Criterion C: Development

The product that's developed is a C# product. It is an inventory management system that has been developed for the client to store various product information for his general store. The program allows the client to add, update, delete, view and search products. The client can also use the program keep track of sales. This program is protected by a login screen that requires simply the user's username and password to access product and sales information.

Libraries Used:

```
1   using System;
2   using System.Collections.Generic;
3   using System.Data;
4   using System.Data.SqlClient;
5   using System.Linq;
6   using System.Text;
7   using System.Threading.Tasks;
8
```

Code 1: Libraries Used

Techniques Used:

- Microsoft Visual Studio IDE has been used.
- The program uses SQL (Structured Query Language) to store product data. SQL Server Management Studio is used to manage and operate the SQL server used for the product.
- .Net framework in C# has been used to develop the application.
- System.drawing NuGet graphic packages are used for the graphical components.
- Static variables and methods are used

Login Page:

Image 1: Login Page

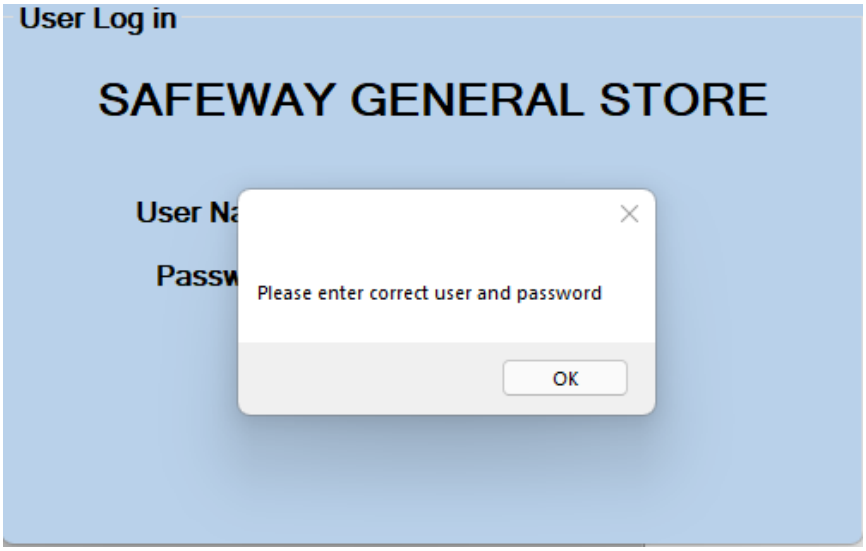


Image 2: Login Error Message

When the user runs the program, a login window will appear. This window requests the user to input their 'username' and 'password'. If the user enters the wrong information, the program will not grant access to the program. This allows the software to be protected from outside interference as any unauthorized user cannot access the program and gain confidential product information.

```
20      private void btnlogin_Click(object sender, EventArgs e)
21      {
22          if(txtpassword.Text != "" && txtuserName.Text!="")
23          {
24              SecUser user=new SecUser();
25              DBHandler repo = new DBHandler();
26
27              user.Password = txtpassword.Text;
28              user.Username = txtuserName.Text;
29              if(repo.LogIn(user))
30              {
31                // Main mainform=new Main();
32                  MainMenu mainform=new MainMenu();
33
34                  mainform.Show();
35                  this.Hide();
36              }
37              else
38              {
39                  MessageBox.Show("Please enter correct user and password");
40              }
41
42
43          }
44          else
45          {
46              MessageBox.Show("Please enter user and password");
47          }
48      }
49
                1 reference
50      private void btnreg_Click(object sender, EventArgs e)
51      {
52          frmNewUser frmNewUserchild = new frmNewUser();
53          // Set the Parent Form of the Child window.
54          //frmCheckoutchild.MdiParent = this;
55          // Display the new form.
56          frmNewUserchild.Show();
57      }
58      //private void txtpassword_TextChanged(object sender, EventArgs e)
59      //{
60
61     // }
62
                1 reference
63      private void txtpassword_TextChanged_1(object sender, EventArgs e)
64      {
65          if (txtpassword.Text != "")
66          {
67              txtpassword.UseSystemPasswordChar = true;
68
69          }
70          else
71          {
72              txtpassword.UseSystemPasswordChar = false;
73          }
```

```
                1 reference
173         public bool LogIn(SecUser parameter)
174         {
175
176             using (SqlConnection con = new SqlConnection(conString))
177             {
178                 byte[] byteArray = new byte[1024];
179                 using (SqlCommand cmd = new SqlCommand("spLogin", con))
180                 {
181                     con.Open();
182                     cmd.CommandType = CommandType.StoredProcedure;
183
184                     cmd.Parameters.Add("@pUserID", SqlDbType.NVarChar, 50).Value = parameter.Username;// parameter.Name;
185                     cmd.Parameters.Add("@pPassword", SqlDbType.NVarChar, 50).Value = parameter.Password;//parameter.UserID;
186
187                     SqlDataAdapter da = new SqlDataAdapter(cmd);
188                     DataTable dt = new DataTable();
189                     da.Fill(dt);
190
191
192                     foreach (DataRow dr in dt.Rows)
193                     {
194                         if (dr["UserName"] != null)
195                         {
196                             if (parameter.Username == (string)dr["UserName"] && parameter.Password == (string)dr["Password"])
197                                 return true;
198
199                         }
200                     }
201
202
203                     return false;
204                 }
205             }
206
207
208         }
209
```

Code 2: Login Page codes along with verification

Main Window:

After entering the correct username and password, the program will open the main window.

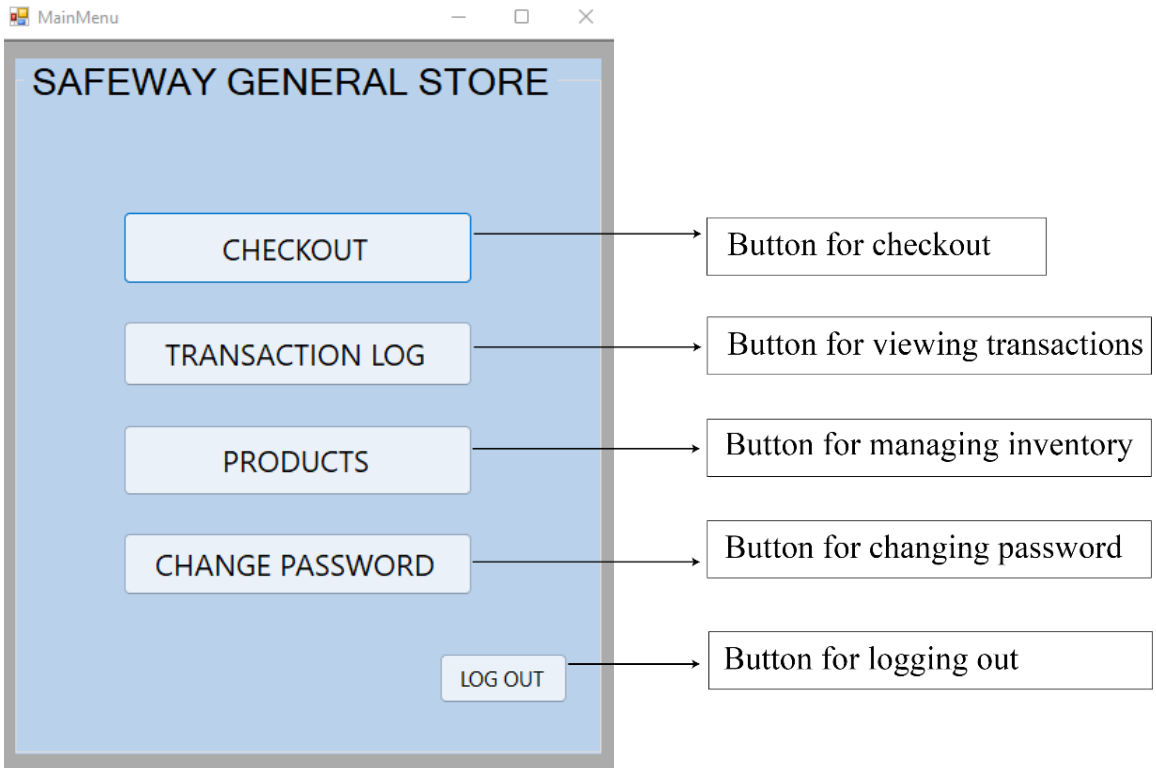


Image 3: Main Page

The main window consists of 4 main categories, Checkout, Transaction Log, Products and Change Password.

Product Window:

Upon clicking on the 'Product' button from the main window, the program directs the user to the program window.

