, EventArgs e)
        {
            if (CheckInput())
                return;
            string is_admain = null;
```

```
    if (DetermineUser(ref is_admain))
    {
        InterfaceForm form = new InterfaceForm(bool.Parse(is_admain));
        form.ShowDialog();
    }

}
/// <summary>
/// method help to Determine user the program
/// </summary>
/// <param name="is_admain"> the state if user(admain or not)</param>
/// <returns>true is data user exist in database, false if not</returns>
private bool DetermineUser(ref string is_admain)
{
    try
    {
        Sql.Open();
        SqlCommand cmd = new SqlCommand("select * from users where User_name='" + username_textBox.Text +
            "' and Password ='" + password_textBox.Text + "'", Sql);//instruction to get this data from database if it exists
        SqlDataReader rd = cmd.ExecuteReader();
        if (rd.HasRows)
        {
            while (rd.Read())
            {
                if (username_textBox.Text == rd.GetValue(1).ToString())
                {
                    is_admain = rd.GetValue(3).ToString();
                }
            }
            Sql.Close();
            username_textBox.Text = "";
            password_textBox.Text = "";
            return true;
        }
```

```
        else
        {

            MessageBox.Show("incorrect username or password!", "Unsuccessful Operation",
          MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Sql.Close();
            return false;
        }
    }
    catch (SqlException)
    {
        MessageBox.Show("incorrect username or password!", "Unsuccessful Operation",
          MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Sql.Close();
    }
    return false;

}

/// <summary>
/// method to check if input data has <'> symbol
/// </summary>
private bool CheckInput()
{
    if (username_textBox.Text.Contains("'") || password_textBox.Text.Contains("'"))
    {
        MessageBox.Show("You cannot enter this symbol <'>!", "Not Allowed Entry",
          MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
```

```
        /// click on pic event to close the form
        /// </summary>
        private void Close_pictureBox3_Click(object sender, EventArgs e)
        {
            if (DialogResult.Yes == MessageBox.Show("Are you sure you want to exit?", "EXIT",
              MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
               this.Close();
        }

        /// <summary>
        /// click on pic event to minimize the form
        /// </summary>
        private void Minimize_pictureBox4_Click(object sender, EventArgs e)
        {
            WindowState = FormWindowState.Minimized;
        }

        /*
         * these 2 events help to make good Effects when cursor mouse enter or leave Logain_button boundaries
         */
        private void Logain_button_MouseEnter(object sender, EventArgs e)
        {
            Logain_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
            Logain_button.ForeColor = System.Drawing.Color.DarkGreen;
        }
        private void Logain_button_MouseLeave(object sender, EventArgs e)
        {
            Logain_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Logain_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.15 OperationsC.cs

```
using System;
using System.Windows.Forms;

namespace Car_Service
{
    //Inherited class from Windows.forms.UserControl class
    public partial class OperationsC : UserControl
    {
        bool isAdmain;

        public bool IsAdmain { get => isAdmain; set => isAdmain = value; }

        //constrocter
        public OperationsC()
        {
            InitializeComponent();//F12 to see the functions of this method
        }

        /// <summary>
        /// click on pic event, to move user to BuyingProcess form
        /// </summary>
        private void Order_pictureBox2_Click(object sender, EventArgs e)
        {
            BuyingProcess buying = new BuyingProcess();
            buying.Show();
        }

        /// <summary>
        /// click on pic event, to move user to Reports form
        /// </summary>
        private void Report_pictureBox1_Click(object sender, EventArgs e)
        {
```

```
            if (isAdmain)
            {
                try
                {
                    Reports reports = new Reports();
                    reports.ShowDialog();
                }
                catch (InvalidOperationException ex)
                {
                    MessageBox.Show($"{ex.Message}!!", "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
            }
            else
                MessageBox.Show($"You are not an admain, so you can't add a user!!",
                  "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        /// <summary>
        /// click on pic event, to move user to CarRegisters form
        /// </summary>
        private void Repair_pictureBox1_Click(object sender, EventArgs e)
        {
            CarRegisters delivery = new CarRegisters();
            delivery.ShowDialog();
        }
    }
}
```

3.16 Program.cs

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

```
namespace Car_Service
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Thread.CurrentThread.CurrentCulture = new CultureInfo("en-US");
         Application.Run(new MainForm());
      }
   }
}
```

3.17 PurchaseInvoice.cs

```
using System.Windows.Forms;
using CarServiceLibrary;

namespace Car_Service
{
   public partial class PurchaseInvoice : Form
   {
      readonly DataGridViewRow row;//to save received data invoice
      readonly string billSer;//text JSON serialization of tools array (tools which was bought from specefic company)
      public PurchaseInvoice(DataGridViewRow row, string billSer)
      {
         InitializeComponent();
         this.billSer = billSer;//F12 to see the functions of this method
         this.row = row;
```

```
    FillData();
    FillDataGridView();
}

/// <summary>
/// deserialization tools's array and fill data into dataGridView (table)
/// </summary>
private void FillDataGridView()
{
    double totalAmount = 0;

    Tool[] tools = JSONserialize.Deserialize(billSer);
    for (int i = 0; i < tools.Length; i++)
    {
        totalAmount += tools[i].PurchasePrice * tools[i].Quantity;
        bill_list_dataGridView.Rows.Add(tools[i].ToolName.ToString(), tools[i].ToolUnit.ToString(),
            tools[i].PurchasePrice.ToString(), tools[i].Quantity.ToString());

    }
    total_textBox2.Text = $"{totalAmount:0.00}";
}

/// <summary>
/// fill client's information in labels
/// </summary>
private void FillData()
{
    id_label2.Text += $"  {row.Cells[0].Value}";
    compName_label.Text += $"  {row.Cells[1].Value}";
    compOwnerlabel2.Text += $"  {row.Cells[2].Value}";
    phone_label2.Text += $"  {row.Cells[3].Value}";
    adress_label.Text += $"  {row.Cells[4].Value}";
    email_label3.Text += $"  {row.Cells[5].Value}";
    date_label.Text += $"  {row.Cells[6].Value}";
```

```
        }
    }
}
```

3.18 PurchasesHistory.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class PurchasesHistory : Form
    {
        DataTable temp = new DataTable();
        DataTable purchases = new DataTable();
        readonly ShoppingDatabase purchase = new ShoppingDatabase();
        //constroctor
        public PurchasesHistory()
        {
            InitializeComponent();//F12 to see the functions of this method
            DownloadToolsData();
        }
        /// <summary>
        /// get CarDelivery database into salesGridView (table)
        /// </summary>
        private void DownloadToolsData()
        {
            try
            {
                ReadClass read = new ReadClass("Shopping");
```

