# GROUP ASSIGNMENT

## CT044-3-1 Introduction to Object Oriented Programming

## Laptop Repair Service Management System

**APU1F2009MMT, APU1F2009CS(IS)**

**Group 8**

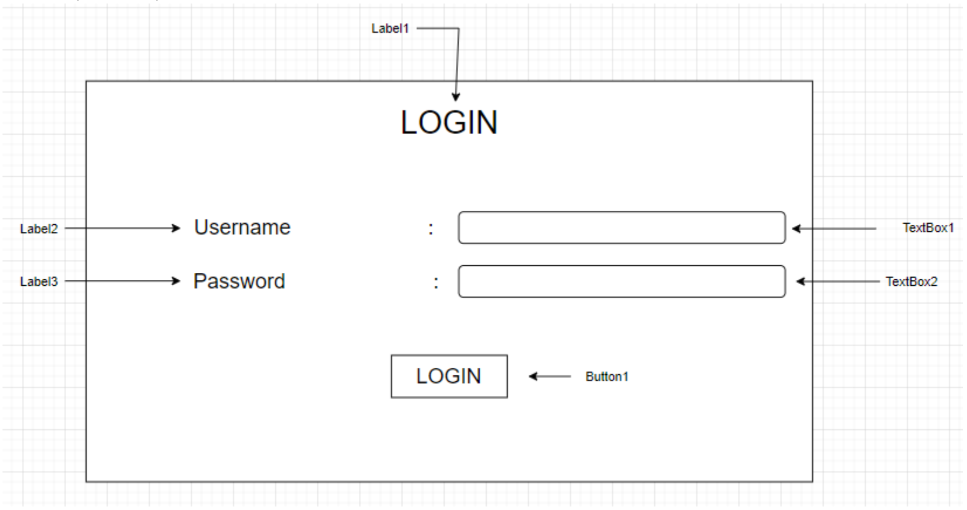
| | |
|---|---|
| **Group Members** | **: 1. Ariel Amerigo Joe Banua (TP063209)** |
| | **2. Nathaniel Sudiono (TP063926)** |
| | **3. Mikael Owen Kartika (TP061843)** |
| | **4. Stanley Lais (TP061843)** |
| **Lecture** | **: DR. SATHIAPRIYA RAMIAH** |
| **Hand Out Dates** | **: 18 APRIL 2021** |
| **Hand In Dates** | **: 19 JUNE 2021** |

## Contents

## Contents

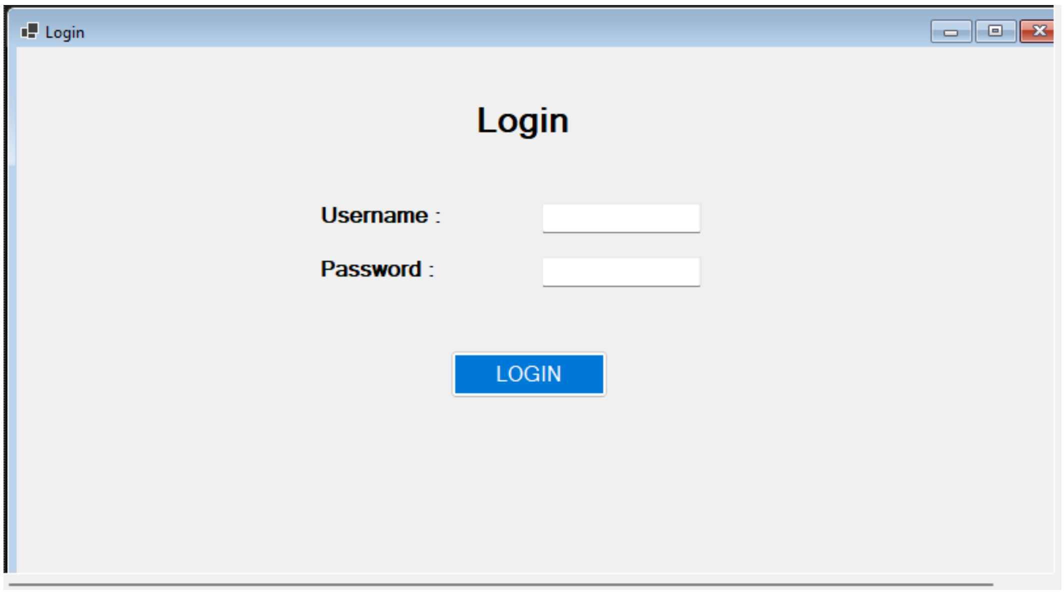
## Storyboard

Login Form (Owen)

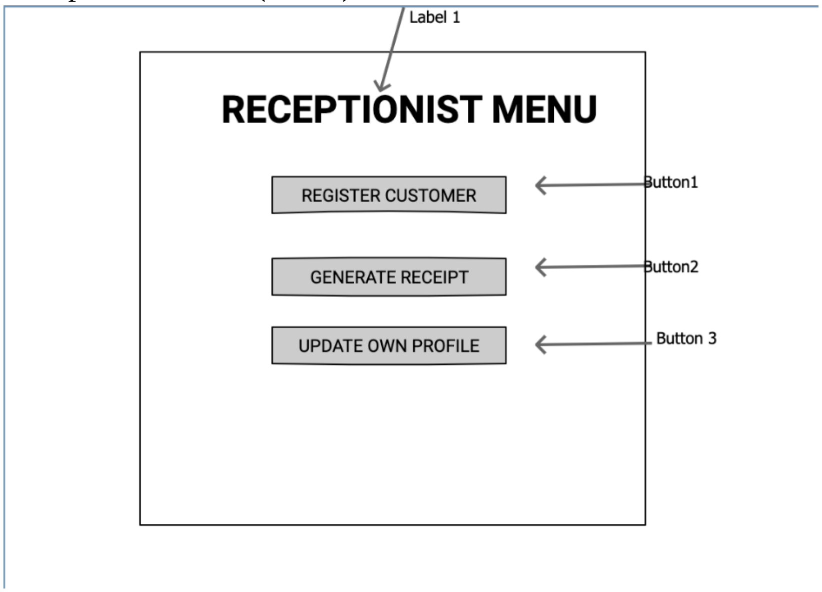


| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblLogin | To label form title |
| Label 2 | lblUsername | To label username textbox |
| Label 3 | lblPassword | To label password textbox |
| Textbox 1 | txtUsername | To fill username |
| Textbox 2 | txtPassword | To fill password |
| Button 1 | btnLogin | To continue login |

Sample UI :

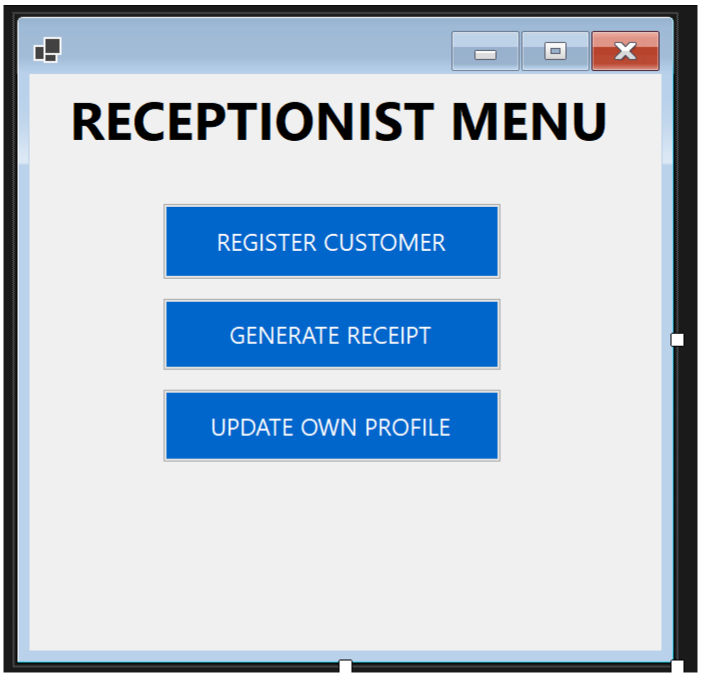


## Receptionist Menu (Owen)

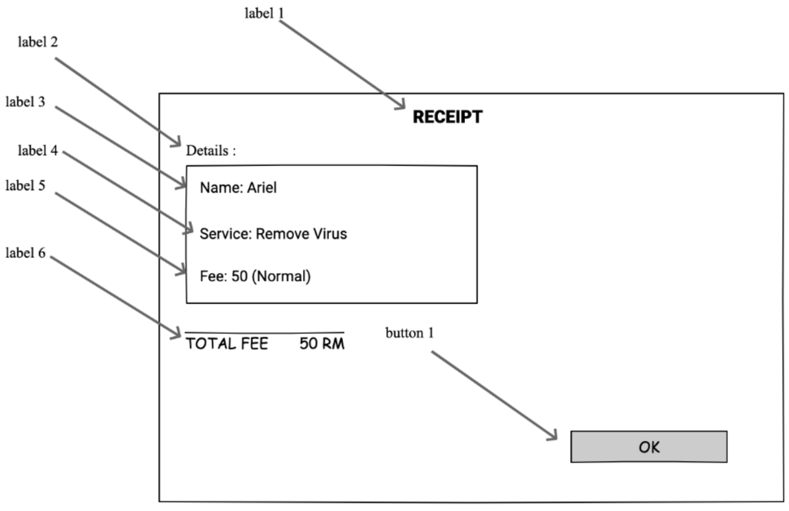


| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblHeader | To label form header |
| Button 1 | btnRegister | To let customer register |
| Button 2 | btnReceipt | To allow receptionist generate receipt |
| Button 3 | btnUpdate | To change receptionist's profile |

Sample UI :

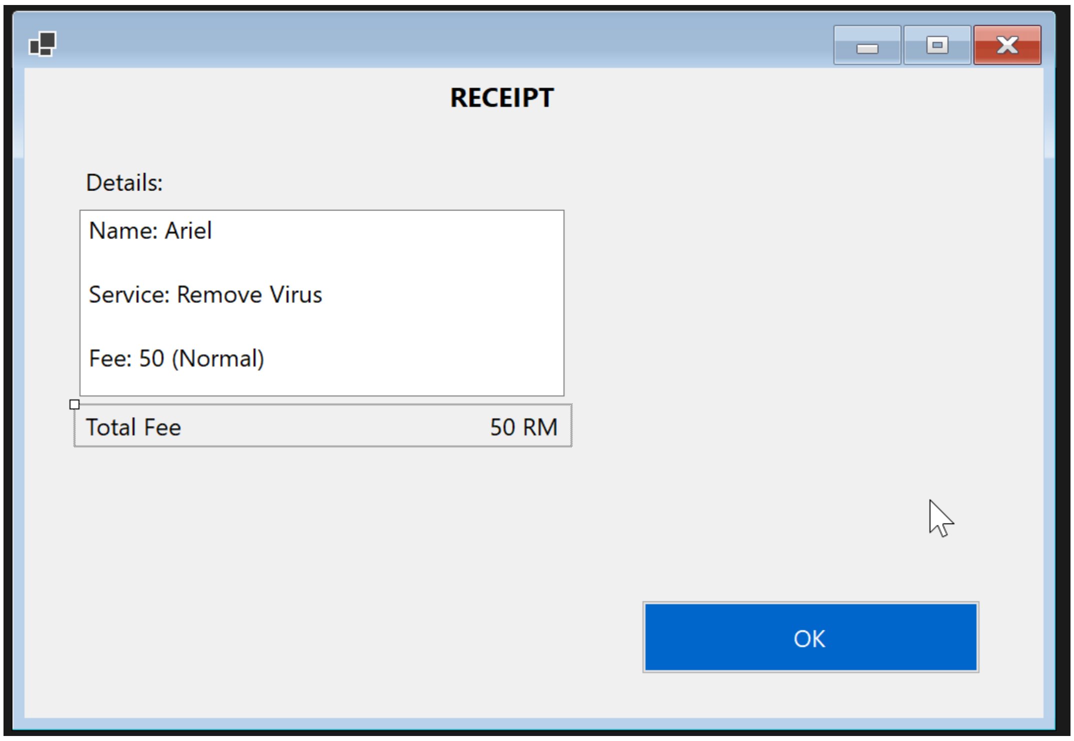


## Generate Receipt (Owen)



| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblTitle | To label form header |
| Label 2 | lblDetails | To show user details |
| Label 3 | lblName | To show user name |
| Label 4 | lblService | To show user service request |
| Label 5 | lblFee | To show service costs |
| Label 6 | lblTotal | To present total fee |
| Button 1 | btnReceipt | To proceed the receipt |

Sample UI :



Details:

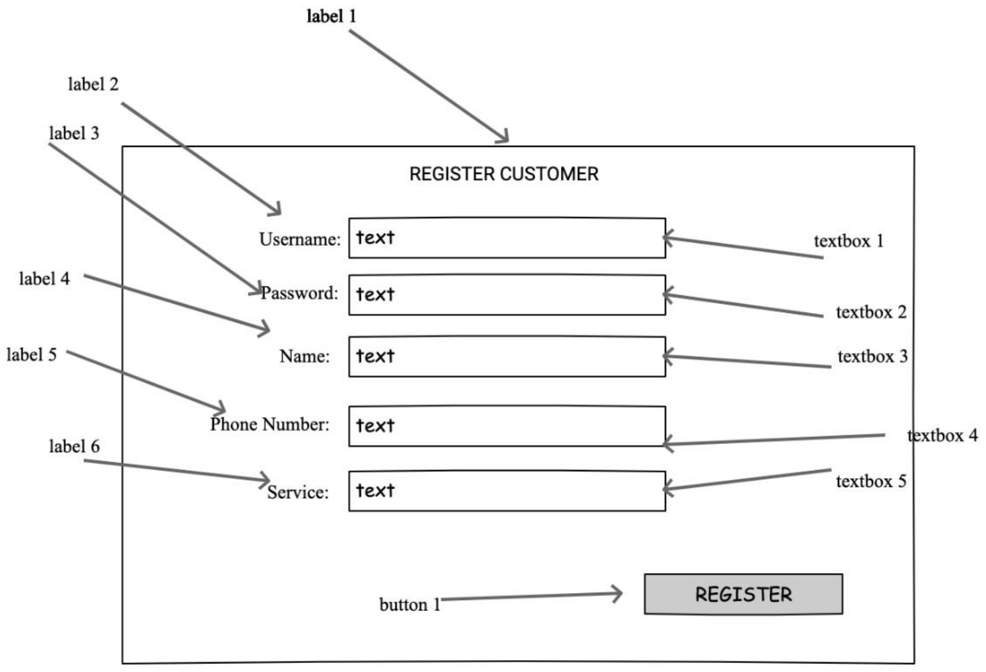
Name: Ariel

Service: Remove Virus

Fee: 50 (Normal)