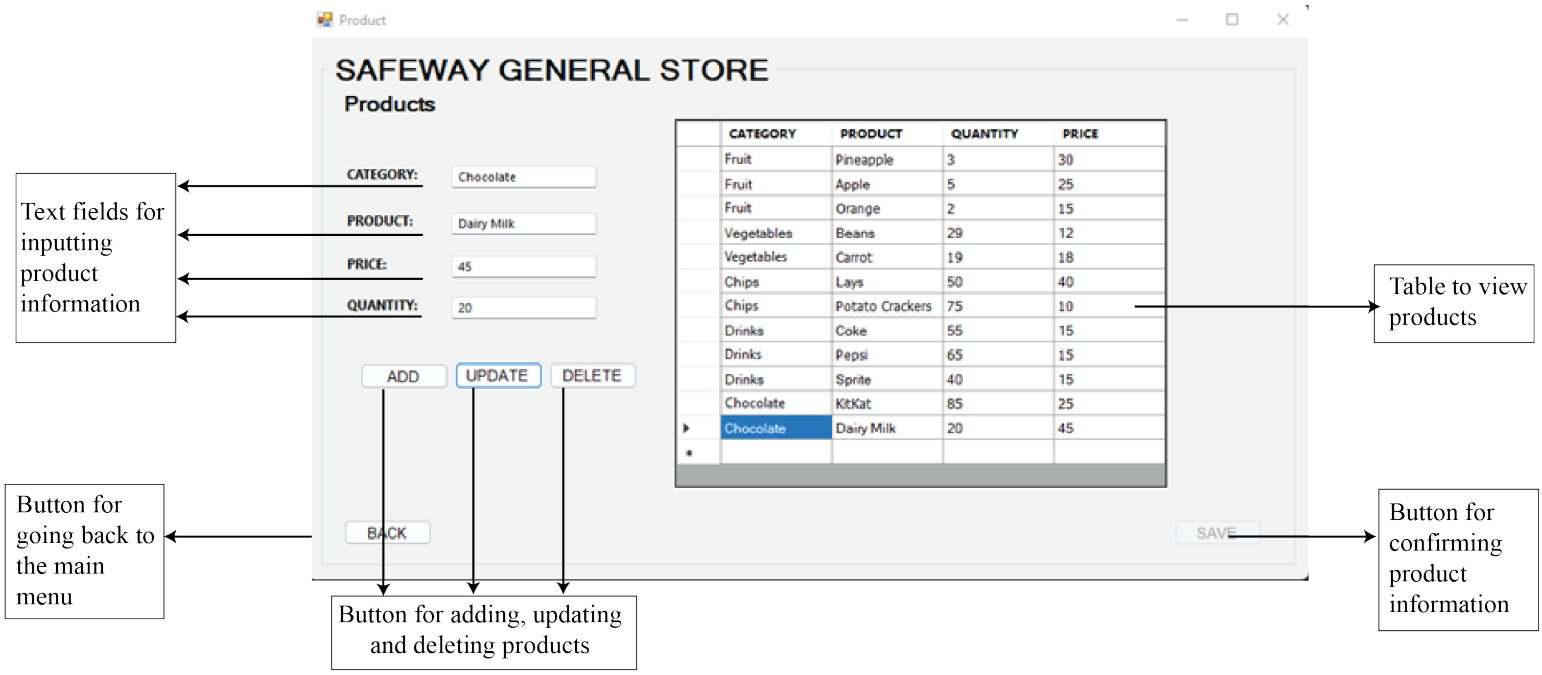


Image 4: Product Page

The product window allows the user to add, update and delete product data. This page includes text fields for entering data and buttons for adding the product into the program, making the program minimalistic and user friendly.

Adding product data into the database:

The C# code below were used to add the product information into the SQL database with the help of Microsoft SQL Server Management Studio.

```
23          private void gridLayout()...
34          private void btnadd_Click(object sender, EventArgs e) // add the product
35          {
36              ProductObject prod = new ProductObject();
37
38
39              if(checkproduct())
40              {
41                  prod.ProductCatagory = txtcat.Text;
42                  prod.ProductName = txtprod.Text;
43                  prod.ProductPrice = float.Parse(txtprice.Text); // if show the price the data type will be float
44                  prod.ProductQuantity = int.Parse(txtquantity.Text);
45
46
47                  DataGridViewRow newRow = new DataGridViewRow();
48                  newRow.CreateCells(datagrdProduct);
49
50                  newRow.Cells[0].Value = prod.ProductCatagory;
51                  newRow.Cells[1].Value = prod.ProductName;
52                  newRow.Cells[2].Value = prod.ProductQuantity;
53                  newRow.Cells[3].Value = prod.ProductPrice;
54
55                  datagrdProduct.Rows.Add(newRow);
56
57              }
58
59
60              btnsave.Enabled = true;
61
62
63          }
64
```

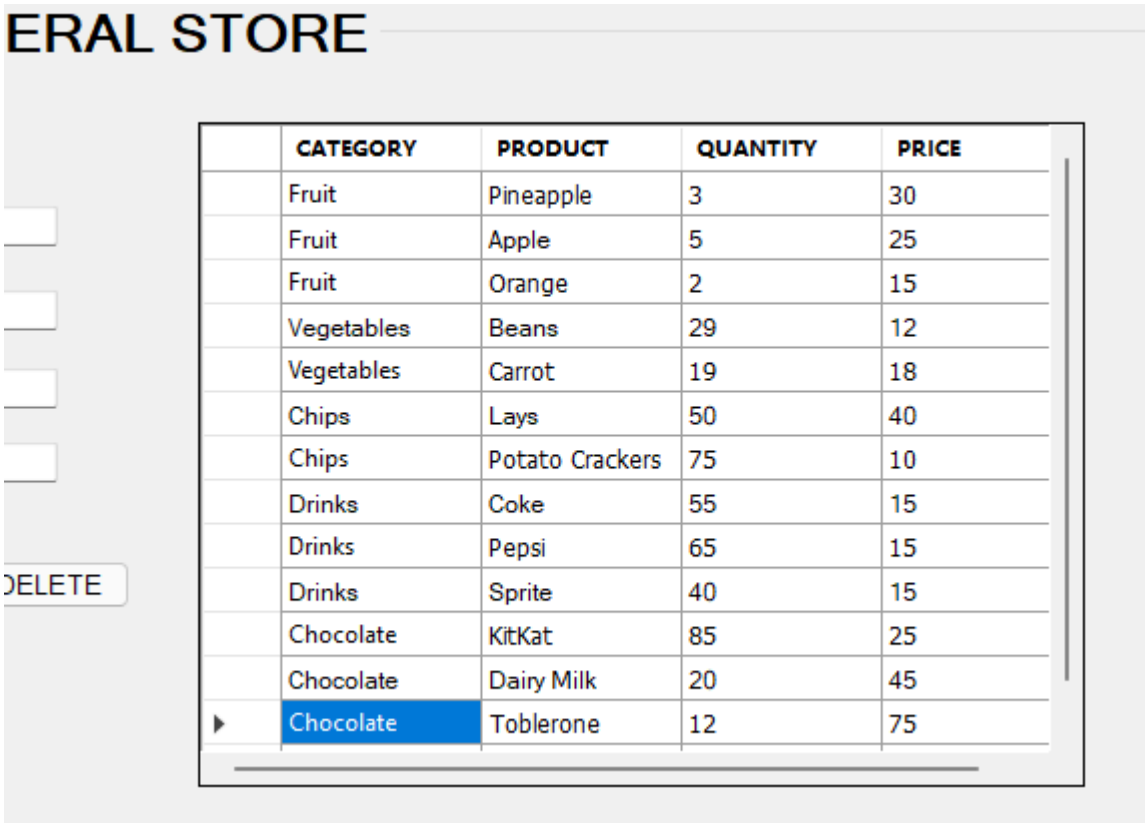
Code 2: Button for adding data into database

```
52
53          public bool AddProduct(ProductObject prodparameter)
54          {
55
56              try
57              {
58                  //SqlCommand sqlCom = new SqlCommand("select * from dms_DeviceLog where ID = (select MAX(ID) from dms_DeviceLog)");
59
60                  //conn.Open();
61                  //query = "spInsertTemplate";
62
63                  using (SqlConnection con = new SqlConnection(conString))
64                  {
65
66                      using (SqlCommand cmd = new SqlCommand("spAddProduct", con))
67                      {
68
69                          con.Open();
70                          cmd.CommandType = CommandType.StoredProcedure;
71
72                          cmd.Parameters.Add("@pCategory", SqlDbType.NVarChar,50).Value = prodparameter.ProductCatagory;
73                          cmd.Parameters.Add("@pName", SqlDbType.NVarChar, 50).Value = prodparameter.ProductName;
74                          cmd.Parameters.Add("@pPrice", SqlDbType.Float).Value = prodparameter.ProductPrice;
75                          cmd.Parameters.Add("@pQuantity", SqlDbType.Int).Value = prodparameter.ProductQuantity;
76
77
78                          cmd.ExecuteNonQuery();
79
80                          return true;
81                      }
82                  }
83
84              }
85              catch (Exception errorException)
86              {
87                  return false;
88              }
89
```

Code 3: Adding data into database [1]

[1] "How to insert data to a database in C#'," Stack Overflow, February 2022, https://stackoverflow.com/questions/57448296/how-to-insert-data-to-a-database-in-c-sharp

User can use the text fields on the side to input product information. If the data is validated, the user can click on the ‘add button’ to add the product to the database. Upon successful addition of product into the program, the product will be added to the table on the right-hand side.