```
        purchases = read.GetDataTable();
        purchases.Columns.RemoveAt(7);
        if (purchases != null)
            purchasesGridView.DataSource = purchases;
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessfully Operation",
          MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
}
/// <summary>
/// double click event to get the invoice of shopping register
/// </summary>
private void SalesGridView_MouseDoubleClick(object sender, MouseEventArgs e)
{
    try
    {
        ShoppingDatabase purchase = new ShoppingDatabase();
        PurchaseInvoice invoice = new PurchaseInvoice(purchasesGridView.SelectedRows[0],
            purchase.GetBillRow(int.Parse(purchasesGridView.SelectedRows[0].Cells[0].Value.ToString())));
        invoice.ShowDialog();
    }
    catch (Exception) { }
}

/// <summary>
```

```
/// check Box event to allow searching by date
/// </summary>
private void On_off_checkBox_CheckedChanged(object sender, EventArgs e)
{
   if (On_off_checkBox.Checked)
   {
      from_dateTimePicker1.Enabled = true;
      to_dateTimePicker2.Enabled = true;
   }
   else
   {
      from_dateTimePicker1.Value = DateTime.Parse("1/25/1900 11:59 PM");
      to_dateTimePicker2.Value = DateTime.Parse("12/25/2099 11:59 PM");
      from_dateTimePicker1.Enabled = false;
      to_dateTimePicker2.Enabled = false;
   }
}

/// <summary>
/// event of searching operation
/// </summary>
private void To_dateTimePicker2_ValueChanged(object sender, EventArgs e)
{
   temp = purchase.SearchFor(serName_textBox.Text, ownerName_textBox1.Text,
      from_dateTimePicker1.Value, to_dateTimePicker2.Value);
   temp.Columns.RemoveAt(7);
   purchasesGridView.DataSource = temp;
}

/// <summary>
/// event of searching operation
/// </summary>
private void OwnerName_textBox1_TextChanged(object sender, EventArgs e)
{
```

```
    if (!serName_textBox.Text.Contains("") && !ownerName_textBox1.Text.Contains(""))
    {
        temp = purchase.SearchFor(serName_textBox.Text, ownerName_textBox1.Text,
            from_dateTimePicker1.Value, to_dateTimePicker2.Value);
        temp.Columns.RemoveAt(7);
        purchasesGridView.DataSource = temp;
    }
}

/// <summary>
/// button click event to delete purchase register from database
/// </summary>
private void Delete_button_Click(object sender, EventArgs e)
{
    if (purchasesGridView.SelectedRows.Count > 0)
        if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this invoice?",
            "Deleting Operation", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
        {
            DeleteFromDatabase tooldelete = new DeleteFromDatabase();
            tooldelete.DeleteRow("Shopping", purchasesGridView.SelectedRows[0].Cells[0].Value.ToString());
            purchasesGridView.Rows.RemoveAt(purchasesGridView.SelectedRows[0].Index);
        }
}

/*
* these two events help to make good Effects when cursor mouse enter or leave
*  Delete_button boundaries
*/
private void Delete_button_MouseEnter(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
    Delete_button.ForeColor = System.Drawing.Color.Red;
}
private void Delete_button_MouseLeave(object sender, EventArgs e)
```

```
        {
            Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Delete_button.ForeColor = System.Drawing.Color.White;

        }
    }
}
```

3.19 Reports.cs

```
using System;
using System.Data;
using System.Linq;
using System.Data.SqlClient;
using System.Windows.Forms;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class Reports : Form
    {
        readonly EmployeesDatabase employees = new EmployeesDatabase();//help in getting Wages of staffs
        readonly ToolsDatabase tools = new ToolsDatabase();
        Invoice invoice = new Invoice();
        double staffWages;

        //constrocter
        public Reports()
        {
            InitializeComponent();//F12 to see the functions of this method
            try
            {
                staffWages = employees.GetStaffWages();
                InitializePayment();
                InitializeSales();
```

```
}
catch (SqlException)
{
    MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Close();
}
catch (ArgumentNullException ex)
{
    MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Close();
}
catch (NullReferenceException ex)
{
    MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Close();
}
catch (InvalidCastException ex)
{
    MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Close();
}
catch (IndexOutOfRangeException ex)
{
    MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    Close();
}
catch (Exception ex)
{
    MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
```

```
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            Close();
        }
    }
    /// <summary>
    /// fill sales data.
    /// </summary>
    private void InitializeSales()
    {
        invoice = new Invoice();
        DataTable sales = invoice.GetforReport(from_dateTimePicker1.Value, to_dateTimePicker2.Value, "SalesInvoice");
        if (sales.Rows.Count > 0)
        {
            sales_dataToolsGridView.DataSource = GetAfterGroup(sales);//group the similer data together
            DataTable table = (DataTable)sales_dataToolsGridView.DataSource;
            double sum = table.AsEnumerable().Sum(r => r.Field<double>("Total_Amount"));//get sum of Total_Amount in all
invoices
            int sumtools = table.AsEnumerable().Sum(r => r.Field<int>("Quantity"));//get sum of Quantity of used tools in all
invoices
            soldTools_label.Text = $"Sold Tools:  {sum:0.00}";
            numOfSold_label2.Text = $"Number of sold Tools:  {sumtools}";
        }
        else
        {
            sales_dataToolsGridView.DataSource = sales;
            soldTools_label.Text = $"Sold Tools:";
            numOfSold_label2.Text = $"Number of sold Tools:";
        }
        double totalProfit = 0;
        for (int i = 0; i < sales_dataToolsGridView.Rows.Count; i++)
        {
            totalProfit += double.Parse(sales_dataToolsGridView.Rows[i].Cells[3].Value.ToString()) *
                tools.GetPurchasesPrice(sales_dataToolsGridView.Rows[i].Cells[1].Value.ToString(),
                sales_dataToolsGridView.Rows[i].Cells[2].Value.ToString());
```

```
        }
        totalProfit_label4.Text = $"Total Profits:  {totalProfit}";
    }