Total Fee 50 RM

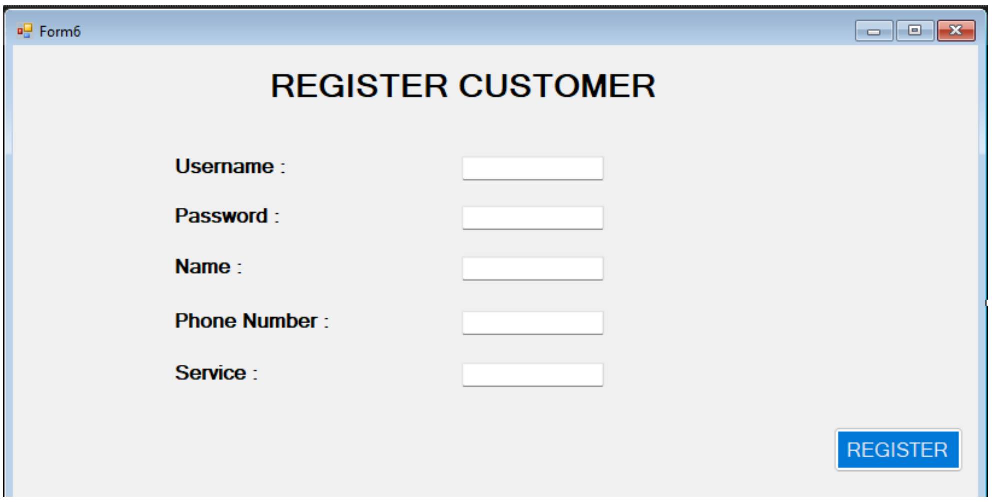
Register Customer (Owen)



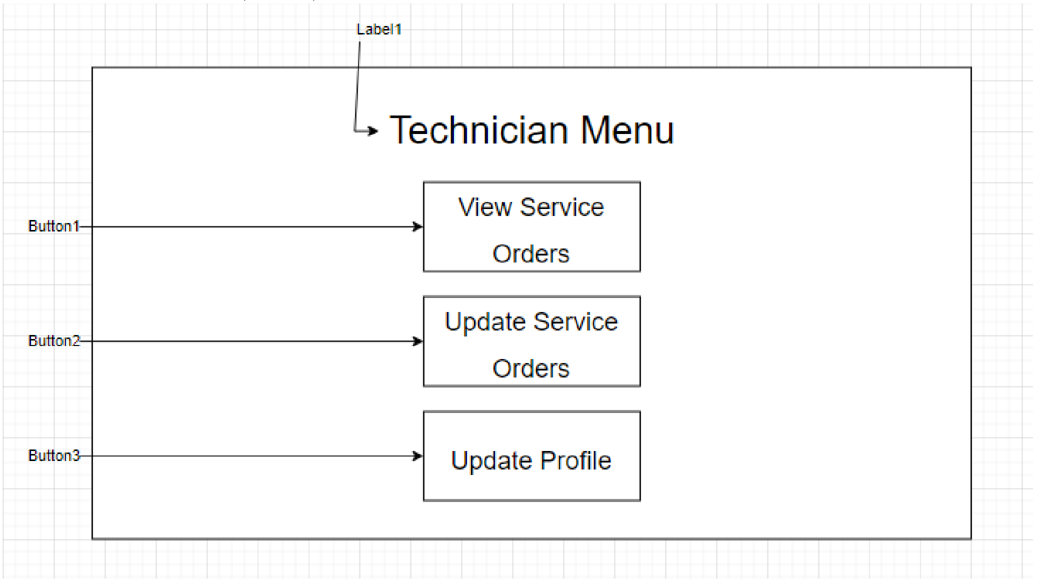
| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblTitle | To inform the title |
| Label 2 | lblUsername | To label the username box |
| Label 3 | lblPassword | To label the password box |

| | | |
|---|---|---|
| Label 4 | lblName | To label the name box |
| Label 5 | lblPhone | To label the phone number box |
| Label 6 | lblService | To label the service box |
| Textbox 1 | txtboxUsername | To fill username |
| Textbox 2 | txtboxPassword | To fill password |
| Textbox 3 | txtboxName | To fill name |
| Textbox 4 | txtboxPhone | To fill phone number |
| Textbox 5 | TxtboxService | To know what part need services |
| Button 1 | btnRegister | To register customer |

Sample UI :

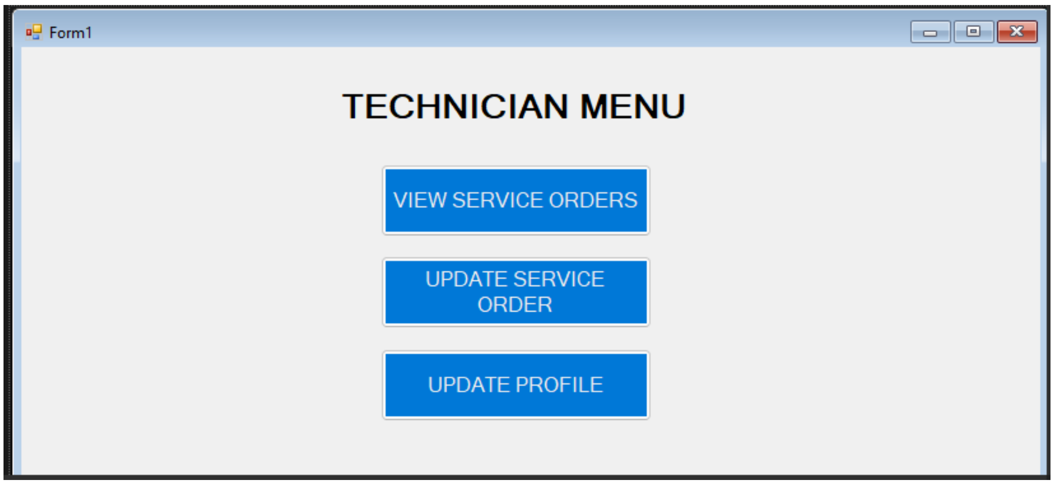


Technician Menu (Ariel)

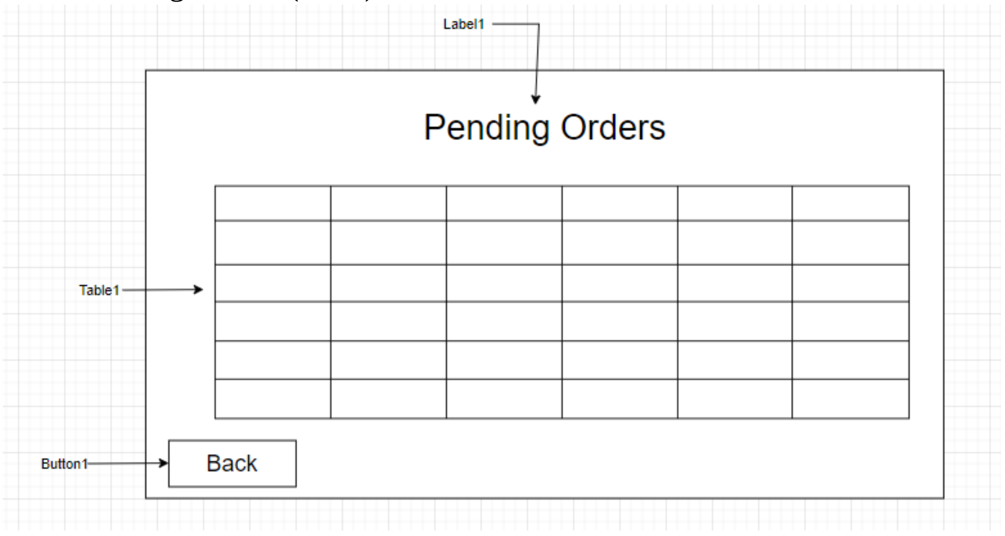


| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblTitle | To label the header which is the technician menu |
| Button 1 | btnViewOrder | To let users go to the “view service orders” form page |
| Button 2 | btnUpdateOrder | To let users go to the “Update service orders” form page |
| Button 3 | btnUpdateProfile | To let users go to the “Update Profile” form page |

Sample UI :

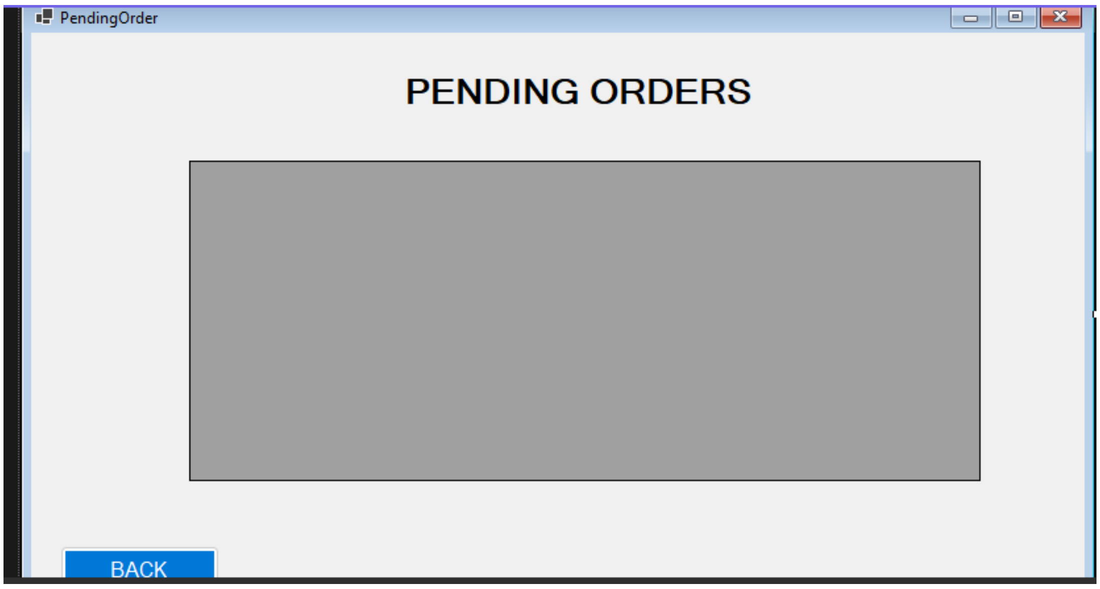


View Pending Orders (Ariel)

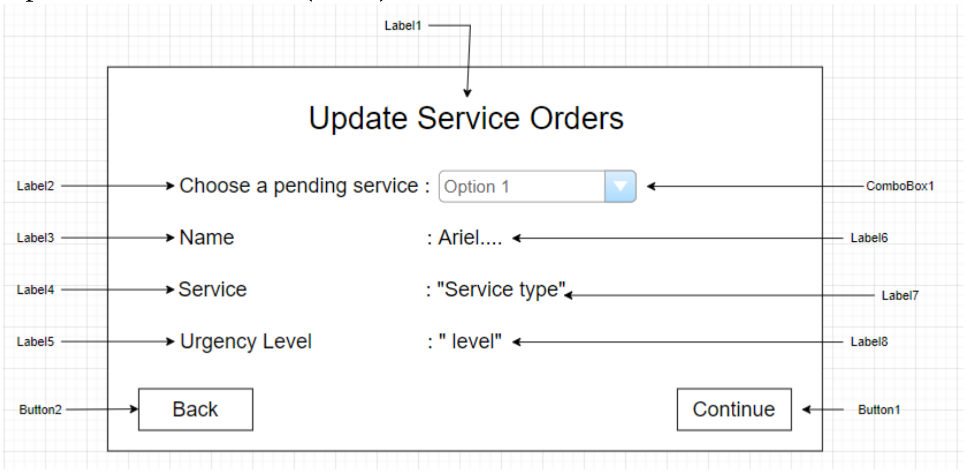


| Control | Control Name | Description |
|---|---|---|
| Label 1 | lblTitle | To label the header which is the Pending Orders |
| Table 1 | dgvPendingOrders | To display all the pending orders in a tabular form |
| Button 1 | btnBack | To go back to the previous page |

Sample UI :

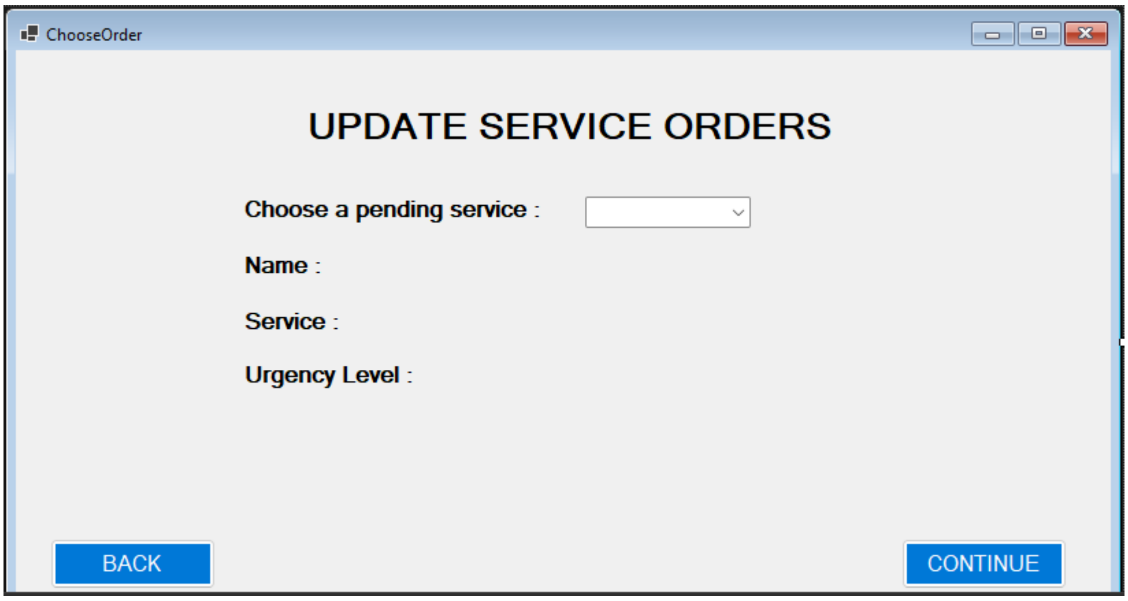


Update Service Orders (Ariel)