	CATEGORY	PRODUCT	QUANTITY	PRICE
	Fruit	Pineapple	3	30
	Fruit	Apple	5	25
	Fruit	Orange	2	15
	Vegetables	Beans	29	12
	Vegetables	Carrot	19	18
	Chips	Lays	50	40
	Chips	Potato Crackers	75	10
	Drinks	Coke	55	15
	Drinks	Pepsi	65	15
	Drinks	Sprite	40	15
	Chocolate	KitKat	85	25
	Chocolate	Dairy Milk	20	45
▶	Chocolate	Toblerone	12	75

Image 5: Product Table View

However, in order to prevent incorrect data type to be inputted into the corresponding product information fields, there are various validations used in this part of the program. If the wrong input is given, error messages will pop up.

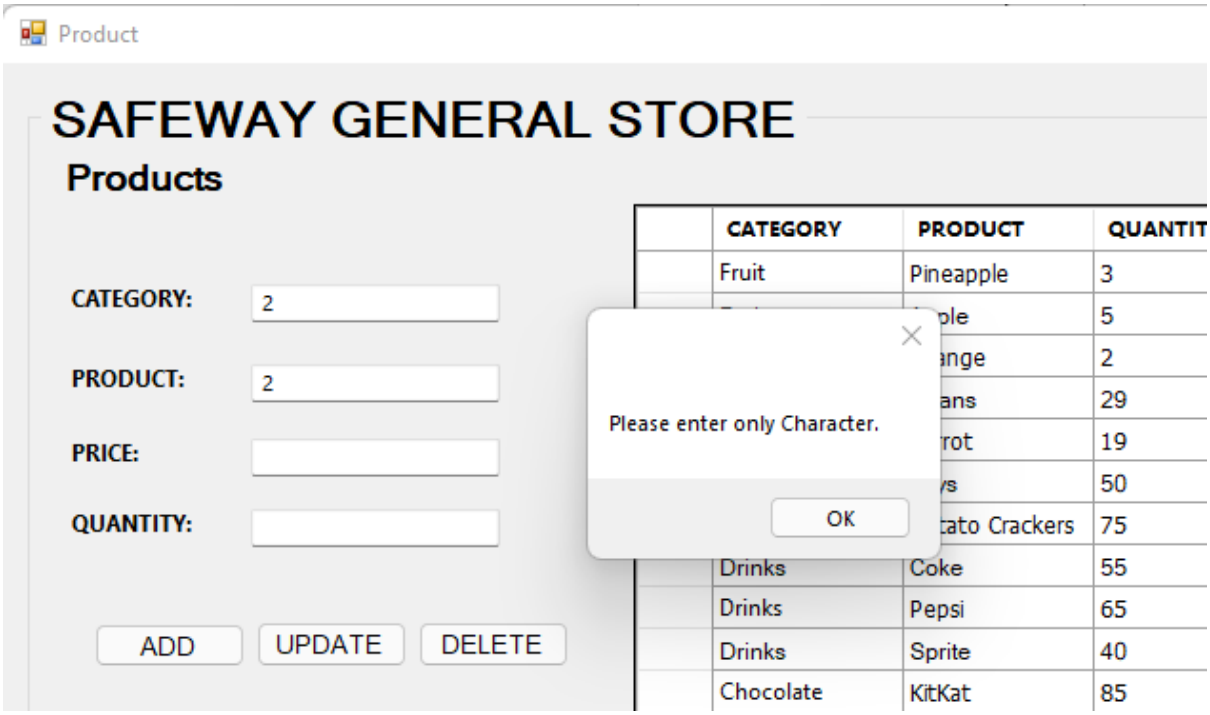


Image 6: Error message for wrong input

```
1 reference
private void txtcat_TextChanged(object sender, EventArgs e)
{

    if (!System.Text.RegularExpressions.Regex.IsMatch(txtcat.Text, "^[a-zA-Z ]*$"))
    {
        MessageBox.Show("Please enter only Character.");
        txtcat.Text = txtcat.Text.Remove(txtcat.Text.Length - 1);
    }
}

1 reference
private void txtprod_TextChanged(object sender, EventArgs e)
{
    if (!System.Text.RegularExpressions.Regex.IsMatch(txtprod.Text, "^[a-zA-Z ]*$"))
    {
        MessageBox.Show("Please enter only Character.");
        txtprod.Text = txtprod.Text.Remove(txtcat.Text.Length - 1);
    }
}
```

Code 4: Validation [2]

The image above is the code used to validate the 'Category' and 'Product" text field. This validation will only allow letters, white space and backspace to be entered, if any other data type such as integers are inputted, the error message will pop up and the product will not be added. If any text fields are empty spaces, an error would appear as well.

[2] "Validation of numbers and letters using C# for a beginner'," Stack Overflow, February 2022, https://stackoverflow.com/questions/8867097/validation-of-numbers-and-letters-using-c-sharp-for-a-beginner

```
2 references
142     private bool checkproduct()
143     {
144         if (txtcat.Text == "")...
149
150
151         if (txtprod.Text == "")
152         {
153             MessageBox.Show("Please Insert the Product Name ");
154         }
155
156
157         if (txtprice.Text == "")
158         {
159             MessageBox.Show("Please Insert the  price");
160             return false;
161         }
162
163
164         if (txtquantity.Text == "")
165         {
166             MessageBox.Show("Please Insert the Quantity");
167             return false;
168         }
169
170
171         return true;
172     }
173
```

Code 5: Validation code for empty spaces in text fields

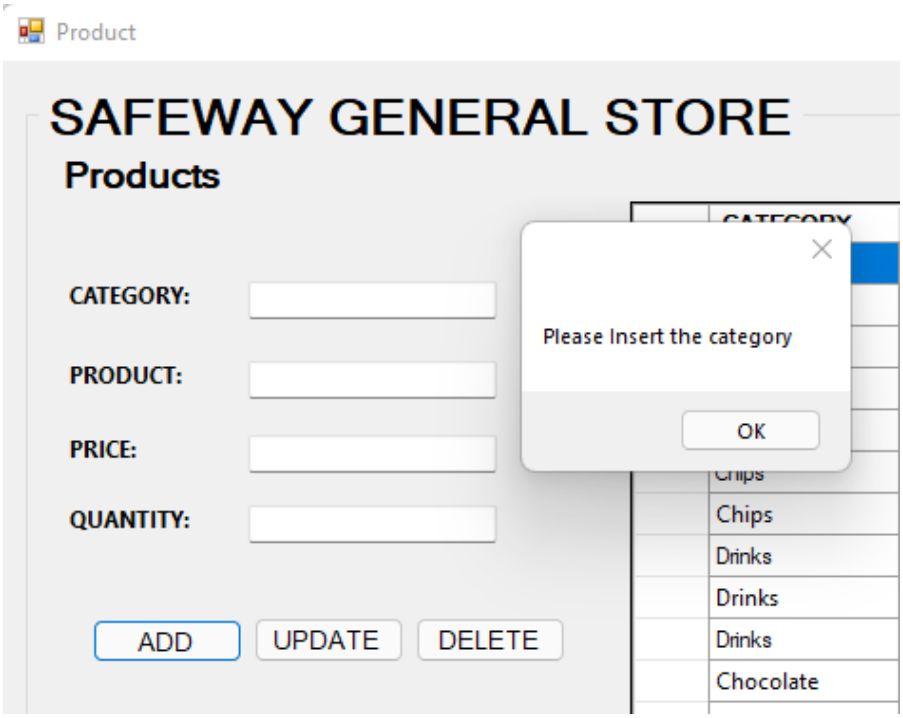


Image 7: Error message for blank text fields

Updating existing product data into the database:

If the user wants to edit existing product data, he simply needs to click on the product from the table that is displayed. The text fields will automatically be filled with the existing product information. The user can simply edit the necessary product information and click on the 'update' button. Once the data input is validated and saved into the database, a 'saved successfully' message will pop up.

```
1 reference
private void btnupdate_Click(object sender, EventArgs e) // if you want to update the product
                                                         // then select the item from the product list the update it from the txt box
{
    ProductObject prod = new ProductObject();
    if (checkproduct())
    {
        prod.ProductCatagory = txtcat.Text;
        prod.ProductName = txtprod.Text;
        prod.ProductPrice = float.Parse(txtprice.Text);
        prod.ProductQuantity = int.Parse(txtquantity.Text);


        if(repo.UpdateProductDetails(prod))
        {
            MessageBox.Show(prod.ProductName + " Update Sucessful");
            DisplayData();
            textClear();

        }

    }



}
```

Code 6: Button for updating data into database

```
1 reference
public bool UpdateProductDetails(ProductObject prodparameter)
{
    try
    {

        using (SqlConnection con = new SqlConnection(conString))
        {

            using (SqlCommand cmd = new SqlCommand("sp_UpdateProduct", con))
            {

                con.Open();
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.Add("@pCategory", SqlDbType.NVarChar, 50).Value = prodparameter.ProductCatagory;
                cmd.Parameters.Add("@pName", SqlDbType.NVarChar, 50).Value = prodparameter.ProductName;
                cmd.Parameters.Add("@pPrice", SqlDbType.Float).Value = prodparameter.ProductPrice;
                cmd.Parameters.Add("@pQuantity", SqlDbType.Int).Value = prodparameter.ProductQuantity;


                cmd.ExecuteNonQuery();

                return true;
            }
        }

    }
    catch (Exception errorException)
    {
        return false;
    }
}
```

Code 7: Updating data into database [3]

[3] Ehtesham Mehmood, "Insert, Update, Delete, Display Data in Mysql Using C#." C# Corner, June 3, 2020. https://www.c-sharpcorner.com/UploadFile/9582c9/insert-update-delete-display-data-in-mysql-using-C-Sharp/.

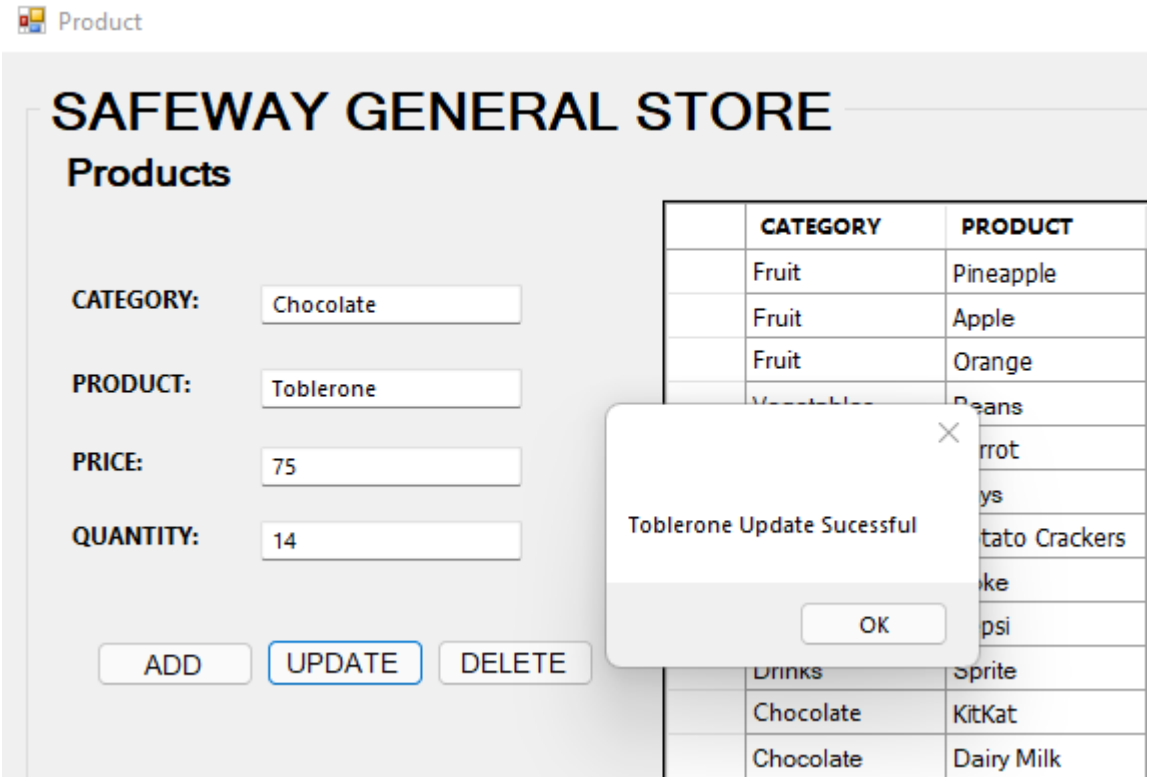


Image 8: Success message for updating product

Deleting a product data from the database:

If the user wants to edit delete a certain product, he simply needs to select on the product from the table and click on the delete button. If the product is deleted successfully from the database, a ‘deleted successfully’ message will pop up.

```
1 reference
private void btndelete_Click(object sender, EventArgs e) // select the item and delete it
{
    ProductObject prodObj = new ProductObject();
    prodObj.ProductCatagory = txtcat.Text;
    prodObj.ProductPrice  = float.Parse(txtprice.Text) ;
    prodObj.ProductName = txtprod.Text;
    prodObj.ProductQuantity = int.Parse(txtquantity.Text);



    if (repo.deleteProduct(prodObj))
    {
        MessageBox.Show("Deleted successfully");
        DisplayData();
        textClear();
    }
    else
    {
        MessageBox.Show("Not successfully deleted");
    }
}
```

Code 8: Button for deleting data into database

```
1 reference
public bool deleteProduct(ProductObject prodOBJ)
{
    try
    {
        //SqlCommand sqlCom = new SqlCommand("select * from dms_DeviceLog where ID = (select MAX(ID) from dms_DeviceLog)");

        //conn.Open();
        //query = "spInsertTemplate";

        using (SqlConnection con = new SqlConnection(conString))
        {

            using (SqlCommand cmd = new SqlCommand("sp_deleteProduct", con))
            {

                con.Open();
                cmd.CommandType = CommandType.StoredProcedure;


                cmd.Parameters.Add("@pProductName", SqlDbType.NVarChar, 50).Value = prodOBJ.ProductName;


                cmd.ExecuteNonQuery();

                return true;
            }
        }

    }
    catch (Exception errorException)
    {
        return false;
    }
}
```

Code 9: Deleting data from database

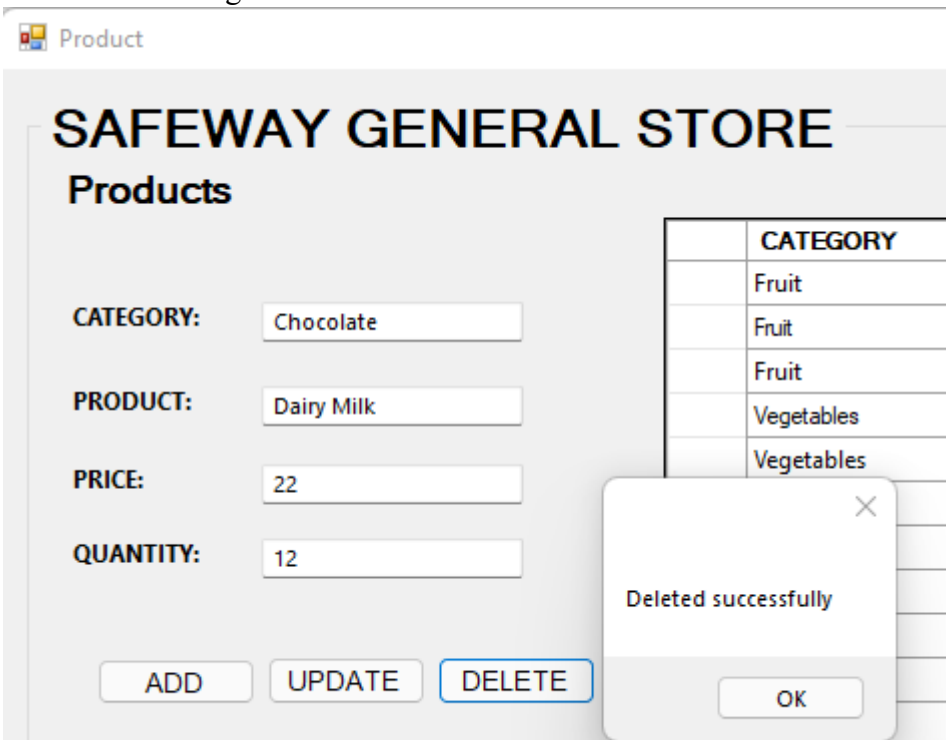


Image 9: Product deleted success message [4]

Once the user made the necessary changes to the products, the user has to click on the 'save' button on the bottom right corner. The user then can click on the back button to return to the main window.