    /// <summary>
    /// fill payment data.
    /// </summary>
    private void InitializePayment()
    {
        /*
         * set staffs wages
         */
        invoice = new Invoice();
        double months = (to_dateTimePicker2.Value.Month + to_dateTimePicker2.Value.Year * 12) -
            (from_dateTimePicker1.Value.Month + from_dateTimePicker1.Value.Year * 12);
        staffWages_label3.Text = $"Staff Wages:  {(months * staffWages):0.00}";

        DataTable payments = invoice.GetforReport(from_dateTimePicker1.Value, to_dateTimePicker2.Value,
"PurchasesInvoice");
        if (payments.Rows.Count > 0)
        {
            purchases_dataGridView.DataSource = GetAfterGroup(payments);//group the similer data together
            DataTable table = (DataTable)purchases_dataGridView.DataSource;
            double sum = table.AsEnumerable().Sum(r => r.Field<double>("Total_Amount"));//get sum of Total_Amount in all
invoices
            int sumtools = table.AsEnumerable().Sum(r => r.Field<int>("Quantity"));//get sum of Quantity of bougth tools in all
invoices
            purchasesTools_label.Text = $"Purchased Tools:  {sum:0.00}";
            numOfPurchasesTools_label3.Text = $"Number of Purchased Tools:  {sumtools}";
        }
        else
        {
            purchases_dataGridView.DataSource = payments;
```

```
        purchasesTools_label.Text = $"Purchased Tools:";
        numOfPurchasesTools_label3.Text = $"Number of Purchased Tools:";
    }
}

/// <summary>
/// method to group datatable
/// </summary>
/// <param name="table"> table befor grouping</param>
/// <returns>table after grouping it </returns>
private DataTable GetAfterGroup(DataTable table)
{
    table = table.AsEnumerable()
     .GroupBy(r => new { Col1 = r["Tool_Name"], Col2 = r["Tool_Unit"] })
     /*
      * make anonymous opject every different register
      */
     .Select(g =>
     {
         var row = table.NewRow();
         row["ID"] = g.Min(r => r.Field<int>("Id"));
         row["Tool_Name"] = g.Key.Col1;
         row["Tool_Unit"] = g.Key.Col2;
         row["Quantity"] = g.Sum(r => r.Field<int>("Quantity"));
         row["Total_Amount"] = g.Sum(r => r.Field<double>("Total_Amount"));
         row["Date"] = g.Min(r => r.Field<DateTime>("Date"));
         return row;
     })
     .CopyToDataTable();

    return table;
}

/// <summary>
```

```
        /// event will be fired when data will change
        /// </summary>
        private void To_dateTimePicker2_ValueChanged(object sender, EventArgs e)
        {
            staffWages = employees.GetStaffWages();
            InitializePayment();
            InitializeSales();
        }
    }
}
```

3.20 SendOrder.cs

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class SendOrder : Form
    {
        static DataGridView order;
        static ReadClass read;
        readonly ToolsDatabase toolsDatabase = new ToolsDatabase();
        Dictionary<string, string> companyNames;
        /// <summary>
        /// constrocter
        /// </summary>
        /// <param name="grid">the data which will be ordered</param>
        public SendOrder(DataGridView grid)
        {
```

```
    InitializeComponent();//F12 to see the functions of this method
    try
    {
        read = new ReadClass("Companies");
        order = grid;
        FillCompanyListComboBox();
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentNullException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
}

/// <summary>
/// download companies from database into Companies ListComboBox
/// </summary>
private void FillCompanyListComboBox()
{
    companyNames = read.GetCompaniesNames();
    foreach (var name in companyNames)
```

```
    {
        company_list_comboBox.Items.Add(name.Key);
    }
}

/// <summary>
/// click button event to send order to company by entering required data
/// </summary>
private void Send_button_Click(object sender, EventArgs e)
{
    /*
     * delegate with two methods which will be called back if sending operation was done in send email class
     */
    Del del = SavePurchases;
    del += InitializingBoxes;

    if (!int.TryParse(port_textBox.Text, out int port) && port <= 0)
    {
        MessageBox.Show($"The value of port is invalid!", "Invalid Value",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    if (CheckInput() && !HasInvalidChar() && companyNames.ContainsKey(company_list_comboBox.Text.ToString()))
    {
        SendEmail sending = new SendEmail(userName_textBox.Text, password_textBox.Text, message_textBox.Text,
            companyNames[company_list_comboBox.Text.ToString()], subject_textBox.Text, smtp_textBox.Text,
            port, ssl_checkBox.Checked, order);
        sending.Send(del);
    }

}
/// <summary>
/// cjecking input
/// </summary>
```

```
private bool HasInvalidChar()
{
   if (userName_textBox.Text.Contains(",") || userName_textBox.Text.Contains("'") ||
       password_textBox.Text.Contains(",") || password_textBox.Text.Contains("'"))
   {
      MessageBox.Show($"there is invalid character(<'>|<,>) in some box !", "Invalid Value",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
      return true;
   }
   return false;
}

/// <summary>
/// checking input data
/// </summary>
private bool CheckInput()
{
   if (string.IsNullOrEmpty(userName_textBox.Text.Trim()) || string.IsNullOrEmpty(password_textBox.Text.Trim())
      || company_list_comboBox.SelectedItem == null)
   {
      MessageBox.Show($"you have forgot to fill some main information!!", "Invalid operation",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
      return false;
   }
   return true;
}

/// <summary>
/// saving the bougth tools into ShoppingDatabase
/// </summary>
private void SavePurchases()
{

   ShoppingDatabase writer = new ShoppingDatabase();
```

```
    List<string> row = writer.GetSpecificRow(company_list_comboBox.SelectedItem.ToString(), "Companies");
    int length = row.Count;
    List<string> shoppingRow = new List<string>();
    for (int i = 0; i < length - 1; i++)
    {
        shoppingRow.Add(row[i]);
    }
    shoppingRow.Add(DateTime.Now.ToString());
    shoppingRow.Add(GetSerialize());
    writer.AddData("Shopping", inputrow: shoppingRow);
}

/// <summary>
/// Serialization of tools array
/// </summary>
private string GetSerialize()
{
    Tool[] tools = new Tool[order.Rows.Count];
    for (int i = 0; i < order.Rows.Count; i++)
    {
        tools[i] = new Tool(order.Rows[i]);
    }
    SaveToolsIntoInvoice(tools);
    string getStringJSON = JSONserialize.Serialize(tools);
    return getStringJSON;
}

/// <summary>
/// save data of bougth tools into specefic database, which help in showing report operation
/// </summary>
private void SaveToolsIntoInvoice(Tool[] tools)
{
    Invoice invoice = new Invoice("PurchasesInvoice");
    List<string> row;
```

```
    for (int i = 0; i < tools.Length; i++)
    {
        row = new List<string>() { tools[i].ToolName, tools[i].ToolUnit, (tools[i].Quantity).ToString(),
            (tools[i].Quantity * tools[i].PurchasePrice).ToString(), DateTime.Now.ToString() };
        invoice.AddData(row);
        toolsDatabase.EditQuantity(tools[i].ToolName, tools[i].ToolUnit, (int)tools[i].Quantity);
    }
}

/*
 * these two events help to make good Effects when cursor mouse enter or leave
 * Send_button boundaries
 */
private void Send_button_MouseEnter(object sender, EventArgs e)
{
    Send_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
    Send_button.ForeColor = System.Drawing.Color.DarkGreen;

}
private void Send_button_MouseLeave(object sender, EventArgs e)
{
    Send_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Send_button.ForeColor = System.Drawing.Color.White;
}

/// <summary>
/// clean boxes from previous input
/// </summary>
private void InitializingBoxes()
{
    subject_textBox.Text = "";
    message_textBox.Text = "";
    userName_textBox.Text = "";
    password_textBox.Text = "";
```

```
            company_list_comboBox.Text = "Choice Comapny";
        }
    }
}
```