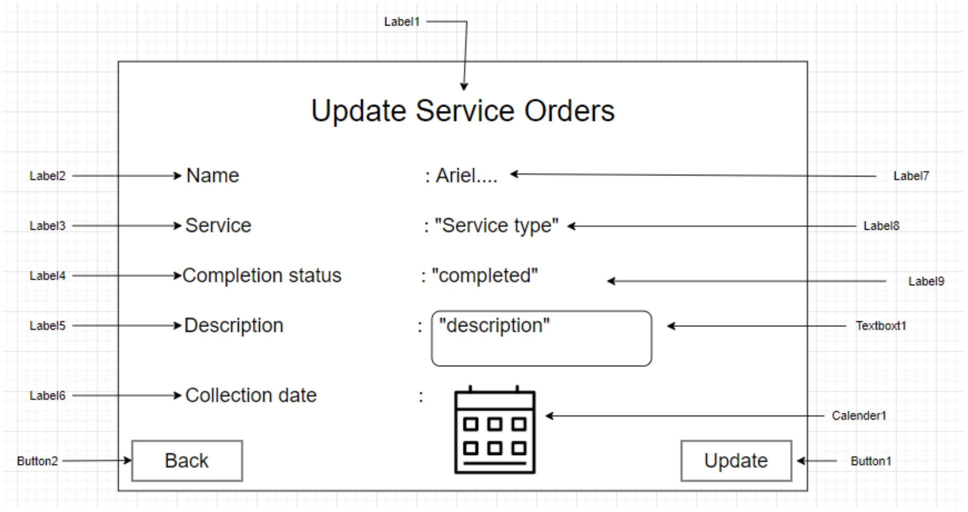


| Control | Control name | Description |
|---|---|---|
| Label 1 | lblTitle | To label the header which is the "Update Service Orders" |
| Label 2 | lblPendingService | To label the related control which is ComboBox1 |
| Label 3 | lblName | To label the related control which is Label6 |
| Label 4 | lblService | To label the related control which is Label7 |
| Label 5 | lblUrgency | To label the related control which is Label8 |
| Label 6 | lblNameResult | To display the selected order's customer name |
| Label 7 | lblServiceResult | To display the selected order's service type |
| Label 8 | lblUrgencyResult | To display the selected order's urgency level |
| ComboBox1 | cmbPendingService | To display different pending services for users to select and show its details |
| Button 1 | btnContinue | To confirm the selected order is correct and goes into the next form |
| Button 2 | btnBack | To go back to the previous page |

Sample UI :



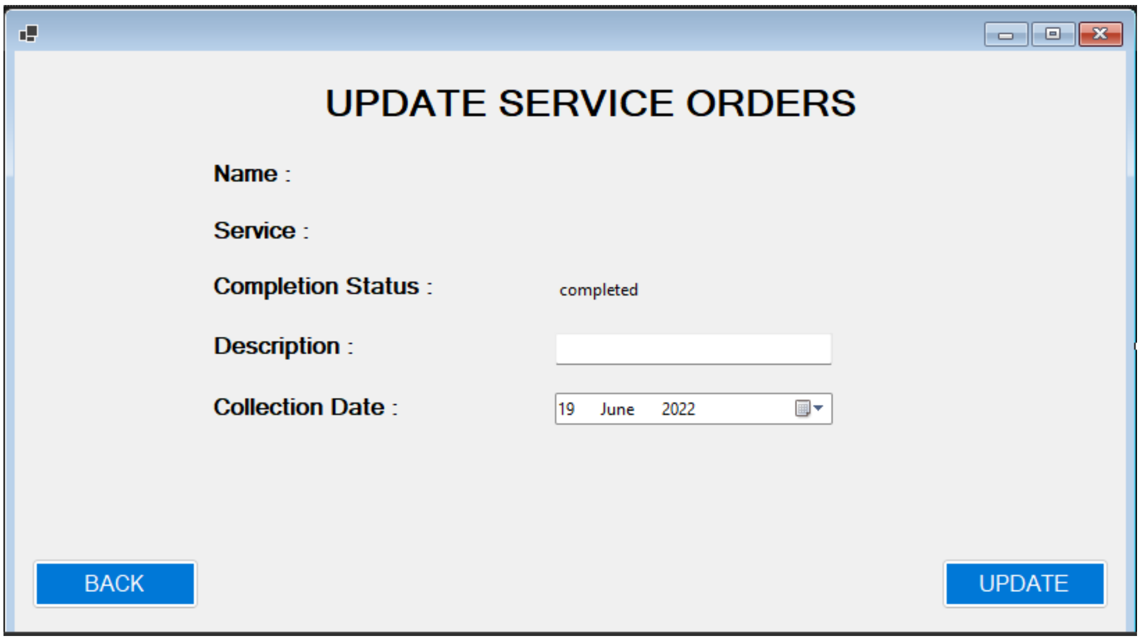
Update Service Orders Continuation (Ariel)



| Control | Control name | Description |
|---|---|---|
| Label 1 | lblTitle | To label the header which is the “Update Service Orders” |
| Label 2 | lblName | To label the related control which is Label7 |
| Label 3 | lblService | To label the related control which is Label8 |
| Label 4 | lblStatus | To label the related control which is ComboBox1 |
| Label 5 | lblDescription | To label the related control which is Textbox1 |
| Label 6 | lblCollection | To label the related control which is Calender1 |
| Label 7 | lblNameDetail | To display the name of the customer for this order |
| Label 8 | lblServiceDetail | To display the service type for this order |
| Label 9 | lblCompleted | To display that the status is completed |

| | | |
|---|---|---|
| TextBox1 | txtDescription | To allow users to add description to the order |
| Calender 1 | dtpCollectionDate | To allow users to select a collection date from the calendar |
| Button 1 | btnUpdate | To confirm the data and updates the service orders based on the given data |
| Button 2 | BtnBack | To go back to previous page |

Sample UI :


UPDATE SERVICE ORDERS

Name :

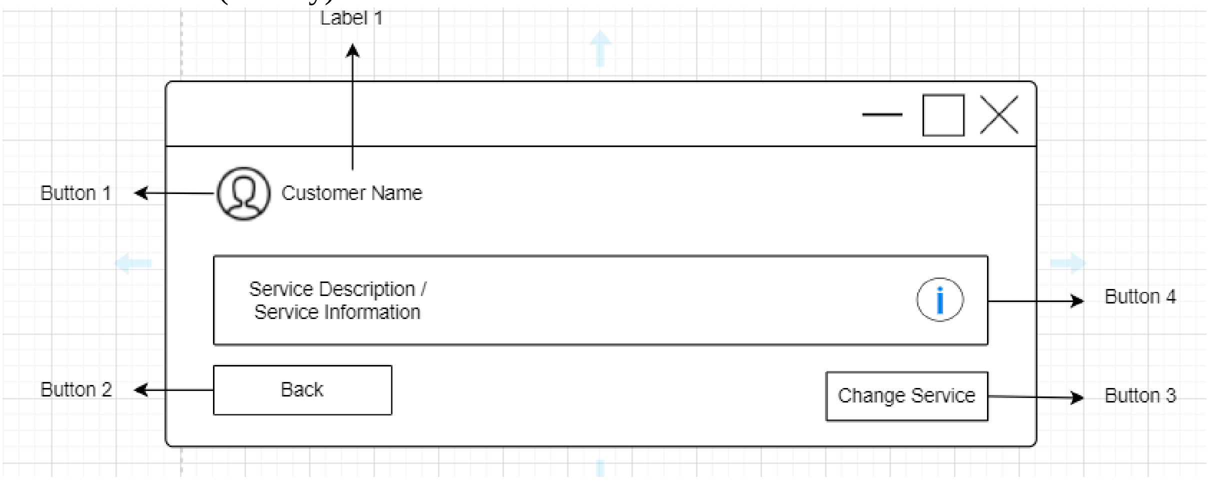
Service :

Completion Status : completed

Description :

Collection Date : 19 June 2022

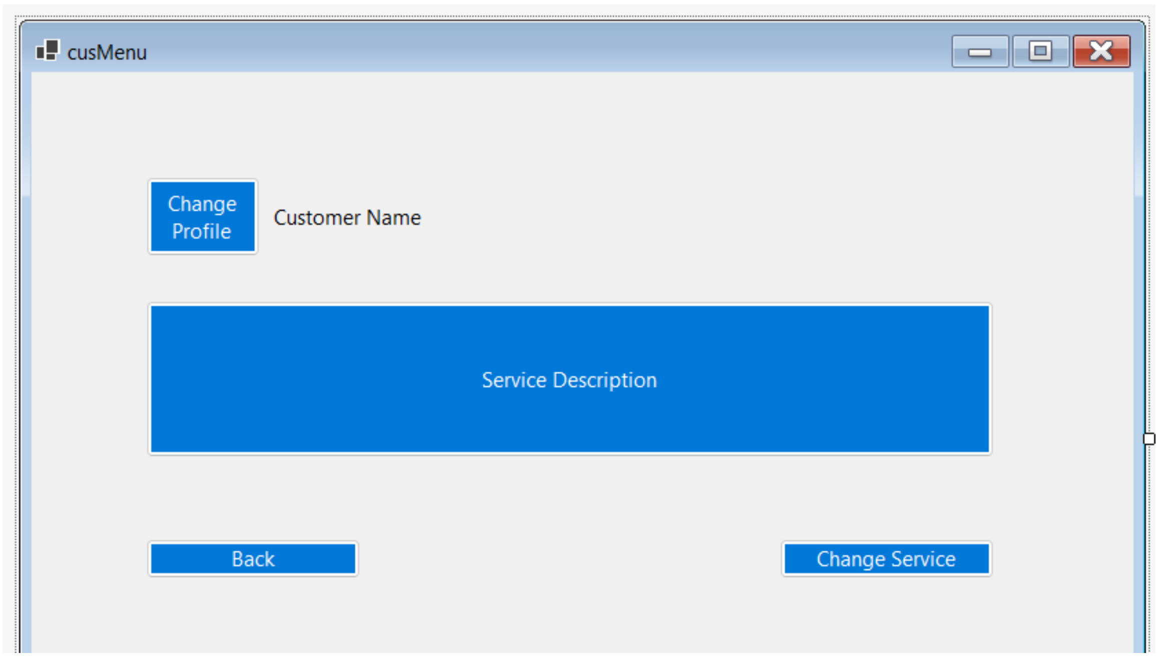
Customer Menu (Stanley)



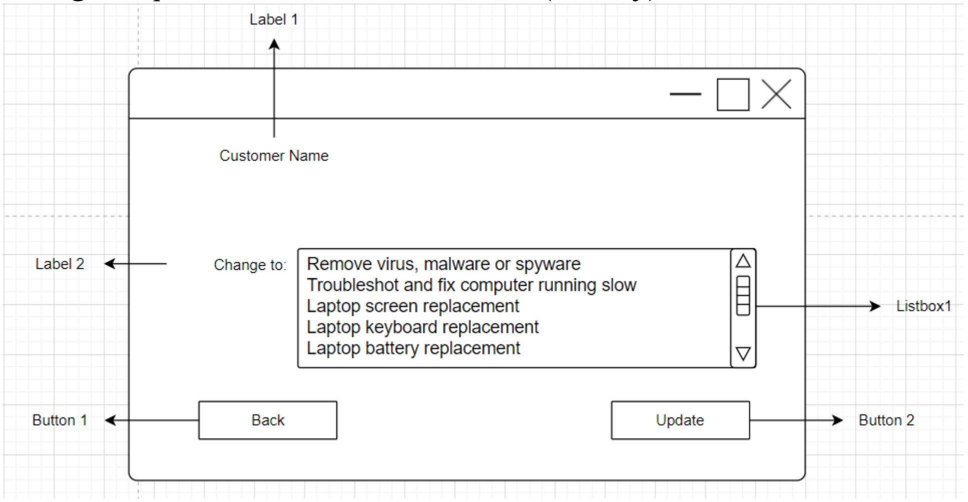
| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | cusName | Display customer name |
| Button 1 | cusProfile | Profile button for customer |

| | | |
|---|---|---|
| Button 2 | cusBack1 | Back button for customer |
| Button 3 | cusChange | Button to change requested service |
| Button 4 | cusView | Button to view service description, laptop collection date and total amount to be paid |

Sample UI :



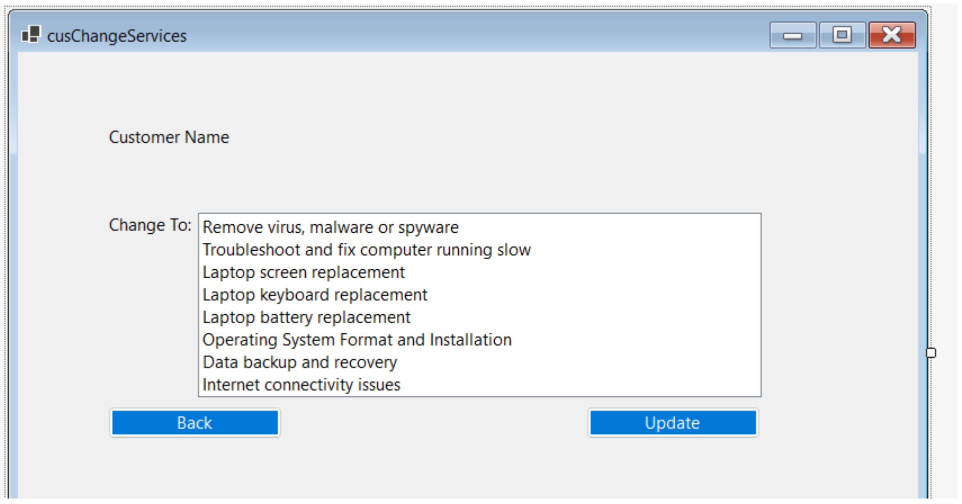
Change Requested Services Continuation (Stanley)