[4] Ehtesham Mehmood, "Insert, Update, Delete, Display Data in Mysql Using C#." C# Corner, June 3, 2020. https://www.c-sharpcorner.com/UploadFile/9582c9/insert-update-delete-display-data-in-mysql-using-C-Sharp/.

Checkout Window:

The user can use the checkout window in order to record a sale of a product. Upon clicking the checkout button, the window below will be opened.

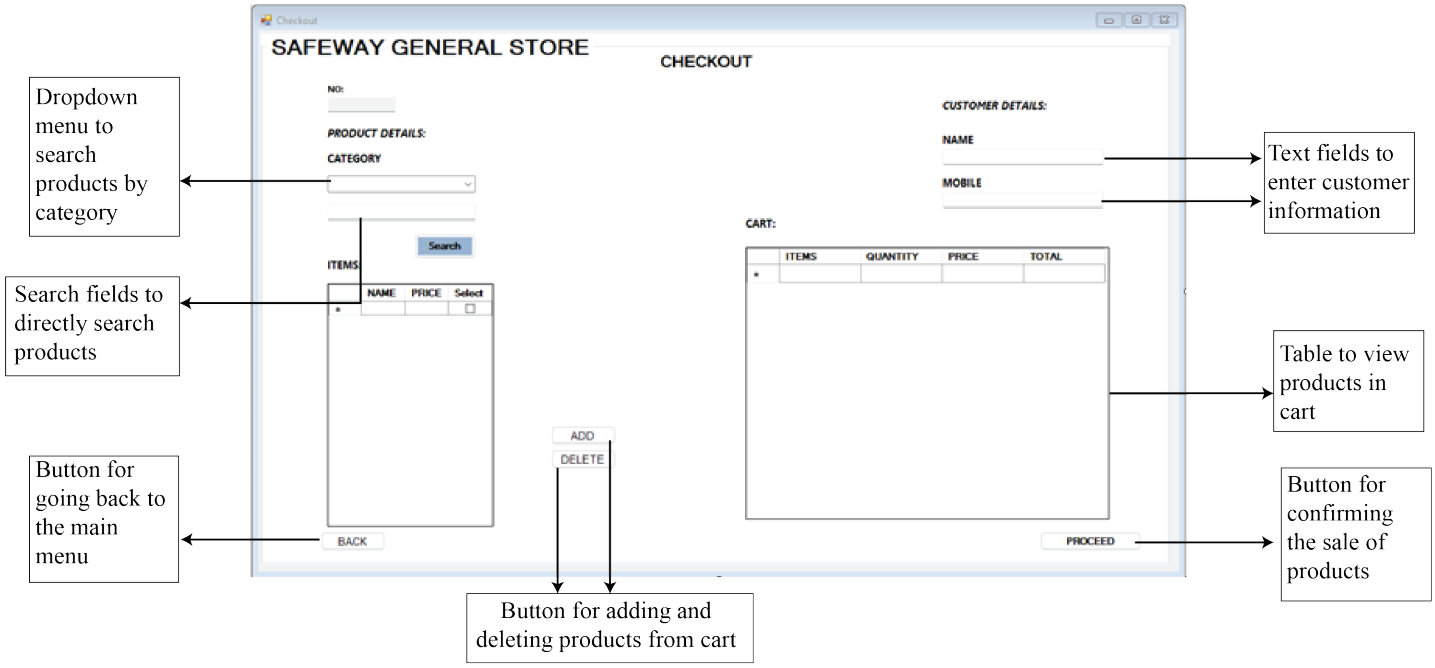


Image 10: Checkout Window

```
1 reference
private bool checkproduct()
{

    if (txtname.Text == "")
    {
        MessageBox.Show("Please Insert the Customer Name ");
    }

    if (txtmobile.Text == "")
    {
        MessageBox.Show("Please Insert the  Mobile No ");
        return false;
    }



    return true;
}

1 reference
private void btnback_Click(object sender, EventArgs e)...

1 reference
private void txtmobile_TextChanged(object sender, EventArgs e)
{
    if (System.Text.RegularExpressions.Regex.IsMatch(txtmobile.Text, "[^0-9]"))
    {
        MessageBox.Show("Please enter only numbers.");
        txtmobile.Text = txtmobile.Text.Remove(txtmobile.Text.Length - 1);
    }
}
```

Code 10: Validation for customer name and phone number

Firstly, the user needs to enter the name and mobile number of the customer. A validation check for the name and mobile number is in place as well. If wrong character input or empty space is given, an error message would popup.

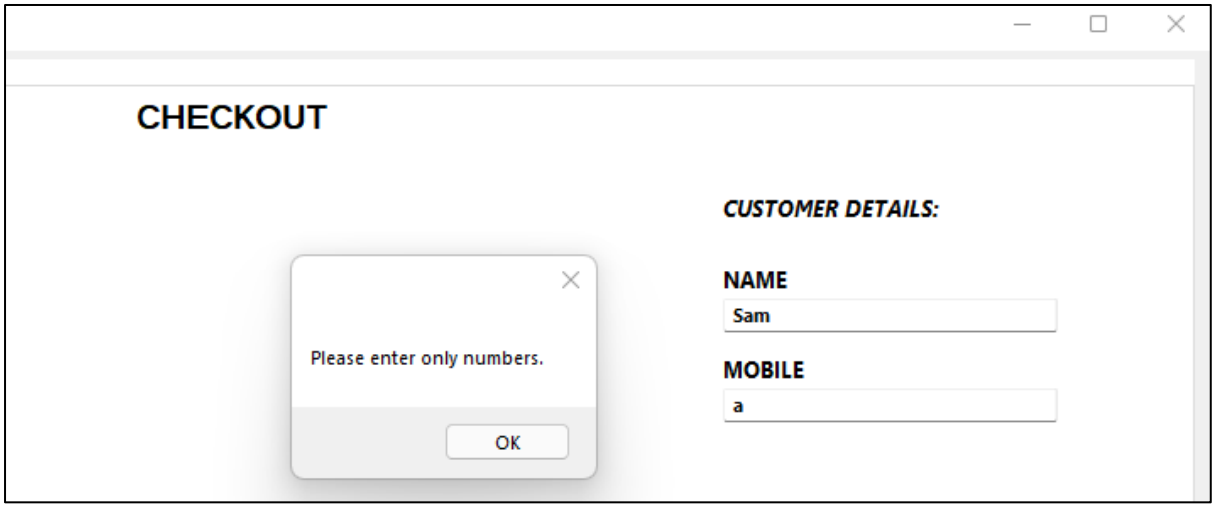


Image 8: Error message for wrong input

Afterwards, the user can select the category from the drop the menu or search the product specifically. The items will then be viewed on the item table and then the user can use the add and delete button to put the products into cart.

Searching a product into the database:

User can also use the search function in the checkout page to search for a product. User simply have to enter the product name in the search text field and click on the 'search' button.

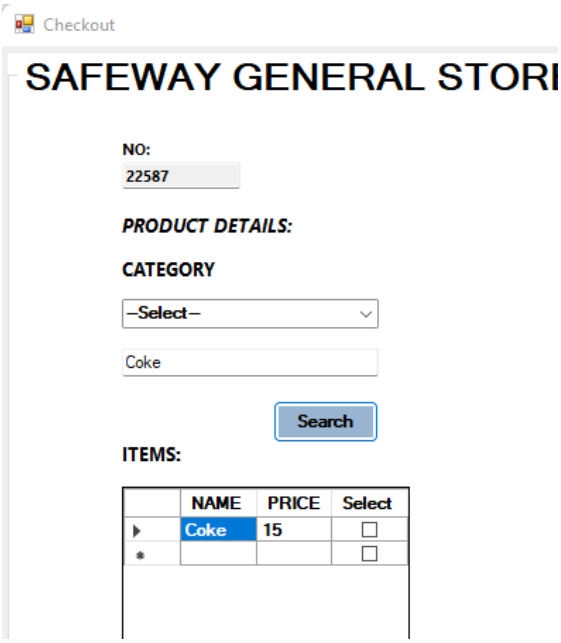


Image 9a: Search field

```
1 reference
private void btnSearch_Click(object sender, EventArgs e)
{
    String SearchItems = txtSearch.Text;



    dataGridViewItem.Rows.Clear();




    DataTable list = repo.GetProductItemList(SearchItems);



    foreach (DataRow dr in list.Rows)
    {
        string[] listda = { dr["Name"].ToString(), dr["Price"].ToString() };
        dataGridViewItem.Rows.Add(listda);



    }
}
```