3.21 ShowEditCompanies.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using DataBasesLibrary;
using CarServiceLibrary;



namespace Car_Service
{
    public partial class ShowEditCompanies : Form
    {
        DataTable table = new DataTable();
        readonly CompaniesDataBase companiesData = new CompaniesDataBase();
        /// <summary>
        /// constrocter
        /// </summary>
        public ShowEditCompanies()
        {
            InitializeComponent();//F12 to see the functions of this mehtod
            DownloadToolsData();
        }


        /// <summary>
        /// download data of Companies into dataCompaniesGridView(table)
```

```
/// </summary>
private void DownloadToolsData()
{
    try
    {
        ReadClass read = new ReadClass("Companies");
        table = read.GetDataTable();
        if (table != null)
            dataCompaniesGridView.DataSource = table;
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentNullException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
}

/// <summary>
/// Make Enabled of boxes true and fill the boxes by data of selected row in table
/// </summary>
private void EnabledAndFillBoxes()
```

```
        {
            if (table != null)
            {
                //Make Enabled of boxes true
                id_textBox.Enabled = true;
                companyName_textBox.Enabled = true;
                companyOwner_textBox.Enabled = true;
                phoneNumber_textBox.Enabled = true;
                adress_textBox.Enabled = true;
                note_textBox.Enabled = true;
                email_textBox1.Enabled = true;
                //fill the boxes by data of selected row in table
                id_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[0].Value.ToString();
                companyName_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[1].Value.ToString();
                companyOwner_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[2].Value.ToString();
                phoneNumber_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[3].Value.ToString();
                adress_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[4].Value.ToString();
                email_textBox1.Text = dataCompaniesGridView.SelectedRows[0].Cells[5].Value.ToString();
                entryDate_dateTimePicker.Value =
DateTime.Parse(dataCompaniesGridView.SelectedRows[0].Cells[6].Value.ToString());
                note_textBox.Text = dataCompaniesGridView.SelectedRows[0].Cells[7].Value.ToString();
            }

        }

        /// <summary>
        /// click button event to edit selected company data
        /// </summary>
        private void Edit_button_Click(object sender, EventArgs e)
        {
            try
            {
                if (ISFullData() && IsValidEmail() && CheckPhoneNumber() && !Exist())
                {
```

```
            dataCompaniesGridView.SelectedRows[0].Cells[0].Value = id_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[1].Value = companyName_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[2].Value = companyOwner_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[3].Value = phoneNumber_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[4].Value = adress_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[7].Value = note_textBox.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[5].Value = email_textBox1.Text;
            dataCompaniesGridView.SelectedRows[0].Cells[6].Value = entryDate_dateTimePicker.Value;

            List<string> row = new List<string>() { id_textBox.Text, companyName_textBox.Text,
            companyOwner_textBox.Text,phoneNumber_textBox.Text, adress_textBox.Text,
            email_textBox1.Text,entryDate_dateTimePicker.Value.ToString(),
            note_textBox.Text};

            CompaniesDataBase companyEditor = new CompaniesDataBase();
            companyEditor.EditRow(row);
            MessageBox.Show($"The company information was edited!!", "successful Operation",
               MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
    }
    catch (SqlException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

/// <summary>
/// check if company's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
```

```
    CompaniesDataBase _companiesData = new CompaniesDataBase();
    if (_companiesData.GetSpecificRow(companyName_textBox.Text, companyOwner_textBox.Text).Count > 0)
    {
        MessageBox.Show($"This Company already exist in you database!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// make sure if phonenumber box contain only possitive numbers
/// </summary>
/// <returns>false if phonenumber box contain char or negative number</returns>
private bool CheckPhoneNumber()
{
    if (uint.TryParse(phoneNumber_textBox.Text, out _))
        return true;
    MessageBox.Show($"Phone Number box can't contain a char or negative number!!",
            "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// check if email address input is valid or not
/// </summary>
bool IsValidEmail()
{
    bool methodresult = false;
    try
    {

        var addr = new System.Net.Mail.MailAddress(email_textBox1.Text);
        methodresult = addr.Address == email_textBox1.Text;
```

```
            }
            catch (Exception ex)
            {
                MessageBox.Show($"{ex.Message}, please enter another one!!",
                    "Invalid Email Adress", MessageBoxButtons.OK, MessageBoxIcon.Warning);

            }
            return methodresult;
        }

        /// <summary>
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool ISFullData()
        {
            if (!string.IsNullOrEmpty(companyName_textBox.Text.Trim()) &&
!string.IsNullOrEmpty(companyOwner_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(adress_textBox.Text.Trim()) && !string.IsNullOrEmpty(email_textBox1.Text.Trim()))
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
        private void CompanyName_textBox_Validating(object sender, CancelEventArgs e)
        {
            companyName_textBox.Text = companyName_textBox.Text.Trim();
            companyOwner_textBox.Text = companyOwner_textBox.Text.Trim();
            phoneNumber_textBox.Text = phoneNumber_textBox.Text.Trim();
            adress_textBox.Text = adress_textBox.Text.Trim();
```