Label 1
Customer Name
Label 2
Change to:
Remove virus, malware or spyware
Troubleshot and fix computer running slow
Laptop screen replacement
Laptop keyboard replacement
Laptop battery replacement
Listbox1
Button 1
Back
Update
Button 2

| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | cusName | Display customer name |
| Label 2 | cusChangeTo | Display tittle in the form |
| Listbox 1 | cusChangeService | Listbox to change requested service |

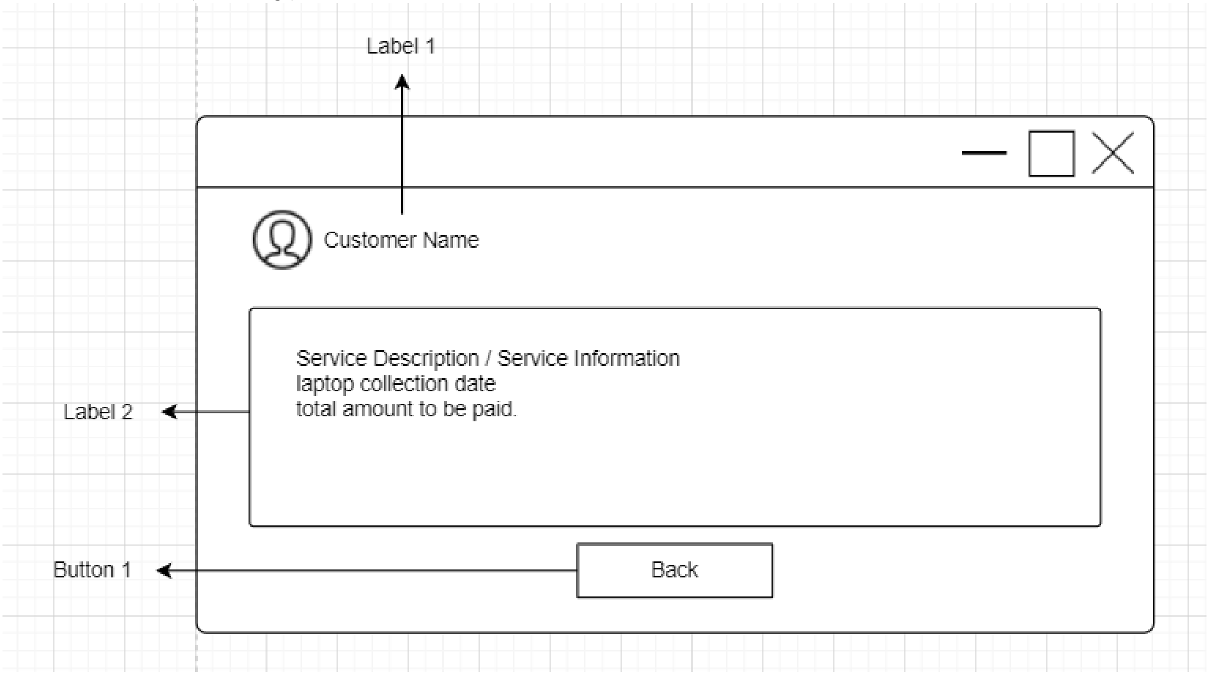
| | | |
|---|---|---|
| Button 1 | cusBack3 | Back button for customer |
| Button 2 | cusConfirm | Confirm button for customer to change requested service |

Sample UI :



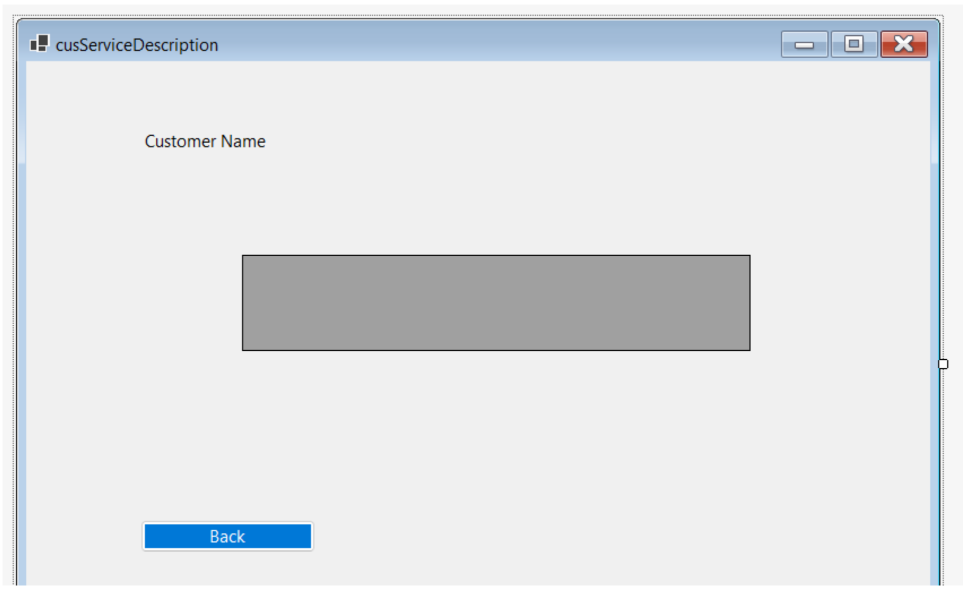
Remove virus, malware or spyware
Troubleshoot and fix computer running slow
Laptop screen replacement
Laptop keyboard replacement
Laptop battery replacement
Operating System Format and Installation
Data backup and recovery
Internet connectivity issues

View Service (Stanley)

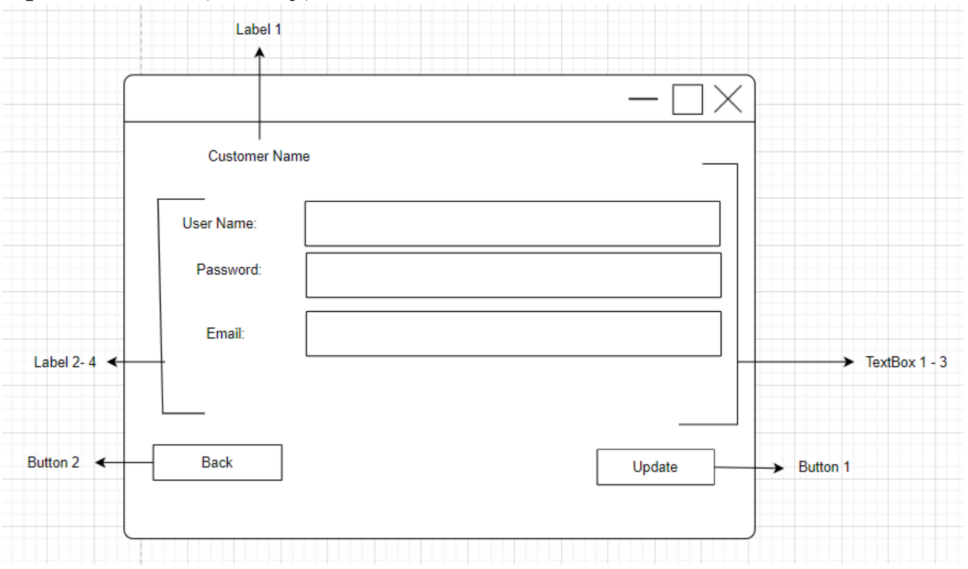


| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | cusName | Display customer name |
| Label 2 | cusServiceDesc | Display service description, laptop collection date and total amount to be paid |
| Button 1 | cusBack2 | Back button for customer |

Sample UI :



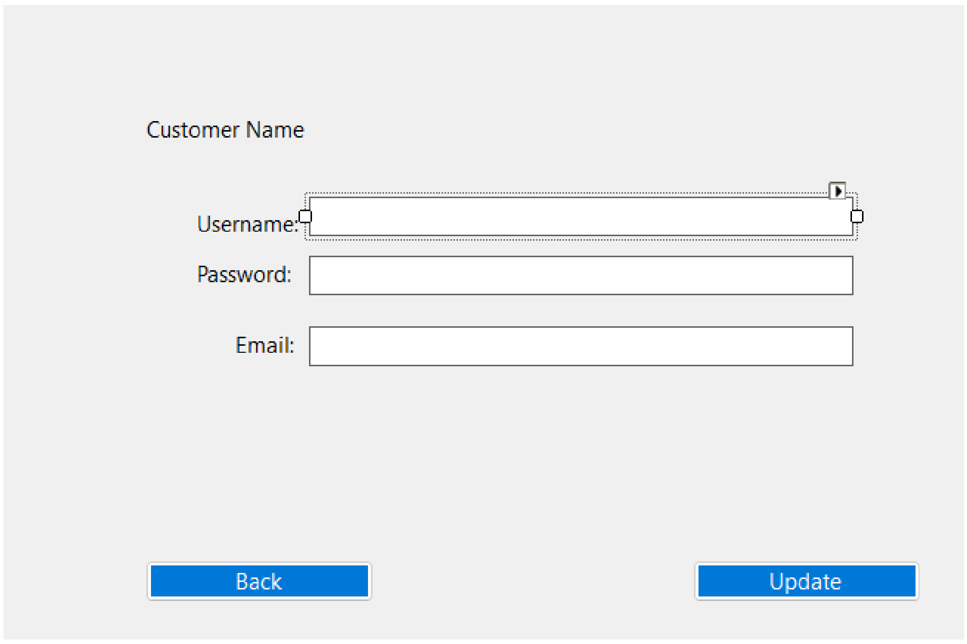
Update Profile (Stanley)



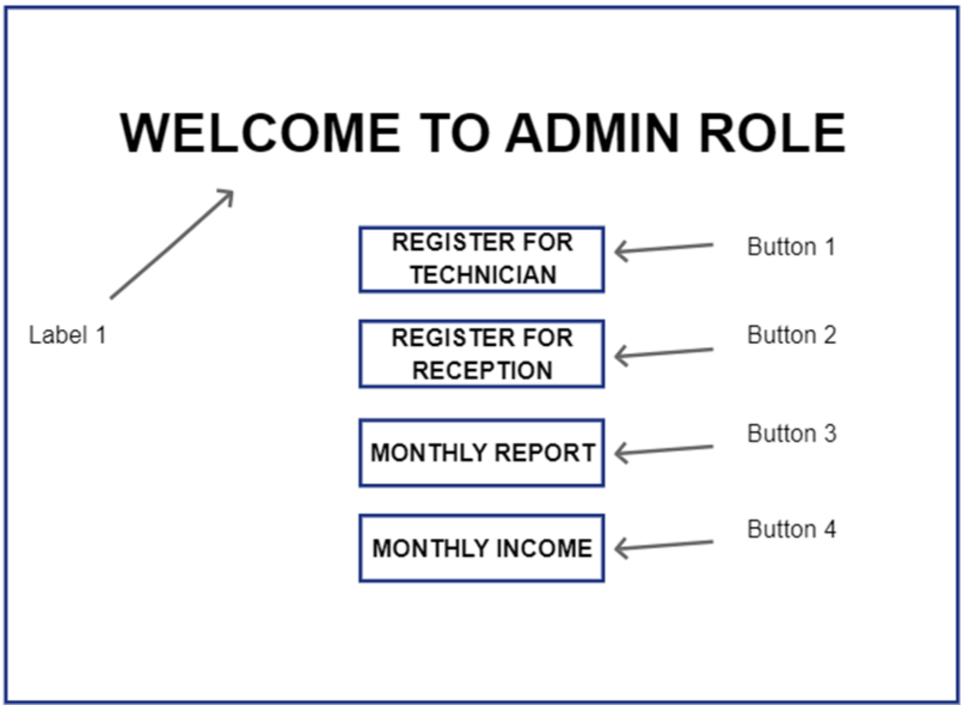
| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | cusName | Display customer name |
| Label 2 | cusUserName | Display customer user name |
| Label 3 | cusPassword | Display customer passsword |
| Label 4 | cusEmail | Day pertama Email |
| Textbox 1 | usernameTxt | A place to fill username |
| Textbox 2 | passwordTxt | A place to fill password |
| Textbox 3 | emailTxt | A place to fill email |

| | | |
|---|---|---|
| | | |
| Button 1 | updateBtn | Update the data |
| Button 2 | backBtn | Back to customer |

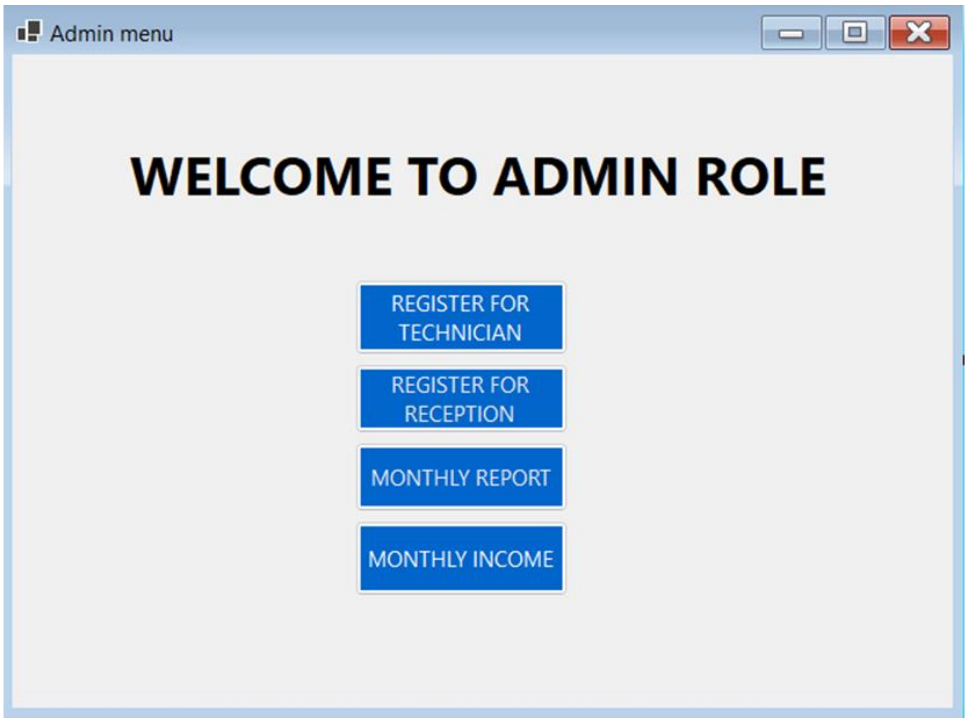
Sample UI :