Code 11: Search button code

```
1 reference
public DataTable GetProductItemList(string ItemName)
{



    using (SqlConnection con = new SqlConnection(conString))
    {



        using (SqlCommand cmd = new SqlCommand("spGetItemList", con))
        {
            con.Open();
            cmd.CommandType = CommandType.StoredProcedure;



            cmd.Parameters.Add("@pItem", SqlDbType.NVarChar, 50).Value = ItemName;// parameter.Name;



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

Code 12: Searching 'Products' from database [5]

[5] "Searching for and displaying sql database data C#," Stack Overflow, February 2022, https://stackoverflow.com/questions/26678009/searching-for-and-displaying-sql-database-data-c-sharp

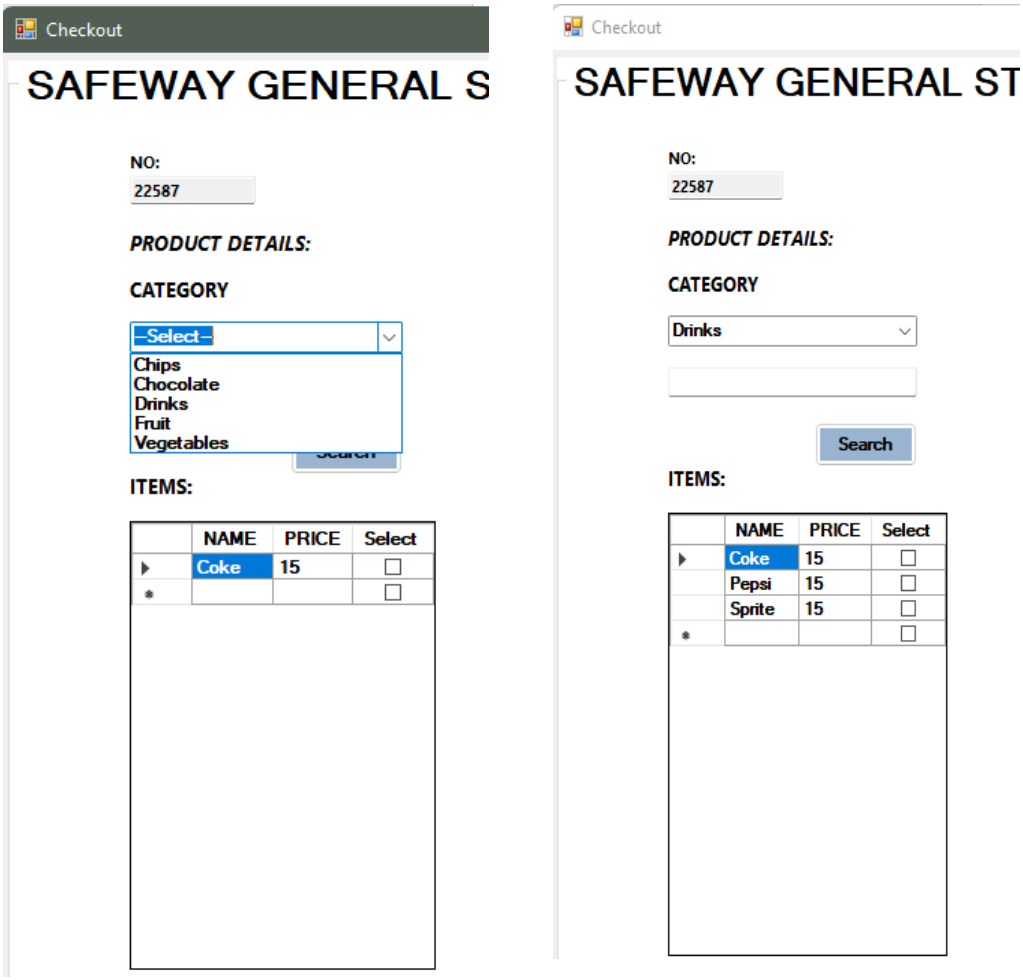


Image 9b: Viewing products by category

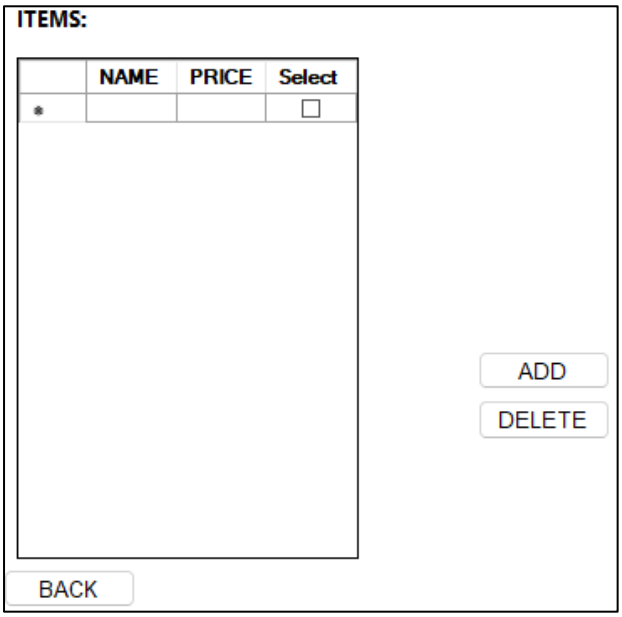


Image 10: Buttons to add items to the cart or delete items from the cart

```
1 reference
private void btnadd_Click(object sender, EventArgs e)
{
    DataTable dt = new DataTable();
    dt.Columns.Add("NAME");
    dt.Columns.Add("PRICE");
    foreach (DataGridViewRow row in dataGridViewItem.Rows)
    {
        bool isSelected = Convert.ToBoolean(row.Cells["chkbox"].Value);
        if (isSelected)
        {
            if(checkProdArrival(row.Cells[0].Value.ToString()))
            {
                if (AddItem(row.Cells[0].Value.ToString(), row.Cells[1].Value.ToString()))
                    DisplayData();
            }
            else
            {
                MessageBox.Show("PRODUCT Is not avilable");
            }


        }
    }
```

Code 13: Add product to cart button

```
1 reference
public bool AddItemBill(string itemName, string itemPrice)
{

    try
    {
        //SqlCommand sqlCom = new SqlCommand("select * from dms_DeviceLog where ID = (select MAX(ID) from dms_DeviceLog)");

        //conn.Open();
        //query = "spInsertTemplate";

        using (SqlConnection con = new SqlConnection(conString))
        {

            using (SqlCommand cmd = new SqlCommand("sp_InsertItemBill", con))
            {

                con.Open();
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.Add("@Item", SqlDbType.NVarChar, 50).Value = itemName;
                cmd.Parameters.Add("@Price", SqlDbType.Float).Value = float.Parse(itemPrice);



                cmd.ExecuteNonQuery();