```
    email_textBox1.Text = email_textBox1.Text.Trim();
    note_textBox.Text = note_textBox.Text.Trim();
}

/// <summary>
/// click button event to delete specefic data company
/// </summary>
private void Delete_button_Click(object sender, EventArgs e)
{

    if (dataCompaniesGridView.SelectedRows.Count > 0)
        if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this Company ?", "Delete",
       MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
        {
            DeleteFromDatabase tooldelete = new DeleteFromDatabase();
            tooldelete.DeleteRow("Companies", dataCompaniesGridView.SelectedRows[0].Cells[0].Value.ToString());
            dataCompaniesGridView.Rows.RemoveAt(dataCompaniesGridView.SelectedRows[0].Index);
        }
}

/// <summary>
/// cell click event
/// </summary>
private void DataToolsGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
    EnabledAndFillBoxes();
}



/// <summary>
/// searching operation
/// </summary>
private void SerName_textBox1_TextChanged(object sender, EventArgs e)
```

```
{
    DataTable temp = new DataTable();
    CompaniesDataBase companies = new CompaniesDataBase();
    if (!serName_textBox1.Text.Contains("") && !serOwner_textBox1.Text.Contains(""))
    {
        temp = companies.SearchFor(serName_textBox1.Text, serOwner_textBox1.Text);
        dataCompaniesGridView.DataSource = temp;
    }

}
/*
 * these 4 events help to make good Effects when cursor mouse enter or leave
 * Delete_button and Edit_button boundaries
 */
private void Delete_button_MouseEnter(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
    Delete_button.ForeColor = System.Drawing.Color.Red;
}
private void Delete_button_MouseLeave(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Delete_button.ForeColor = System.Drawing.Color.White;
}
private void Edit_button_MouseEnter(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
    Edit_button.ForeColor = System.Drawing.Color.DarkGreen;
}
private void Edit_button_MouseLeave(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Edit_button.ForeColor = System.Drawing.Color.White;
}
```

```
    }
}
```

3.22 ShowEditDelete.cs

```
using System;
using System.Windows.Forms;

namespace Car_Service
{
    //Inherited class from Windows.forms.UserControl class
    public partial class ShowEditDelete : UserControl
    {
        bool isAdmain;
        public bool IsAdmain { get => isAdmain; set => isAdmain = value; }

        //constrocter
        public ShowEditDelete()
        {
            InitializeComponent();//F12 to see the functions of this method
        }

        /// <summary>
        /// click on pic event, to move user to ShowEditTools form
        /// </summary>
        private void ShowTool_pictureBox2_Click(object sender, EventArgs e)
        {
            ShowEditTools editor = new ShowEditTools();
            editor.ShowDialog();
        }

        /// <summary>
        /// click on pic event, to move user to ShowEditCompanies form
        /// </summary>
```

```
        private void ShowCompany_pictureBox3_Click(object sender, EventArgs e)
        {
            ShowEditCompanies editCompanies = new ShowEditCompanies();
            editCompanies.ShowDialog();
        }

        /// <summary>
        /// click on pic event, to move user to ShowEditUsers form if he is admain
        /// </summary>
        private void ShowUser_pictureBox1_Click(object sender, EventArgs e)
        {
            if (IsAdmain)
            {
                ShowEditUsers showEditUsers = new ShowEditUsers();
                showEditUsers.ShowDialog();
            }
            else
                MessageBox.Show($"You are not an admain, so you can't add a user!!",
                    "Invalid Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);

        }

        /// <summary>
        /// click on pic event, to move user to ShowEditEmployees form
        /// </summary>
        private void ShowEmployee_pictureBox4_Click(object sender, EventArgs e)
        {
            ShowEditEmployees employeeEditor = new ShowEditEmployees();
            employeeEditor.ShowDialog();
        }