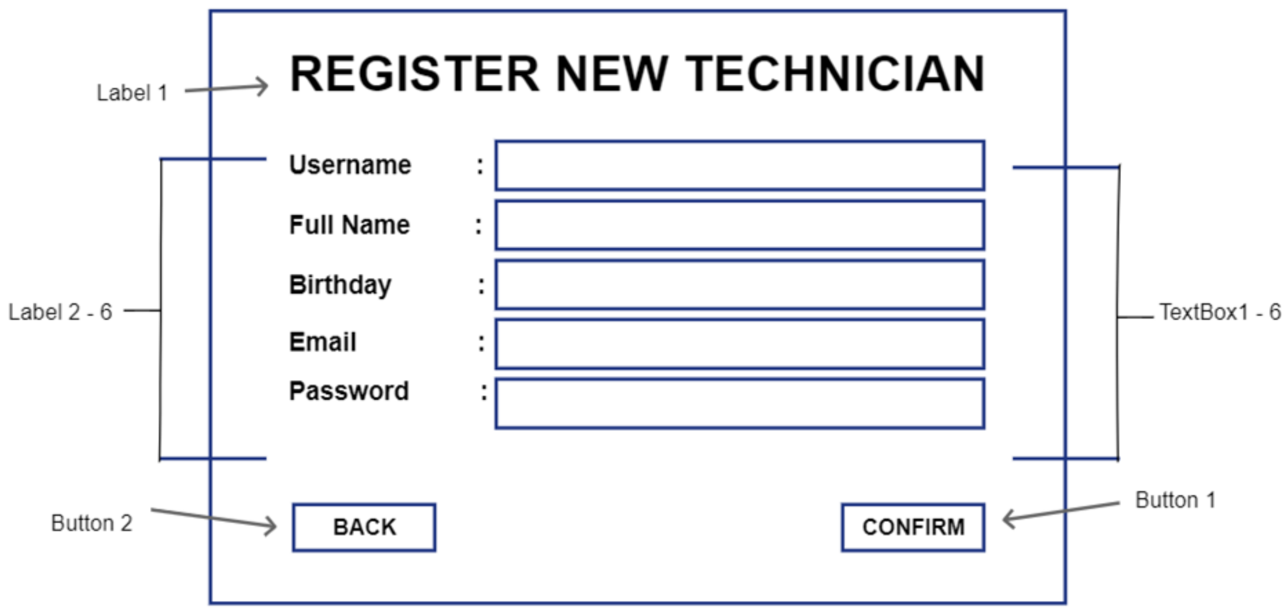
**User Interface for Admin Menu**



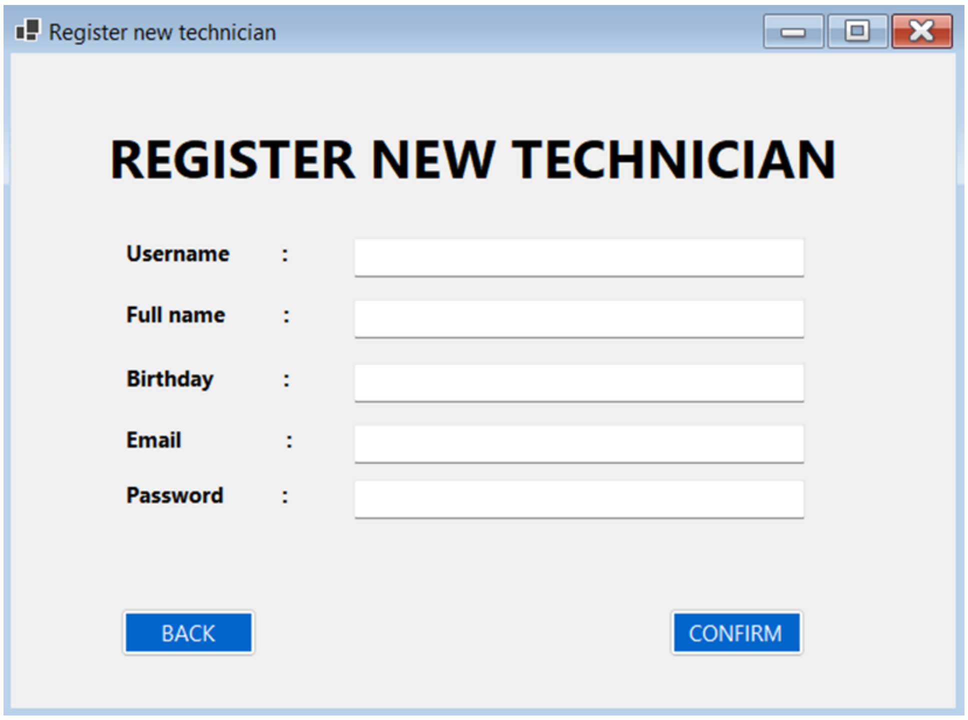
| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | welLbl | Display tittle in the form |
| Button 1 | RegTBtn | Register button for technician |
| Button 2 | RegRBtn | Register button for reception |
| Button 3 | MrepBtn | Button to check monthly service report |
| Button 4 | MinBtn | Button to view monthly income |



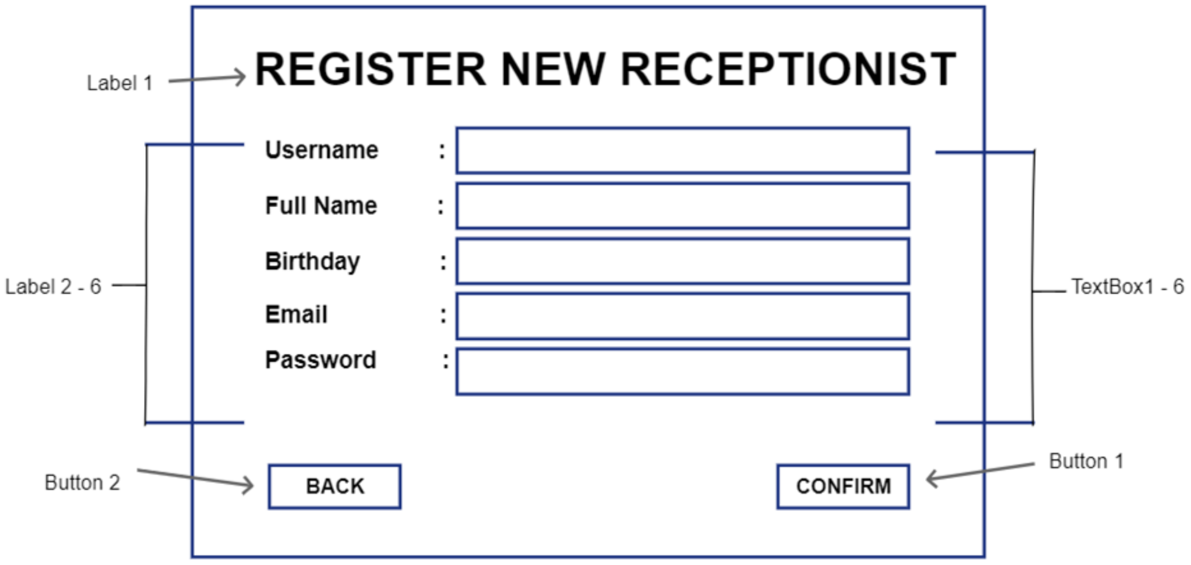
**User Interface for Register New Technician (Nathaniel)**



| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | RegLbl | Display tittle in the form |
| Label 2 | userLbl | Display a name to inform user to input username beside |
| Label 3 | NameLbl | Display a name to inform user to input full name beside |
| Label 4 | BirthLbl | Display a name to inform user to input birthday beside |
| Label 5 | MailLbl | Display a name to inform user to input email number beside |
| Label 6 | PassLbl | Display a name to inform user to input password beside |
| | | |
| TextBox1 | UserTxt | a place to fill the username |
| TextBox2 | NameTxt | a place to fill the full name |
| TextBox3 | BirthTxt | a place to fill the birthday |
| TextBox4 | MailTxt | a place to fill the email |
| TextBox5 | NumTxt | a place to fill the phone number |
| TextBox6 | PassTxt | a place to fill the password |
| Button 1 | ConBtn | To confirm after input all the data |
| Button 2 | BackBtn | go back to previous page |

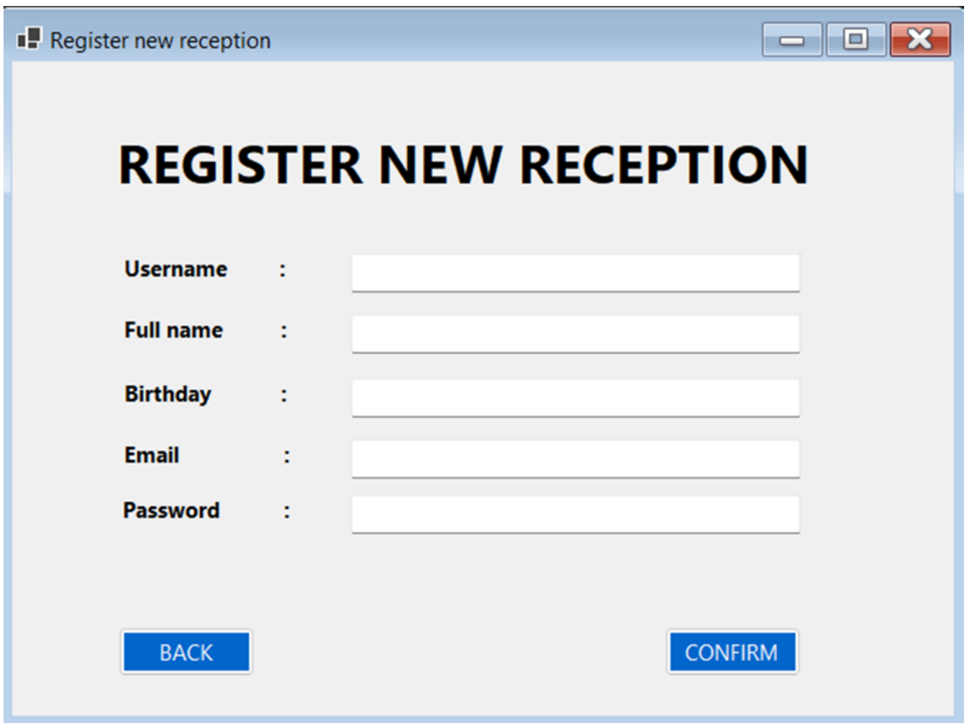


**<u>User Interface for Register New Reception (Natthaniel)</u>**

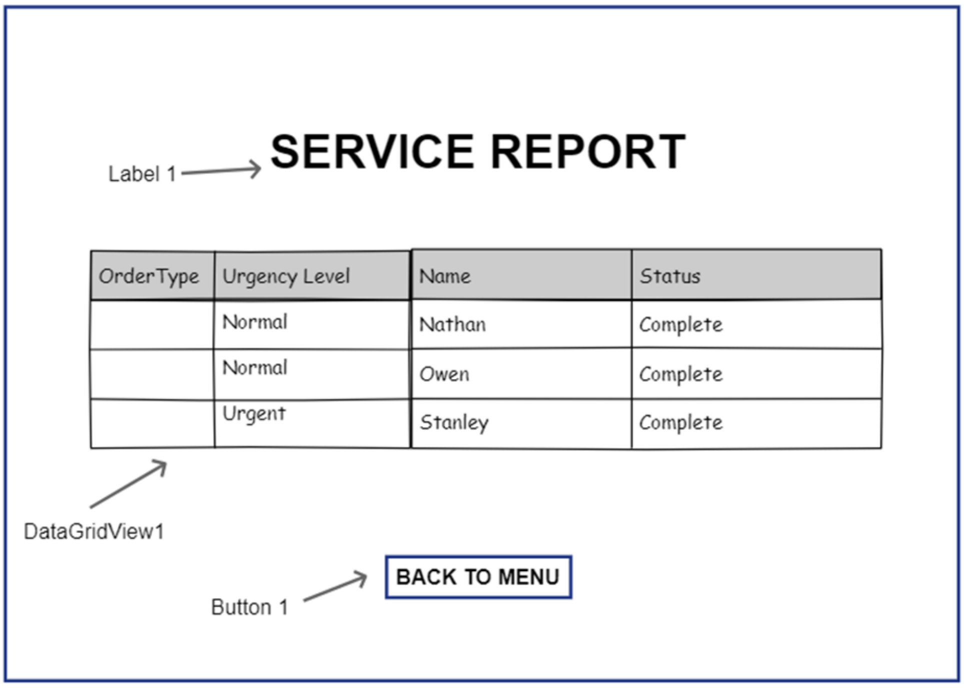


| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | Reg2Lbl | Display tittle in the form |
| Label 2 | User2Lbl | Display a name to inform user to input username beside |
| Label 3 | Name2Lbl | Display a name to inform user to input full name beside |
| Label 4 | Birth2Lbl | Display a name to inform user to input birthday beside |

| | | |
|---|---|---|
| Label 5 | Mail2Lbl | Display a name to inform user to input email number beside |
| Label 6 | Pass2Lbl | Display a name to inform user to input password beside |
| | | |
| TextBox1 | User2Txt | a place to fill the username |
| TextBox2 | Name2Txt | a place to fill the full name |
| TextBox3 | Birth2Txt | a place to fill the birthday |
| TextBox4 | Mail2Txt | a place to fill the email |
| TextBox5 | Num2Txt | a place to fill the phone number |
| TextBox6 | Pass2Txt | a place to fill the password |
| Button 1 | Con2Btn | To confirm after input all the data |
| Button 2 | Back2Btn | go back to previous page |