                return true;
            }
        }

    }
```

Code 14: Adding product to cart

```
1 reference
private bool DeleteItem(string itemname, string price)
{

    return repo.DeleteItemBill(itemname, price);
}
```

Code 15: Delete product to cart button

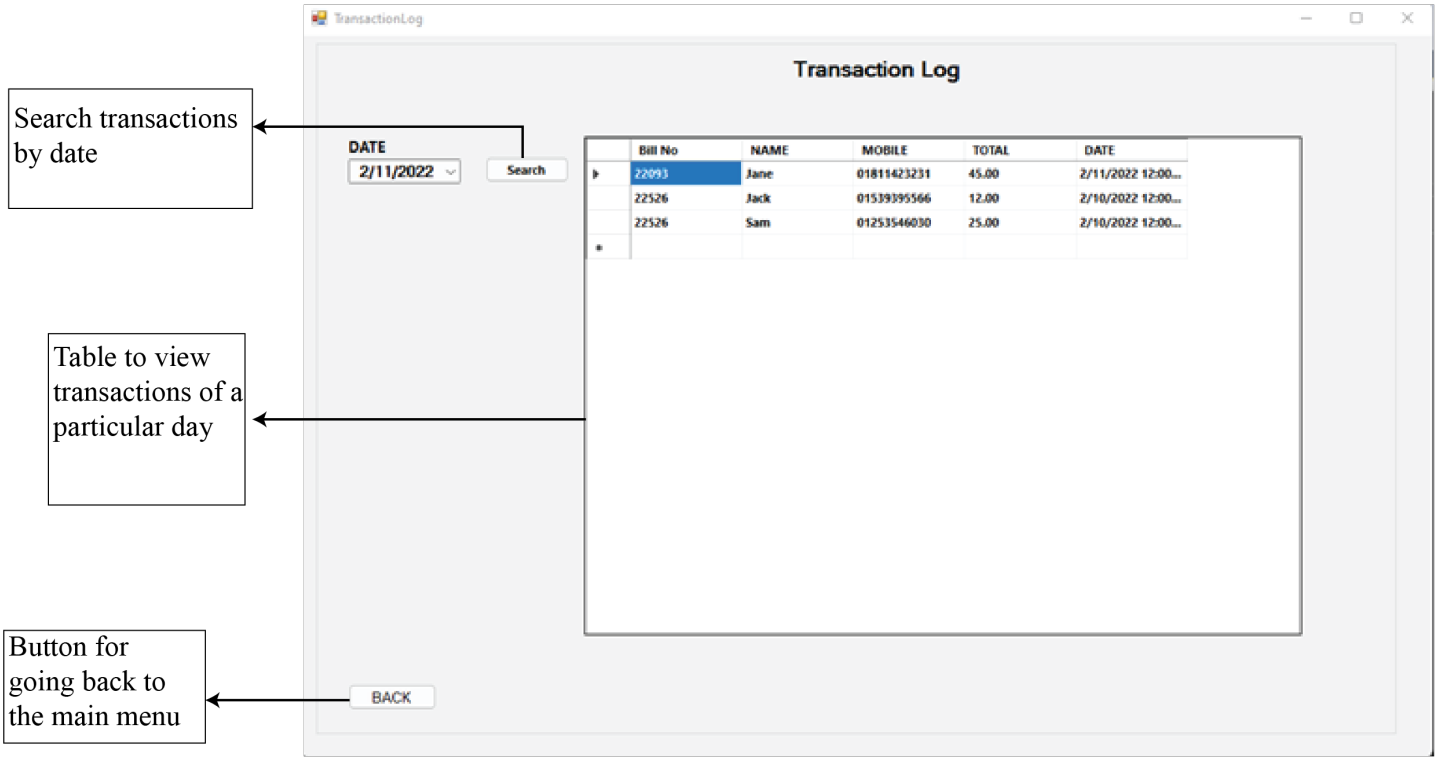
```
1 reference
public bool DeleteItemBill(string itemName, string itemPrice)
{

    try
    {
        //SqlCommand sqlCom = new SqlCommand("select * from dms_DeviceLog where ID = (select MAX(ID) from dms_DeviceLog)");

        //conn.Open();
        //query = "spInsertTemplate";


        using (SqlConnection con = new SqlConnection(conString))
        {

            using (SqlCommand cmd = new SqlCommand("sp_DeleteItemBill", con))
            {

                con.Open();
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.Add("@Item", SqlDbType.NVarChar, 50).Value = itemName;
                cmd.Parameters.Add("@Price", SqlDbType.Float).Value = float.Parse(itemPrice);
                cmd.ExecuteNonQuery();


                return true;
            }
        }

    }
    catch (Exception errorException)
    {
        return false;
    }
```

Code 16: Deleting product from cart

After the products are added into the cart, the user can then click on ‘proceed’ button for checkout.

Transaction Log Window:

The user can click on the transaction log button to view past transaction of sales that took place in his store.



The user can select a particular date and click on the search button to view the list of transactions that took on that particular day. Important information such as Bill No., Name, mobile and the total amount sold is displayed on the table.

```
1 reference
public DataTable GetTransactionLog()
{


    using (SqlConnection con = new SqlConnection(conString))
    {

        using (SqlCommand cmd = new SqlCommand("sp_GetTransactionLog", con))
        {
            con.Open();
            cmd.CommandType = CommandType.StoredProcedure;

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


            //foreach (DataRow dr in dt.Rows)
            //{
            //    if (dr["UserName"] != null)
            //    {
            //        if (parameter.Username == (string)dr["UserName"] && parameter.Password == (string)dr["Password"])
            //            return true;

            //    }
            //}


            return dt;
        }
    }
```

```
1 reference
private void btnSearch_Click(object sender, EventArgs e)
{

    DateTime theDate = dateTimeTransactionLog.Value;
    DisplayData(theDate);


}
```

Code 14: Code for searching transaction records by date

Change Password Window:

If the user wishes to change his password, he can go to the change password window to change it accordingly.

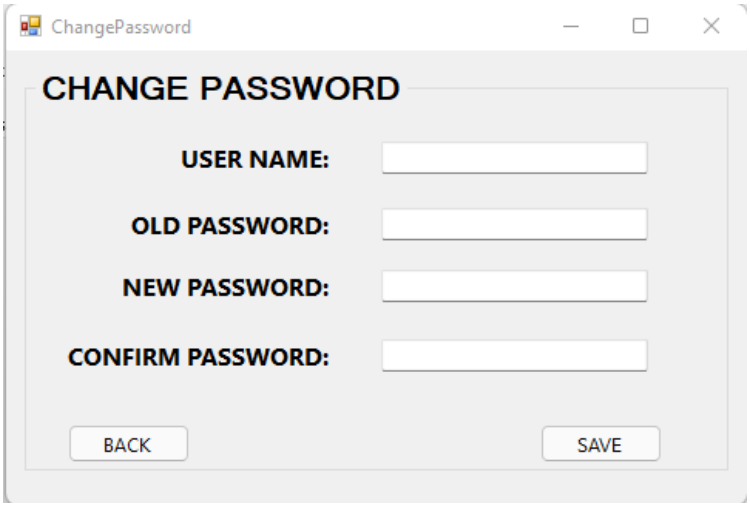


Image 12: Change password window

A verification is taken initially, as the program will be asking the user to input his existing username and password. Once the new password is inputted, the user will again be prompted to re-input the new password. A series of validations will take place such as, if the username old password does not match with the data in the database, and error message will appear. Similarly, if the new password entered doesn't match with the input that is in confirm password, an error message will pop up as well.

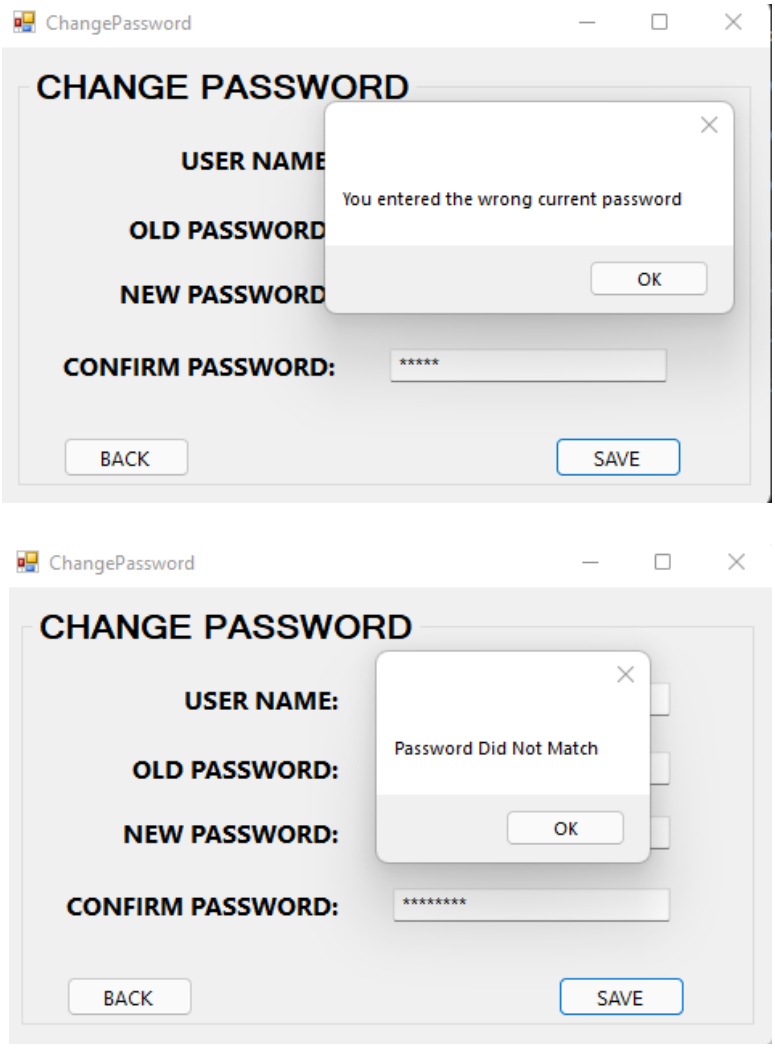


Image 13: Password change error messages

If the validations are successful, a 'password successfully changed' message will popup, informing the user that the password change was successful.

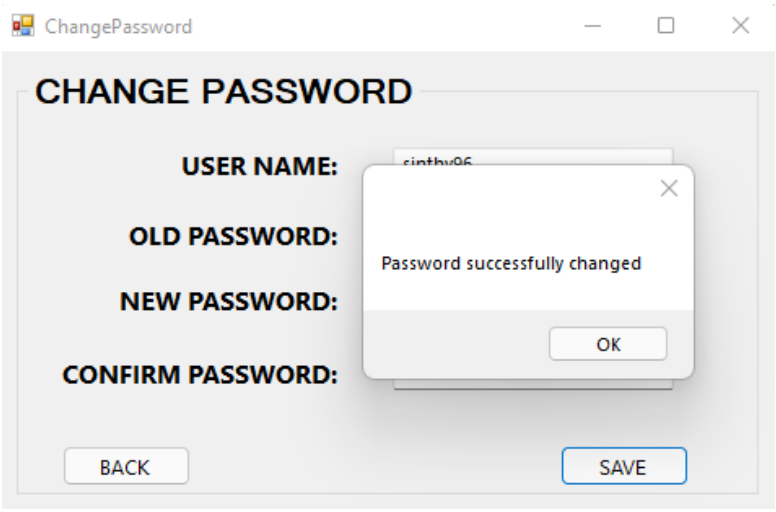


Image 14: Password change success message