    }
}
```

3.23 ShowEditEmployees.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class ShowEditEmployees : Form
    {
        DataTable temp = new DataTable();
        DataTable table = new DataTable();
        readonly EmployeesDatabase employeesDatabase = new EmployeesDatabase();

        /// <summary>
        /// constrocter
        /// </summary>
        public ShowEditEmployees()
        {
            InitializeComponent();//F12 to see the functions of this mehtod
            DownloadToolsData();
        }

        /// <summary>
        /// download data of Staffs into dataEmployeesGridView(table)
        /// </summary>
        private void DownloadToolsData()
        {
            try
```

```
    {
        ReadClass read = new ReadClass("Staffs");
        table = read.GetDataTable();
        var topLeftHeaderCell = dataEmployeesGridView.TopLeftHeaderCell;
        if (table != null && topLeftHeaderCell != null)
            dataEmployeesGridView.DataSource = table;
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentNullException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
}

/// <summary>
/// Make Enabled of boxes true and fill the boxes by data of selected row in table
/// </summary>
private void EnabledAndFillBoxes()
{
    if (table != null)
    {
```

```
            //Make Enabled of boxes true
            id_textBox.Enabled = true;
            employeeName_textBox.Enabled = true;
            adress_textBox.Enabled = true;
            phoneNumber_textBox.Enabled = true;
            career_textBox.Enabled = true;
            salary_textBox.Enabled = true;
            note_textBox.Enabled = true;
            //fill the boxes by data of selected row in table
            id_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[0].Value.ToString();
            employeeName_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[1].Value.ToString();
            adress_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[2].Value.ToString();
            phoneNumber_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[3].Value.ToString();
            career_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[4].Value.ToString();
            salary_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[5].Value.ToString();
            register_dateTimePicker.Value =
DateTime.Parse(dataEmployeesGridView.SelectedRows[0].Cells[6].Value.ToString());
            note_textBox.Text = dataEmployeesGridView.SelectedRows[0].Cells[7].Value.ToString();
        }
    }

    /// <summary>
    /// click button event to edit selected employee's data
    /// </summary>
    private void Edit_button_Click(object sender, EventArgs e)
    {
        try
        {
            if (ISFullData() && CheckPhoneNumber() && !Exist())
            {
                dataEmployeesGridView.SelectedRows[0].Cells[0].Value = id_textBox.Text;
                dataEmployeesGridView.SelectedRows[0].Cells[1].Value = employeeName_textBox.Text;
                dataEmployeesGridView.SelectedRows[0].Cells[2].Value = adress_textBox.Text;
                dataEmployeesGridView.SelectedRows[0].Cells[3].Value = phoneNumber_textBox.Text;
```

```
            dataEmployeesGridView.SelectedRows[0].Cells[4].Value = career_textBox.Text;
            dataEmployeesGridView.SelectedRows[0].Cells[5].Value = salary_textBox.Text;
            dataEmployeesGridView.SelectedRows[0].Cells[6].Value = register_dateTimePicker.Value;
            dataEmployeesGridView.SelectedRows[0].Cells[7].Value = note_textBox.Text;
            List<string> row = new List<string>() { id_textBox.Text, employeeName_textBox.Text,
            adress_textBox.Text,phoneNumber_textBox.Text, career_textBox.Text,
            salary_textBox.Text,register_dateTimePicker.Value.ToString(),
            note_textBox.Text};

            EmployeesDatabase employee = new EmployeesDatabase();
            employee.EditRow(row);
            MessageBox.Show($"The Employee information was edited!!", "successful Operation",
               MessageBoxButtons.OK, MessageBoxIcon.Information);
         }
      }
      catch (SqlException ex)
      {
         MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
      }

   }

   /// <summary>
   /// check if employee's input data already exists in database
   /// </summary>
   /// <returns>true if input data already exists in database, false if not</returns>
   private bool Exist()
   {
      EmployeesDatabase _employeesDatabase = new EmployeesDatabase();
      if (_employeesDatabase.GetSpecificRow(employeeName_textBox.Text).Count > 0)
      {
         MessageBox.Show($"This Employee's name already exist in you database, Please choose another one!!",
"Unsuccessful Operation",
```

```
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// make sure if phonenumber box contain only possitive numbers
/// </summary>
/// <returns>false if phonenumber box contain char or negative number</returns>
private bool CheckPhoneNumber()
{
    if (uint.TryParse(phoneNumber_textBox.Text, out _))
        return true;
    MessageBox.Show($"Phone Number box can't contain a char or negative number!!",
            "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// click button event to delete specefic employee's data
/// </summary>
private void Delete_button_Click(object sender, EventArgs e)
{
    if (dataEmployeesGridView.SelectedRows.Count > 0)
        if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this employee ?", "Delete",
      MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
        {
            DeleteFromDatabase tooldelete = new DeleteFromDatabase();
            tooldelete.DeleteRow("Staffs", dataEmployeesGridView.SelectedRows[0].Cells[0].Value.ToString());
            dataEmployeesGridView.Rows.RemoveAt(dataEmployeesGridView.SelectedRows[0].Index);
        }

}
```

```
        /// <summary>
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
        private void Note_textBox_Validating(object sender, CancelEventArgs e)
        {
            employeeName_textBox.Text = employeeName_textBox.Text.Trim();
            adress_textBox.Text = adress_textBox.Text.Trim();
            phoneNumber_textBox.Text = phoneNumber_textBox.Text.Trim();
            career_textBox.Text = career_textBox.Text.Trim();
            salary_textBox.Text = salary_textBox.Text.Trim();
            note_textBox.Text = note_textBox.Text.Trim();

        }

        /// <summary>
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool ISFullData()
        {
            if (!string.IsNullOrEmpty(employeeName_textBox.Text.Trim()) &&
!string.IsNullOrEmpty(phoneNumber_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(adress_textBox.Text.Trim()) && !string.IsNullOrEmpty(career_textBox.Text.Trim()) &&
                !string.IsNullOrEmpty(salary_textBox.Text.Trim()))
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// cell click event
        /// </summary>
```

```
private void DataToolsGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
    EnabledAndFillBoxes();
}

/// <summary>
/// event of searching operation
/// </summary>
private void SerName_textBox1_TextChanged(object sender, EventArgs e)
{
    if (!serName_textBox1.Text.Contains(""))
    {
        temp = employeesDatabase.SearchFor(serName_textBox1.Text);
        dataEmployeesGridView.DataSource = temp;
    }
}

/*
 * these 4 events help to make good Effects when cursor mouse enter or leave
 * Delete_button and Edit_button boundaries
 */
private void Edit_button_MouseEnter(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
    Edit_button.ForeColor = System.Drawing.Color.DarkGreen;
}
private void Edit_button_MouseLeave(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Edit_button.ForeColor = System.Drawing.Color.White;
}
private void Delete_button_MouseEnter(object sender, EventArgs e)
{
    Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
```

```
            Delete_button.ForeColor = System.Drawing.Color.Red;
        }
        private void Delete_button_MouseLeave(object sender, EventArgs e)
        {
            Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Delete_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.24 ShowEditTools.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class ShowEditTools : Form
    {
        DataTable temp = new DataTable();
        DataTable table = new DataTable();
        readonly ToolsDatabase toolsDatabase = new ToolsDatabase();

        /// <summary>
        /// constrocter
        /// </summary>
        public ShowEditTools()
        {
```

```
    InitializeComponent();//F12 to see the functions of this mehtod
    DownloadToolsData();
}

/// <summary>