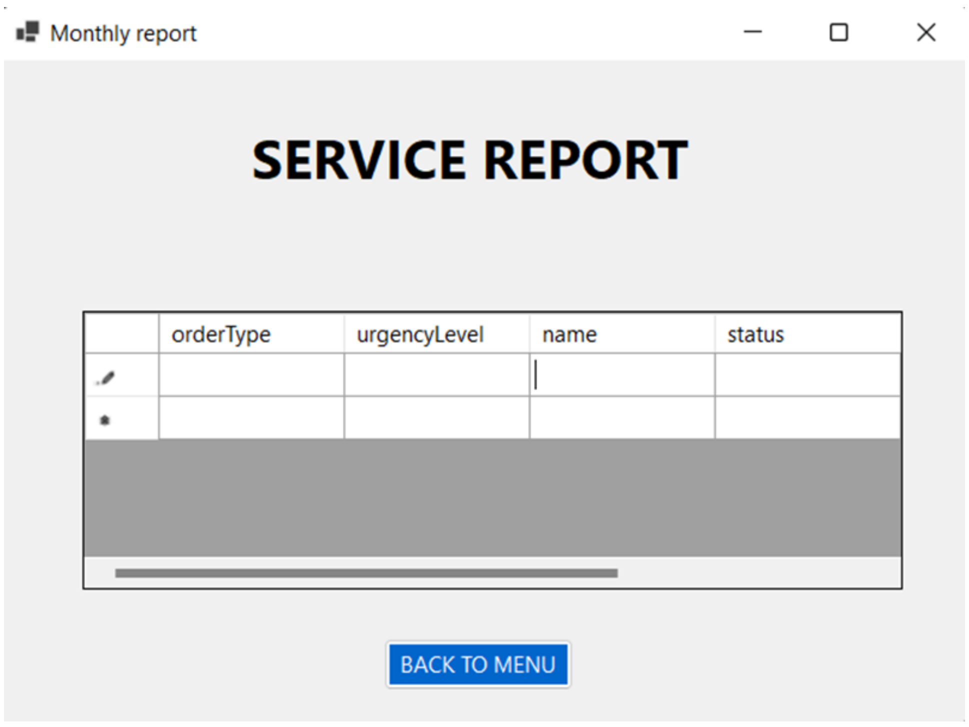


## User Interface for Monthly Report (Nathaniel)

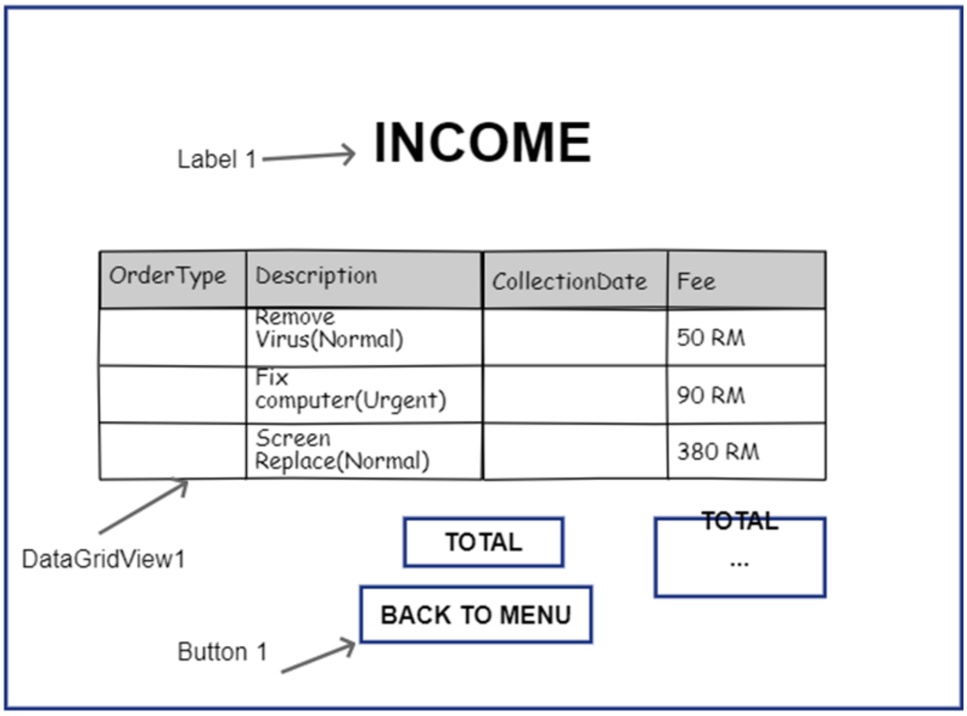


| OrderType | Urgency Level | Name | Status |
|---|---|---|---|
| | Normal | Nathan | Complete |
| | Normal | Owen | Complete |
| | Urgent | Stanley | Complete |

| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | ServLbl | Display tittle in the form |
| DataGridView 1 | DataTable | Display table to see a data of service report |
| Button 1 | BckBtn | Go back to main page menu |

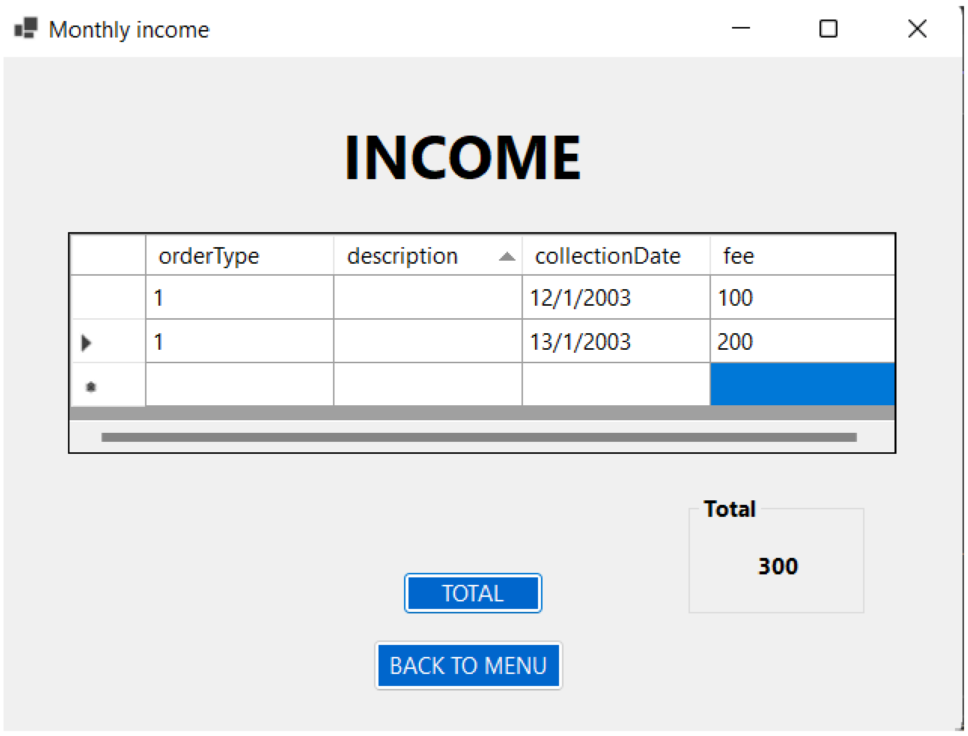


**User Interface for Monthly Income(Nathaniel)**



| OrderType | Description | CollectionDate | Fee |
|---|---|---|---|
| | Remove Virus(Normal) | | 50 RM |
| | Fix computer(Urgent) | | 90 RM |
| | Screen Replace(Normal) | | 380 RM |

| Toolbox | Design name | Explanation |
|---|---|---|
| Label 1 | Serv2Lbl | Display tittle in the form |
| DataGridView 1 | Data2Table | Display table to see a data of income |
| Button 1 | Bck2Btn | Go back to main page menu |
| Button 2 | TotalBtn | Total button to sum the fee in label 2 |
| ComboBox1 | Sign | To show a sign |
| Label 2 | Total | To show a total fee |
| | | |


Monthly income

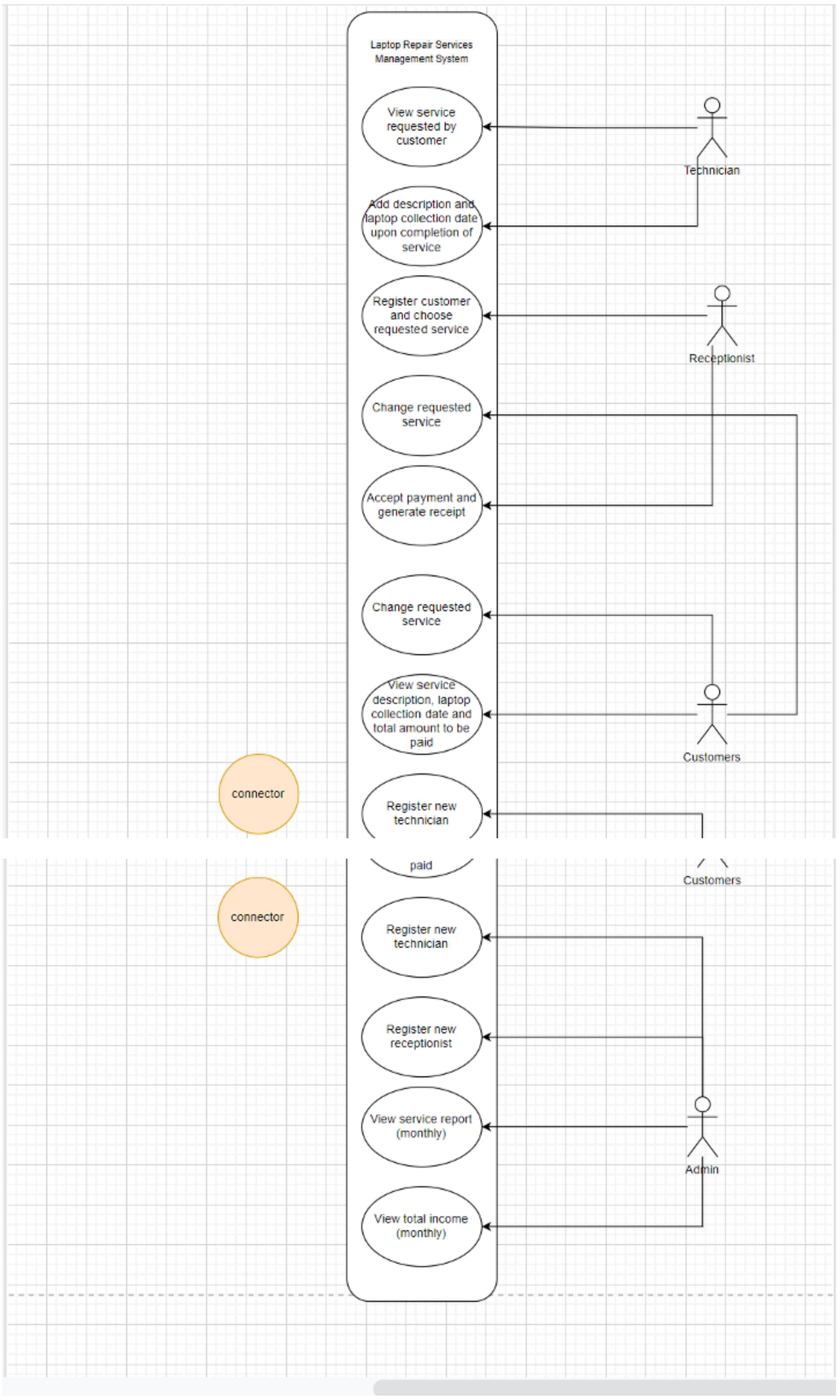
INCOME

| | orderType | description | collectionDate | fee |
|---|---|---|---|---|
| | 1 | | 12/1/2003 | 100 |
| ▸ | 1 | | 13/1/2003 | 200 |
| * | | | | |

Total

300

## Use-Case Diagram


Laptop Repair Services Management System
View service requested by customer
Add description and laptop collection date upon completion of service
Register customer and choose requested service
Change requested service
Accept payment and generate receipt
Change requested service
View service description, laptop collection date and total amount to be paid
Register new technician
Register new receptionist
View service report (monthly)
View total income (monthly)
Technician
Receptionist
Customers
Admin
connector

## Class Diagram

| Technician |
|---|

private string username;

private string Fullname;

private string Birthday;

private string Email;

private string Password;

public Technician(string u, string f, string b, string e, string pn)

public string RegisterTechnician()

**Receptionist**

private string username;

private string Fullname;

private string Birthday;

private string Email;

private string Password;

public Receptionist(string u, string f, string e, string b, string pn)

public string RegisterReceptionist()

**Customers**

private string username;

private string password;

private string fullName;

private string email;

private string birthday;

private string services;

private string urgency;

public Customers(string u, string p, string f, string e, string b, string s, string description)

public Customers(string s, string u)

public string changeServices()

public string addCustomers()

**Admin**

public string income()

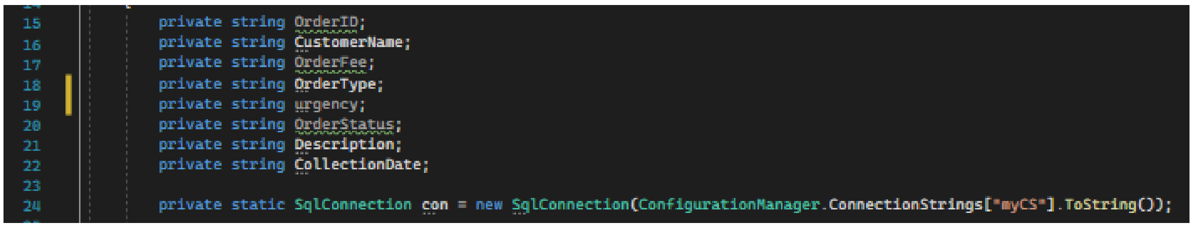
| **User** |
|---|
| private string username; |
| private string password; |
| private string fullName; |
| private string email; |
| private string birthday; |
| private string Oldusername; |
| public User(string un, string p) |
| public User(string u, string p, string e, string ou) |
| public string login(string un) |
| public string updateProfile() |

| **Order** |
|---|
| private string OrderID; |
| private string CustomerName; |
| private string OrderFee; |
| private string OrderType; |
| private string urgency; |
| private string OrderStatus; |
| private string Description; |
| private string CollectionDate; |
| public Order(string cn, string ot, string d, string cd) |
| public static DataTable ViewPendingOrders() |
| public static ArrayList TakePendingOrders() |
| public static ArrayList ShowDetails(char selectedOrder) |
| public string UpdateOrder() |
| public static DataTable ViewService(string username) |

## Code Explanation

### Order.cs (Ariel)

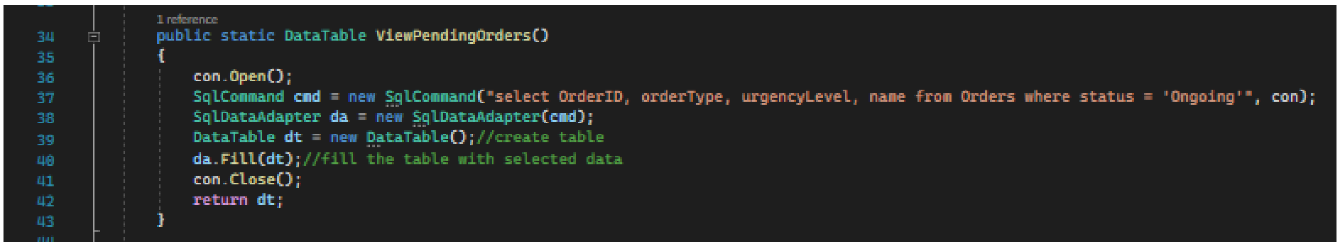
The Order class is a class that handles the methods, contructors, object creations of thing related to orders done by the customers.



```
15          private string OrderID;
16          private string CustomerName;
17          private string OrderFee;
18          private string OrderType;
19          private string urgency;
20          private string OrderStatus;
21          private string Description;
22          private string CollectionDate;
23
24          private static SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["myCS"].ToString());
```

It starts with the creation of attributes which is shown from line 15 – 22 with the attributes related to objects from order class being OrderID, CustomerName, OrderFee, OrderType, urgency, OrderStatus, Description, and CollectionDate. It is then connected to the database by using a connection string and store it in variable con in line 24.