```
1 reference
private void btnsave_Click(object sender, EventArgs e)
{
    SecUser ChangeUser = new SecUser();

    if (checkpass())
    {
        ChangeUser.Username = txtUserName.Text;
        ChangeUser.OldPassword = txtOldPassword.Text;
        ChangeUser.NewPassword = txtNewPassword.Text;
        ChangeUser.Password = txtConfirmPassword.Text;



        if (ChangeUser.NewPassword == ChangeUser.Password)
        {
            DBHandler chPass = new DBHandler();

            if (chPass.CheckExistingPassword(ChangeUser))
            {
                if (chPass.ChangePassword(ChangeUser))
                {
                    MessageBox.Show("Password successfully changed");
                    ClearText();

                }

                else
                {
                    MessageBox.Show("Password Did not Match");

                }

            }
            else
            {
                MessageBox.Show("You entered the wrong current password");
            }
        }
        else
        {
            MessageBox.Show("Password Did Not Match");
        }
    }
    }
```

Code 14: Password validation [6]

[6] "How to change password to user account, by c# code?" Stack Overflow, February 2022, https://stackoverflow.com/questions/4253893/how-to-change-password-to-user-account-by-c-sharp-code

```
1 reference
private bool checkpass()
{
    if (txtUserName.Text == "")
    {
        MessageBox.Show("Please Insert the Username");
        return false;
    }

    if (txtOldPassword.Text == "")
    {
        MessageBox.Show("Please Insert your old password ");
    }


    if (txtNewPassword.Text == "")
    {
        MessageBox.Show("Please Insert the new password");
        return false;
    }

    if (txtConfirmPassword.Text == "")
    {
        MessageBox.Show("Please confirm your password");
        return false;
    }

    return true;
}
```

Once the user has completed his work, he can simply logout from the program using the logout button from the main window.

Methods and classes used:

Classes:

No.	Class	Description
1	public class DBHandler	Main class
2	public partial class Login	Login Window Class
3	public partial class MainMenu	Main Menu Window Class
4	public partial class Checkout	Checkout Window Class
5	public partial class frmTransaction	Transaction Log Window Class
6	public partial class frmproduct	Products Window Class
7	public partial class frmChangePassword	Password Change WindowClass

Public Main Methods:

No.	Method	Description
1	public bool AddProduct(ProductObject prodparameter)	Method to push the product data into the database
2	internal bool AddProduct(object prodObj)	Method for adding product into the database
3	public bool AddTransactionLog(ProductObject prodparameter)	Method for storing
4	public bool UserCheck(string UserName, string UserPassword, int strUserType)	for the validations of user id from database
5	public bool LogIn(SecUser parameter)	Method for getting user from database
6	public DataTable deleteProduct()	Method for deleting product from database
7	public DataTable Getcatogry()	Method for getting product category from database
8	public DataTable GetProductCategoryList(string categoryName)	Method for getting product category list for the dropdown menu
9	public DataTable GetProductItemList(string ItemName)	Getting product item list for the search
10	public bool UpdateProductDetails(ProductObject prodparameter)	Method for updating product details in database and table
11	public bool deleteProduct(ProductObject prodOBJ)	Method for deleting product from the database
12	public bool ChangePassword (SecUser ChangePass)	Method for changing user password
13	public bool CheckExistingPassword(SecUser ChangePass)	Method for validating existing username and password from database

14	public bool checkprod (string prodname)	Method for checking whether product is available in stock
15	public bool productCalculation(string prodName)	Method for stock deduction
16	public DataTable GetProductbillList()	Method for getting product cart list
17	public DataTable GetTransactionLog()	Method for pulling data for the transaction log from the database
18	public DataTable GetTransactionLog(DateTime adddate)	searching transaction log by date
19	public bool ClearTempBillTable()	Method for clearing checkout list
20	public bool AddItemBill(string itemName, string itemPrice)	Method for adding item to checkout from database
20	public bool DeleteItemBill(string itemName, string itemPrice)	Method for deleting item to checkout from database

Private methods used in login page:

No.	Method	Description
1	private void btnlogin_Click(object sender, EventArgs e)	Method for logging user to main menu
2	private void txtpassword_TextChanged_1(object sender, EventArgs e)	Verification for logging user into the program

Private methods used in products page:

No.	Method	Description
1	private void btnadd_Click(object sender, EventArgs e)	Add button
2	private void btnupdate_Click(object sender, EventArgs e)	Update button
3	private void btnsave_Click(object sender, EventArgs e)	Save button
4	private void btndelete_Click(object sender, EventArgs e)	Delete Button
5	private void DisplayData()	Displays products in the table
6	private void datagrdProduct_CellClick(object sender, DataGridViewCellEventArgs e)	Method for selecting product from its cell
7	private void textClear()	Clear text from text field
8	private void btnback_Click(object sender, EventArgs e)	Back button to go back to main menu
9	private void txtcat_TextChanged(object sender, EventArgs e)	Method for Category Field validation
10	private void txtprod_TextChanged	Method for Product Field validation

Private methods used in checkout page:

No.	Method	Description
1	private void label1_Click(object sender, EventArgs e)	Method for clicking label
2	private void loadCombobox()	Method for category list
3	private void cmbcat_SelectedIndexChanged(object sender, EventArgs e)	Dropdown category list
4	private void dataGridViewItem_CellContentClick(object sender, DataGridViewCellEventArgs e)	Showing item on grid

5	private void btnadd_Click(object sender, EventArgs e)	Adding items to cart button
6	private void btndelete_Click(object sender, EventArgs e)	Deleting items from cart button
7	private bool checkProdArrival(string prodName)	Check for stock
8	private bool AddItem(string itemname , string price)	Method call for add item action
9	private bool DeleteItem(string itemname, string price)	Method call for delete item action
10	private void DisplayData()	Displaying item on table
11	private void btnSave_Click(object sender, EventArgs e)	Proceed Button
12	private bool checkproduct()	Validation for customer name
13	private void txtmobile_TextChanged(object sender, EventArgs e)	Validation for customer phone number
14	private void btnback_Click(object sender, EventArgs e)	Back button to go back to main menu
15	private void btnSearch_Click(object sender, EventArgs e)	Search Button

Private methods used in transaction log page:

No.	Method	Description
1	private void DisplayData(DateTime adddate)	Display table by date
2	private void btnSearch_Click(object sender, EventArgs e)	Search Button
3	private void btnback_Click(object sender, EventArgs e)	Back button to go back to main menu

Word Count – 999 Words (Excluding captions and headings)

Bibliography

- "How to insert data to a database in C#'," Stack Overflow, February 2022, https://stackoverflow.com/questions/57448296/how-to-insert-data-to-a-database-in-c-sharp
- Mehmood, Ehtesham. "Insert, Update, Delete, Display Data in Mysql Using C#." C# Corner, June 3, 2020. https://www.c-sharpcorner.com/UploadFile/9582c9/insert-update-delete-display-data-in-mysql-using-C-Sharp/.
- "Searching for and displaying sql database data C#," Stack Overflow, February 2022, https://stackoverflow.com/questions/26678009/searching-for-and-displaying-sql-database-data-c-sharp
- "Validation of numbers and letters using C# for a beginner'," Stack Overflow, February 2022, https://stackoverflow.com/questions/8867097/validation-of-numbers-and-letters-using-c-sharp-for-a-beginner