/// download data of tools into dataToolsGridView(table)
/// </summary>
private void DownloadToolsData()
{
    try
    {
        ReadClass read = new ReadClass("Tools");
        table = read.GetDataTable();
        if (table != null)
            dataToolsGridView.DataSource = table;
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentNullException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
```

```
}

/// <summary>
/// Make Enabled of boxes true and fill the boxes by data of selected row in table
/// </summary>
private void EnabledAndFillBoxes()
{
   if (table != null)
   {
      // Make Enabled of boxes true
      name_textBox.Enabled = true;
      unit_textBox.Enabled = true;
      sellPrice_textBox.Enabled = true;
      purchasePrice_textBox.Enabled = true;
      note_textBox.Enabled = true;
      current_numericUpDown.Enabled = true;
      min_numericUpDown.Enabled = true;
      max_numericUpDown.Enabled = true;
      //fill the boxes by data of selected row in table
      id_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[0].Value.ToString();
      name_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[1].Value.ToString();
      unit_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[2].Value.ToString();
      sellPrice_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[3].Value.ToString();
      purchasePrice_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[4].Value.ToString();
      note_textBox.Text = dataToolsGridView.SelectedRows[0].Cells[8].Value.ToString();
      current_numericUpDown.Value = int.Parse(dataToolsGridView.SelectedRows[0].Cells[5].Value.ToString());
      min_numericUpDown.Value = int.Parse(dataToolsGridView.SelectedRows[0].Cells[7].Value.ToString());
      max_numericUpDown.Value = int.Parse(dataToolsGridView.SelectedRows[0].Cells[6].Value.ToString());

   }
}

/// <summary>
/// check if Tool's input data already exists in database
```

```
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
   ToolsDatabase toolsDatabase = new ToolsDatabase();
   if (toolsDatabase.GetSpecificRow(name_textBox.Text, unit_textBox.Text).Count > 0)
   {
      MessageBox.Show($"This tool already exist in you database!!", "Unsuccessful Operation",
         MessageBoxButtons.OK, MessageBoxIcon.Warning);
      return true;
   }
   return false;
}

/// <summary>
/// cell click event
/// </summary>
private void DataToolsGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
   EnabledAndFillBoxes();

}

/// <summary>
/// click button event to edit selected tool's data
/// </summary>
private void Edit_button_Click(object sender, EventArgs e)
{
   try
   {
      if (IsFullData() && CheckQuantity() && CheckPrices() && !Exist())
      {

         dataToolsGridView.SelectedRows[0].Cells[0].Value = id_textBox.Text;
```

```
            dataToolsGridView.SelectedRows[0].Cells[1].Value = name_textBox.Text;
            dataToolsGridView.SelectedRows[0].Cells[2].Value = unit_textBox.Text;
            dataToolsGridView.SelectedRows[0].Cells[3].Value = sellPrice_textBox.Text;
            dataToolsGridView.SelectedRows[0].Cells[4].Value = purchasePrice_textBox.Text;
            dataToolsGridView.SelectedRows[0].Cells[8].Value = note_textBox.Text;
            dataToolsGridView.SelectedRows[0].Cells[5].Value = current_numericUpDown.Value;
            dataToolsGridView.SelectedRows[0].Cells[7].Value = min_numericUpDown.Value;
            dataToolsGridView.SelectedRows[0].Cells[6].Value = max_numericUpDown.Value;
            List<string> row = new List<string>() { id_textBox.Text, name_textBox.Text,
            unit_textBox.Text,sellPrice_textBox.Text, purchasePrice_textBox.Text,
            current_numericUpDown.Value.ToString(),max_numericUpDown.Value.ToString(),
            min_numericUpDown.Value.ToString(), note_textBox.Text};

            ToolsDatabase tools = new ToolsDatabase();
            tools.EditRow(row);
            MessageBox.Show($"The tool information was edited!!", "Successful Operation",
               MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
    }catch(SqlException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }

}

/// <summary>
/// check the type and the logical of entering prices
/// </summary>
/// <returns></returns>
private bool CheckPrices()
{
    if (double.TryParse(sellPrice_textBox.Text, out double sell) &&
```

```
            double.TryParse(purchasePrice_textBox.Text, out double purchase))
        {
            if (sell >= purchase)
                return true;
            //if selling price lower than  purchase the user will be asked about that (it might be a typo)
            else if (DialogResult.Yes == MessageBox.Show("Are you sure that you will sell this tool" +
                " at a lower price than the purchase price ?", "Strange Editing",
               MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
                return true;
            return false;
        }

        else
        {
            MessageBox.Show($"Prices boxes can't contain a char!!", "Unsuccessful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }
    }
    /// <summary>
    /// check the logical of tool's quantity input
    /// </summary>
    /// <returns>true for logical input, fasle for not</returns>
    private bool CheckQuantity()
    {
        if (current_numericUpDown.Value <= max_numericUpDown.Value && min_numericUpDown.Value <=
max_numericUpDown.Value)
            return true;
        MessageBox.Show($"The order of quantity is not logical!!", "Unsuccessful Operation", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
        return false;
    }

    /// <summary>
```

```
        /// check if boxes contain input
        /// </summary>
        /// <returns>true for full data, and false for not</returns>
        private bool IsFullData()
        {
            if (!string.IsNullOrEmpty(name_textBox.Text.Trim()) && !string.IsNullOrEmpty(unit_textBox.Text.Trim()))
                return true;
            MessageBox.Show($"You have forgot to fill some information, please check your input!!", "Unsuccessful Operation",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        /// <summary>
        /// event fire after each change on boxes to delete extra space from first and end the input
        /// </summary>
        private void Name_textBox_Validating(object sender, CancelEventArgs e)
        {
            name_textBox.Text = name_textBox.Text.Trim();
            unit_textBox.Text = unit_textBox.Text.Trim();
            sellPrice_textBox.Text = sellPrice_textBox.Text.Trim();
            purchasePrice_textBox.Text = purchasePrice_textBox.Text.Trim();
            note_textBox.Text = note_textBox.Text.Trim();
        }


        /// <summary>
        /// click button event to delete specefic tool's data
        /// </summary>
        private void Delete_button_Click(object sender, EventArgs e)
        {
            if (dataToolsGridView.SelectedRows.Count > 0)
                if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this tool ?", "Delete",
               MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
                {
```

```
            DeleteFromDatabase tooldelete = new DeleteFromDatabase();
            tooldelete.DeleteRow("Tools", dataToolsGridView.SelectedRows[0].Cells[0].Value.ToString());
            dataToolsGridView.Rows.RemoveAt(dataToolsGridView.SelectedRows[0].Index);
        }
}

/// <summary>
/// event of searching operation
/// </summary>
private void ToolUnit_textBox1_TextChanged(object sender, EventArgs e)
{
    if (!serName_textBox1.Text.Contains("") && !toolUnit_textBox1.Text.Contains(""))
    {
        temp = toolsDatabase.SearchFor(serName_textBox1.Text, toolUnit_textBox1.Text);
        dataToolsGridView.DataSource = temp;
    }
}

/*
 * these 4 events help to make good Effects when cursor mouse enter or leave
 * Delete_button and Edit_button boundaries
 */
private void Edit_button_MouseEnter(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
    Edit_button.ForeColor = System.Drawing.Color.DarkGreen;
}
private void Edit_button_MouseLeave(object sender, EventArgs e)
{
    Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
    Edit_button.ForeColor = System.Drawing.Color.White;
}
private void Delete_button_MouseEnter(object sender, EventArgs e)
{
```

```
            Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
            Delete_button.ForeColor = System.Drawing.Color.Red;
        }
        private void Delete_button_MouseLeave(object sender, EventArgs e)
        {
            Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
            Delete_button.ForeColor = System.Drawing.Color.White;
        }
    }
}
```