1. View Pending Orders

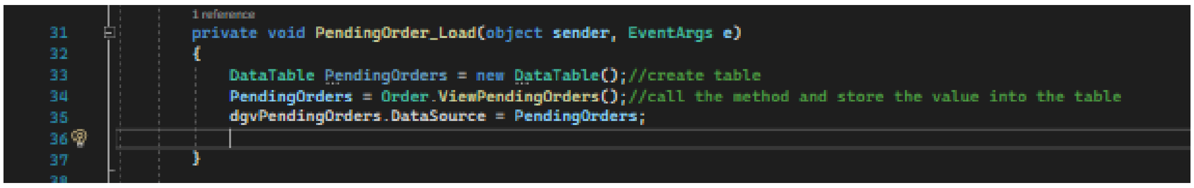


```
            1 reference
34          public static DataTable ViewPendingOrders()
35          {
36              con.Open();
37              SqlCommand cmd = new SqlCommand("select OrderID, orderType, urgencyLevel, name from Orders where status = 'Ongoing'", con);
38              SqlDataAdapter da = new SqlDataAdapter(cmd);
39              DataTable dt = new DataTable();//create table
40              da.Fill(dt);//fill the table with selected data
41              con.Close();
42              return dt;
43          }
```

The method used by the technician to view the pending orders is ViewPendingOrders().

| Code Line | Code Explanation |
|---|---|
| 34 | Create a public static and DataTable method called ViewPendingOrders() |
| 36 | Open connection to the database |
| 37 | Use a database command to select colums OrderID, orderType, urgencyLevel, and name from Orders table with the condition the status being "ongoing" |
| 38 | Create object da from sqlDataAdapater class with the argument cmd |
| 39 | Create a table named dt |
| 40 | Fills the table with the selected data from the database |
| 41 | Close connection to the database |
| 42 | Return table dt to the caller |

The method was called from the PendingOrder form

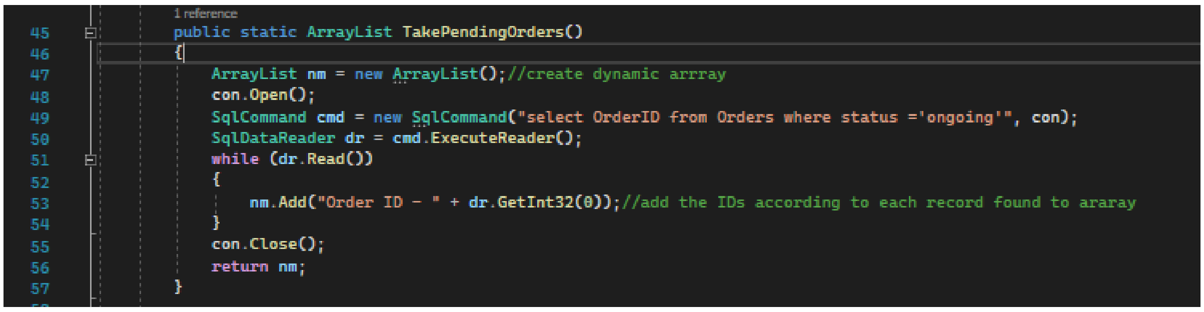


```
            1 reference
31          private void PendingOrder_Load(object sender, EventArgs e)
32          {
33              DataTable PendingOrders = new DataTable();//create table
34              PendingOrders = Order.ViewPendingOrders();//call the method and store the value into the table
35              dgvPendingOrders.DataSource = PendingOrders;
36
37          }
```

| Code Line | Code Explanation |
|---|---|
| 31 | Method to store actions for when the form is loaded |

| | |
|---|---|
| 33 | Create table PendingOrders |
| 34 | Call the method ViewPendingOrders() and store the value into the table PendingOrders |
| 35 | Fill the data grid view in the form with the contents from table PendingOrders |

2. Take pending orders

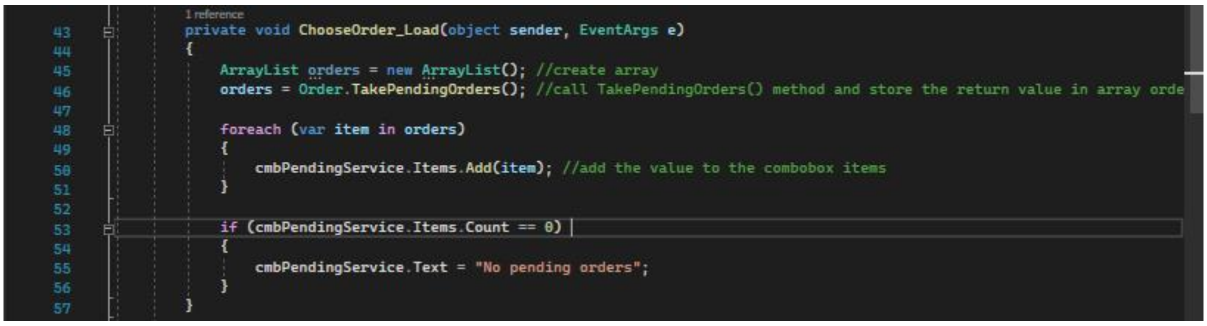


```
        1 reference
45      public static ArrayList TakePendingOrders()
46      {
47          ArrayList nm = new ArrayList();//create dynamic arrray
48          con.Open();
49          SqlCommand cmd = new SqlCommand("select OrderID from Orders where status ='ongoing'", con);
50          SqlDataReader dr = cmd.ExecuteReader();
51          while (dr.Read())
52          {
53              nm.Add("Order ID - " + dr.GetInt32(0));//add the IDs according to each record found to araray
54          }
55          con.Close();
56          return nm;
57      }
```

The method used by the technician to find the orders that are still pending is the TakePendingOrders() method.

| Code Line | Code Explanation |
|---|---|
| 45 | Create a public static and ArrayList method called TakePendingOrders() |
| 47 | Create a dynamic array called nm |
| 48 | Open connection to the database |
| 49 | Use a database command to select colums OrderID from Orders table with the condition the status being "ongoing" |
| 50 | Create object dr from sqlDataReader class |
| 51 - 54 | do a while loop for whenever a line is read, add the extracted orderid by the query to the array and add "Order ID - " in front the string word |
| 55 | Close connection to the database |
| 42 | Return array nm to the caller |

This method is called from the ChooseOrder Form. It is called when the form is loaded to store the available order ids first into th combobox before the user selects it.

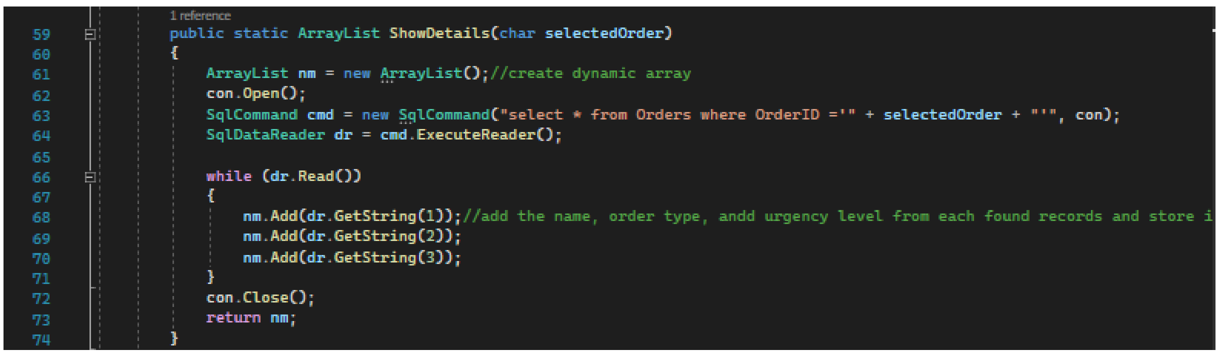


```
1 reference
private void ChooseOrder_Load(object sender, EventArgs e)
{
    ArrayList orders = new ArrayList(); //create array
    orders = Order.TakePendingOrders(); //call TakePendingOrders() method and store the return value in array orde

    foreach (var item in orders)
    {
        cmbPendingService.Items.Add(item); //add the value to the combobox items
    }

    if (cmbPendingService.Items.Count == 0)
    {
        cmbPendingService.Text = "No pending orders";
    }
}
```

| Code Line | Code Explanation |
|---|---|
| 45 | Create a dynamic array called orders |
| 48 | Call the method TakePendingOrders() and store the value into the array orders |
| 48 - 51 | For each item in the array orders, add the value to the cmbPendingService ComboBox. |
| 53 - 54 | If there is no items in the comboBox or there is no pending orders at the moment, show the text "No pending orders" in the combobox |

3. Show Order Details

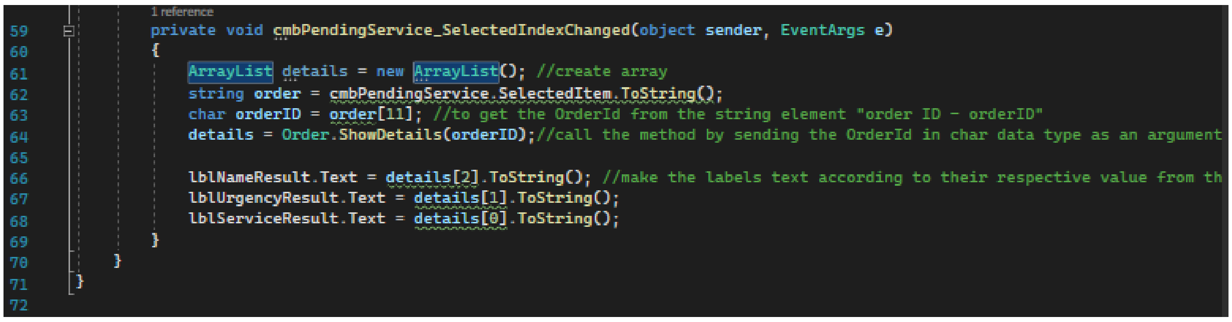
The ShowDetails()method is used by the technician to show the order details from the orderid the user selected.



```
        1 reference
59      public static ArrayList ShowDetails(char selectedOrder)
60      {
61          ArrayList nm = new ArrayList();//create dynamic array
62          con.Open();
63          SqlCommand cmd = new SqlCommand("select * from Orders where OrderID ='" + selectedOrder + "'", con);
64          SqlDataReader dr = cmd.ExecuteReader();
65
66          while (dr.Read())
67          {
68              nm.Add(dr.GetString(1));//add the name, order type, andd urgency level from each found records and store i
69              nm.Add(dr.GetString(2));
70              nm.Add(dr.GetString(3));
71          }
72          con.Close();
73          return nm;
74      }
```

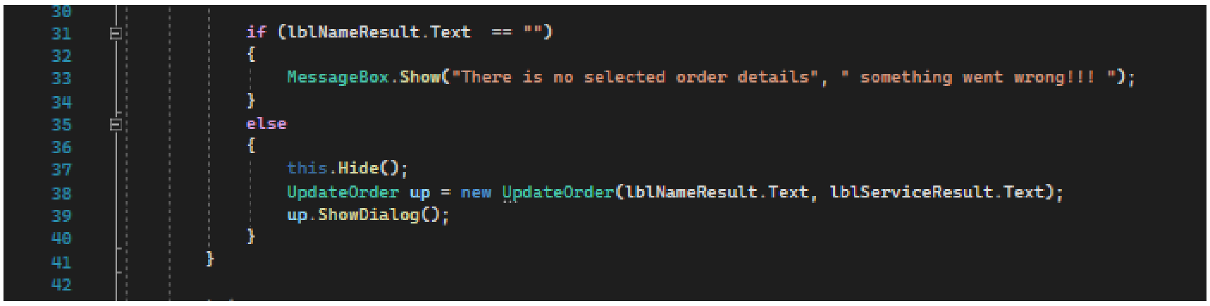
| Code Line | Code Explanation |
|---|---|
| 59 | Create a public static and ArrayList method called ShowDetails() receiving one parameter which is char selectedOrder |
| 61 | Create a dynamic array called nm |
| 62 | Open connection to the database |
| 63 | Use a database command to select all colums from Orders table with the condition the Order ID being the id the user selected from the combobox |
| 64 | Create object dr from sqlDataReader class |
| 66-71 | do a while loop for whenever a line is read, add the extracted name, order type, and urgency level by the query to the array |
| 72 | Close connection to the database |