3.25 ShowEditUsers.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using CarServiceLibrary;
using DataBasesLibrary;

namespace Car_Service
{
    public partial class ShowEditUsers : Form
    {
        DataTable temp = new DataTable();
        DataTable table = new DataTable();
        readonly UserDatabase user = new UserDatabase();

        /// <summary>
        /// constrocter
        /// </summary>
```

```
public ShowEditUsers()
{
    InitializeComponent();
    DownloadToolsData();
}

/// <summary>
/// download data of users into datausersGridView(table)
/// </summary>
private void DownloadToolsData()
{
    try
    {
        ReadClass read = new ReadClass("users");
        table = read.GetDataTable();
        var topLeftHeaderCell = datausersGridView.TopLeftHeaderCell;
        if (table != null && topLeftHeaderCell != null)
            datausersGridView.DataSource = table;
    }
    catch (SqlException)
    {
        MessageBox.Show($"couldn't get the database", "Unsuccessful operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (ArgumentNullException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
```

```
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
      }
    }

    /// <summary>
    /// Make Enabled of boxes true and fill the boxes by data of selected row in table
    /// </summary>
    private void EnabledAndFillBoxes()
    {
      if (table != null)
      {
        // Make Enabled of boxes true
        userName_textBox.Enabled = true;
        password_textBox.Enabled = true;
        isAdmain_comboBox.Enabled = true;
        note_textBox2.Enabled = true;
        //fill the boxes by data of selected row in table
        id_textBox.Text = datausersGridView.SelectedRows[0].Cells[0].Value.ToString();
        userName_textBox.Text = datausersGridView.SelectedRows[0].Cells[1].Value.ToString();
        password_textBox.Text = datausersGridView.SelectedRows[0].Cells[2].Value.ToString();
        isAdmain_comboBox.Text = datausersGridView.SelectedRows[0].Cells[3].Value.ToString();
        note_textBox2.Text = datausersGridView.SelectedRows[0].Cells[4].Value.ToString();
      }
    }

    /// <summary>
    /// check if boxes contain input
    /// </summary>
    /// <returns>true for full data, and false for not</returns>
    private bool ISFullData()
    {
      if (!string.IsNullOrEmpty(userName_textBox.Text.Trim()) && !string.IsNullOrEmpty(password_textBox.Text.Trim())
&&
```

```
        isAdmain_comboBox.SelectedItem != null)
        return true;
    MessageBox.Show($"You have forgot to fill some information, please check your input!!",
        "Unsuccessful Operation", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return false;
}

/// <summary>
/// event fire after each change on boxes to delete extra space from first and end the input
/// </summary>
private void ShowEditUsers_Validating(object sender, CancelEventArgs e)
{
    userName_textBox.Text = userName_textBox.Text.Trim();
    password_textBox.Text = password_textBox.Text.Trim();
    isAdmain_comboBox.Text = isAdmain_comboBox.Text.Trim();
    note_textBox2.Text = note_textBox2.Text.Trim();
}

/// <summary>
/// click button event to edit selected user's data
/// </summary>
private void Edit_button_Click(object sender, EventArgs e)
{
    try
    {
        if (ISFullData() && !Exist())
        {
            datausersGridView.SelectedRows[0].Cells[0].Value = id_textBox.Text;
            datausersGridView.SelectedRows[0].Cells[1].Value = userName_textBox.Text;
            datausersGridView.SelectedRows[0].Cells[2].Value = password_textBox.Text;
            datausersGridView.SelectedRows[0].Cells[3].Value = isAdmain_comboBox.Text;
            datausersGridView.SelectedRows[0].Cells[4].Value = note_textBox2.Text;

            List<string> row = new List<string>() { id_textBox.Text, userName_textBox.Text,
```

```
            password_textBox.Text,isAdmain_comboBox.Text, note_textBox2.Text};
            UserDatabase user = new UserDatabase();
            user.EditRow(row);
            MessageBox.Show($"The User information was edited!!", "successful Operation",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    catch (SqlException ex)
    {
        MessageBox.Show($"{ex.Message}", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

/// <summary>
/// check if user's input data already exists in database
/// </summary>
/// <returns>true if input data already exists in database, false if not</returns>
private bool Exist()
{
    UserDatabase _user = new UserDatabase();
    if (_user.GetSpecificRow(userName_textBox.Text, password_textBox.Text).Count > 0)
    {
        MessageBox.Show($"This user already exist in your database!!", "Unsuccessful Operation",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return true;
    }
    return false;
}

/// <summary>
/// cell click event
/// </summary>
private void DataToolsGridView_CellClick(object sender, DataGridViewCellEventArgs e)
```

```
{
   EnabledAndFillBoxes();
}

/// <summary>
/// click button event to delete specefic user's data
/// </summary>
private void Delete_button_Click(object sender, EventArgs e)
{
   if (datausersGridView.SelectedRows.Count > 0)
      if (DialogResult.Yes == MessageBox.Show("Are you sure you want to delete this user ?", "Delete",
    MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question))
      {
         DeleteFromDatabase tooldelete = new DeleteFromDatabase();
         tooldelete.DeleteRow("users", datausersGridView.SelectedRows[0].Cells[0].Value.ToString());
         datausersGridView.Rows.RemoveAt(datausersGridView.SelectedRows[0].Index);
      }
}

/// <summary>
/// event of searching operation
/// </summary>
private void SerName_textBox1_TextChanged(object sender, EventArgs e)
{

   if (!serName_textBox1.Text.Contains("'"))
   {
      temp = user.SearchFor(serName_textBox1.Text);
      datausersGridView.DataSource = temp;
   }
}

/*
 * these 4 events help to make good Effects when cursor mouse enter or leave
```

```
        * Delete_button and Edit_button boundaries
        */
       private void Edit_button_MouseEnter(object sender, EventArgs e)
       {
          Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.DarkGreen;
          Edit_button.ForeColor = System.Drawing.Color.DarkGreen;
       }
       private void Edit_button_MouseLeave(object sender, EventArgs e)
       {
          Edit_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
          Edit_button.ForeColor = System.Drawing.Color.White;
       }
       private void Delete_button_MouseEnter(object sender, EventArgs e)
       {
          Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
          Delete_button.ForeColor = System.Drawing.Color.Red;
       }
       private void Delete_button_MouseLeave(object sender, EventArgs e)
       {
          Delete_button.FlatAppearance.BorderColor = System.Drawing.Color.White;
          Delete_button.ForeColor = System.Drawing.Color.White;
       }
    }
}
```

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	№ документа	Входящий № сопроводительного документа и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					