The method is also called from ChooseOrder Form when the user picks an option from the combobox



```
                1 reference
59              private void cmbPendingService_SelectedIndexChanged(object sender, EventArgs e)
60              {
61                  ArrayList details = new ArrayList(); //create array
62                  string order = cmbPendingService.SelectedItem.ToString();
63                  char orderID = order[11]; //to get the OrderId from the string element "order ID - orderID"
64                  details = Order.ShowDetails(orderID);//call the method by sending the OrderId in char data type as an argument
65
66                  lblNameResult.Text = details[2].ToString(); //make the labels text according to their respective value from th
67                  lblUrgencyResult.Text = details[1].ToString();
68                  lblServiceResult.Text = details[0].ToString();
69              }
70          }
71      }
72
```

| Code Line | Code Explanation |
|---|---|
| 61 | Create a dynamic array called details |
| 62 | Create the local variable order and store the selected id from the combobox as a string |
| 63 | Get the orderid by selecting the character index from the string "order ID - ..." |
| 64 | Call the method ShowDetails() by sending one argument which is OrderID and store the value into the array details |
| 66-68 | Edit the text for lblNameResult, lblUrgencyResult, lblServiceResult to store the values from array in their respective index to show the details of the selected orders. |

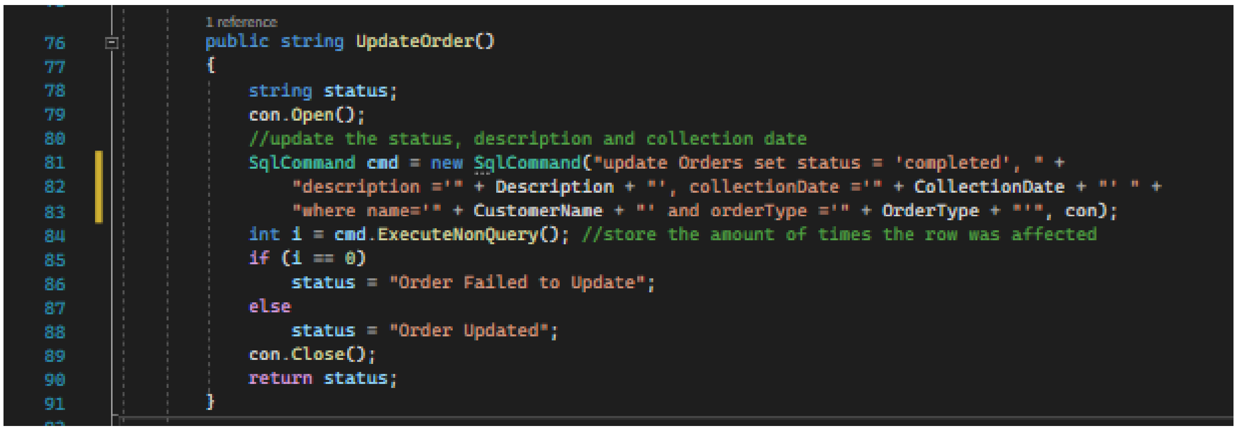


```
30
31                  if (lblNameResult.Text  == "")
32                  {
33                      MessageBox.Show("There is no selected order details", " something went wrong!!! ");
34                  }
35                  else
36                  {
37                      this.Hide();
38                      UpdateOrder up = new UpdateOrder(lblNameResult.Text, lblServiceResult.Text);
39                      up.ShowDialog();
40                  }
41              }
42
```

There is an exception handling to expect user's misinput before proceeding to the next form.

| Code Line | Code Explanation |
|---|---|
| 31 - 34 | If the lblNameResult has no text or value, then a message box will show saying that they didn’t pick any orders and ask them to select it again |
| 36 - 40 | If there is no issues, then the process proceeds by hiding this form, and then going to the UpdateOrder Form which sends two arguments which is lblNameResult.Text and lblServiceResult.Text |

4. Update orders

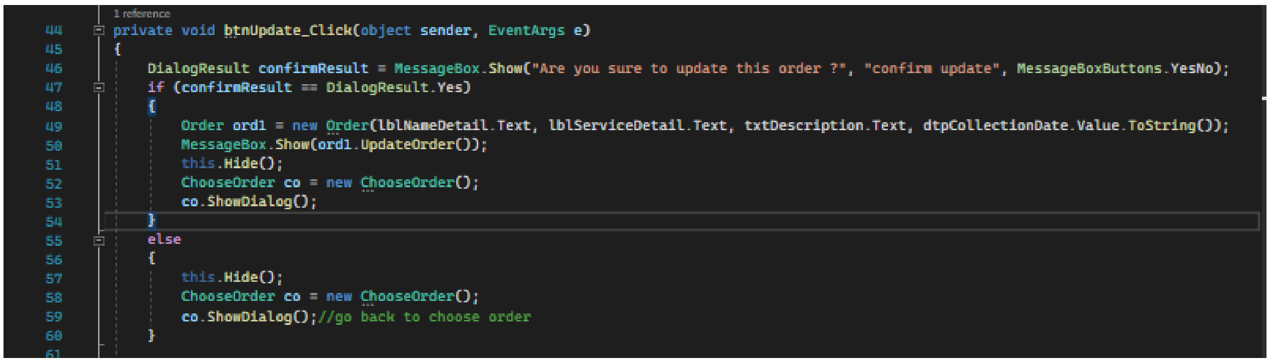


```
        1 reference
76      public string UpdateOrder()
77      {
78          string status;
79          con.Open();
80          //update the status, description and collection date
81          SqlCommand cmd = new SqlCommand("update Orders set status = 'completed', " +
82              "description ='" + Description + "', collectionDate ='" + CollectionDate + "' " +
83              "where name='" + CustomerName + "' and orderType ='" + OrderType + "'", con);
84          int i = cmd.ExecuteNonQuery(); //store the amount of times the row was affected
85          if (i == 0)
86              status = "Order Failed to Update";
87          else
88              status = "Order Updated";
89          con.Close();
90          return status;
91      }
```

The UpdateOrder()method is used by the technician to update the order that is selected.

| Code Line | Code Explanation |
|---|---|
| 76 | Create a public string method called UpdateOrder() |
| 78 | Create a local variable called status |
| 79 | Open connection to the database |
| 81-83 | Use a database command to update status to "sompleted", desription to the description given, collectiondate to the date given from Orders table with the condition the name being the user's name and the ordertype is the one shown by the selected details in the previous form. |
| 84 | Execute the query and store the return value in a variable called i |
| 85-88 | If i is 0, the status is "order failed to update" else the status "order updated" |
| 89 | Close connection to the database |
| 90 | Return status to the caller |

The method is called from the UpdateOrder Form alongside a confirmation box to ask the user for confirmation.



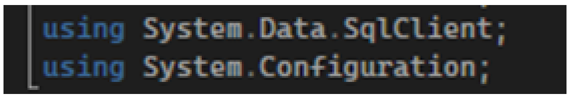
```
        1 reference
44      private void btnUpdate_Click(object sender, EventArgs e)
45      {
46          DialogResult confirmResult = MessageBox.Show("Are you sure to update this order ?", "confirm update", MessageBoxButtons.YesNo);
47          if (confirmResult == DialogResult.Yes)
48          {
49              Order ord1 = new Order(lblNameDetail.Text, lblServiceDetail.Text, txtDescription.Text, dtpCollectionDate.Value.ToString());
50              MessageBox.Show(ord1.UpdateOrder());
51              this.Hide();
52              ChooseOrder co = new ChooseOrder();
53              co.ShowDialog();
54          }
55          else
56          {
57              this.Hide();
58              ChooseOrder co = new ChooseOrder();
59              co.ShowDialog();//go back to choose order
60          }
```

| Code Line | Code Explanation |
|---|---|

| | |
|---|---|
| 46 | Using the DialogResult Class, show a yes or no message box with the question "Are you sure to update this order?" With the box caption being "confirm update" |
| 47-54 | If the answer is yes, create the object ord 1 from the Order class sending 4 arguments so that in the next line, it can call the method  UpdateOrder(). And then hide the form and go back to ChooseOrder Form |
| 55-60 | If the answer is no, hide this form and go back to the ChooseOrder Form |

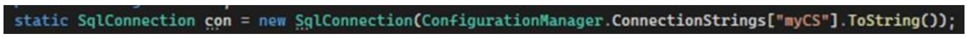
**Technician.cs (Nathaniel)**

Technician.cs is a class method in the program that contains all the field and function in our program that have spatial functions. Contains methods from the form of our program.



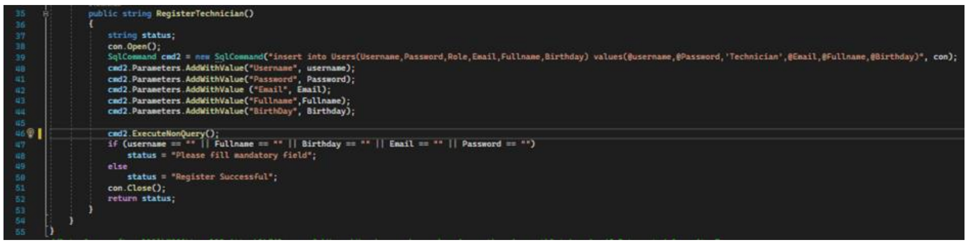
```
using System.Data.SqlClient;
using System.Configuration;
```

```
class Technician
```



```
static SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["myCS"].ToString());
```

To connect data to database, the system connects the whole class using the SQL Connection to our database file.



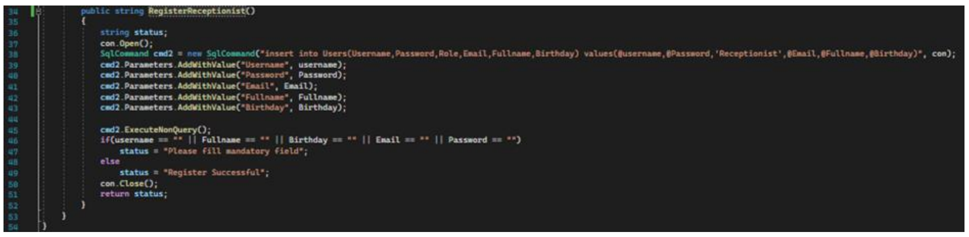
```
35          public string RegisterTechnician()
36          {
37              string status;
38              con.Open();
39              SqlCommand cmd2 = new SqlCommand("insert into Users(Username,Password,Role,Email,Fullname,Birthday) values(@username,@Password,'Technician',@Email,@Fullname,@Birthday)", con);
40              cmd2.Parameters.AddWithValue("Username", username);
41              cmd2.Parameters.AddWithValue("Password", Password);
42              cmd2.Parameters.AddWithValue ("Email", Email);
43              cmd2.Parameters.AddWithValue("Fullname",Fullname);
44              cmd2.Parameters.AddWithValue("BirthDay", Birthday);
45
46              cmd2.ExecuteNonQuery();
47              if (username == "" || Fullname == "" || Birthday == "" || Email == "" || Password == "")
48                  status = "Please fill mandatory field";
49              else
50                  status = "Register Successful";
51              con.Close();
52              return status;
53          }
54      }
55  }
```

| Code Line | Explanation |
|---|---|
| 37 | Create string to check the status of the form |
| 38 | Open the connection to the database |
| 39 | Create a database command to insert values to database |
| 40-44 | Assigning values to a SQL parameter with selected data |
| 46 | Execute the “cmd2” to insert the selected database |
| 47-50 | Execute the “cmd” to insert the selected database and gives value to “status” |
| 51 | Close the connection to the database |

| | |
|---|---|
| 52 | Return a value called “status” to where this method called |

**Receptionist.cs (Nathaniel)**

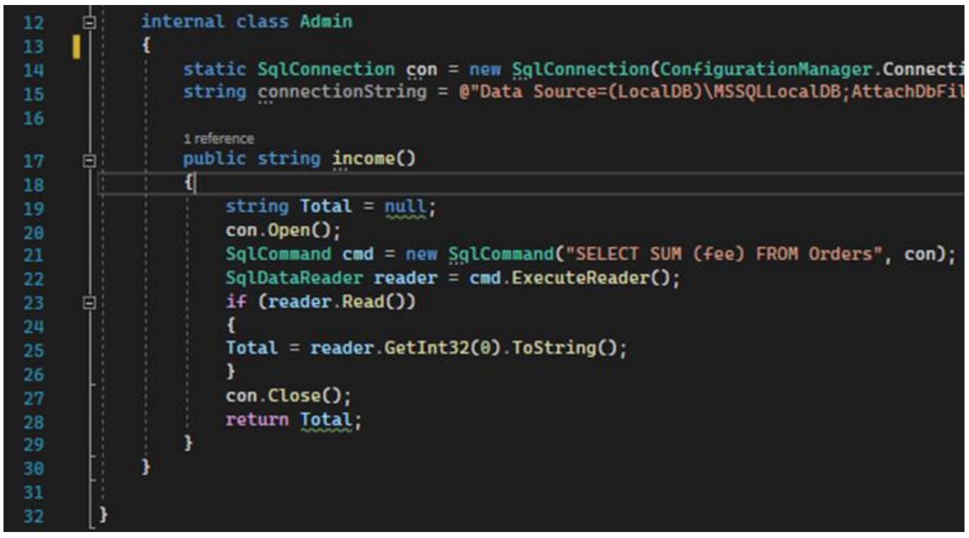
```
11        internal class Receptionist
```



```
34          public string RegisterReceptionist()
35          {
36              string status;
37              con.Open();
38              SqlCommand cmd2 = new SqlCommand("insert into Users(Username,Password,Role,Email,Fullname,Birthday) values(@username,@Password,'Receptionist',@Email,@Fullname,@Birthday)", con);
39              cmd2.Parameters.AddWithValue("Username", username);
40              cmd2.Parameters.AddWithValue("Password", Password);
41              cmd2.Parameters.AddWithValue("Email", Email);
42              cmd2.Parameters.AddWithValue("Fullname", Fullname);
43              cmd2.Parameters.AddWithValue("Birthday", Birthday);
44
45              cmd2.ExecuteNonQuery();
46              if(username == "" || Fullname == "" || Birthday == "" || Email == "" || Password == "")
47                  status = "Please fill mandatory field";
48              else
49                  status = "Register Successful";
50              con.Close();
51              return status;
52          }
53      }
54  }
```

| Code Line | Explanation |
|---|---|
| 36 | Create string to check the status of the form |
| 37 | Open the connection to the database |
| 38 | Create a database command to insert values to database |
| 39-43 | Assigning values to a SQL parameter with selected data |
| 45 | Execute the “cmd2” to insert the selected database |
| 46-59 | Execute the “cmd” to insert the selected database and gives value to “status” |
| 50 | Close the connection to the database |
| 51 | Return a value called “status” to where this method called |

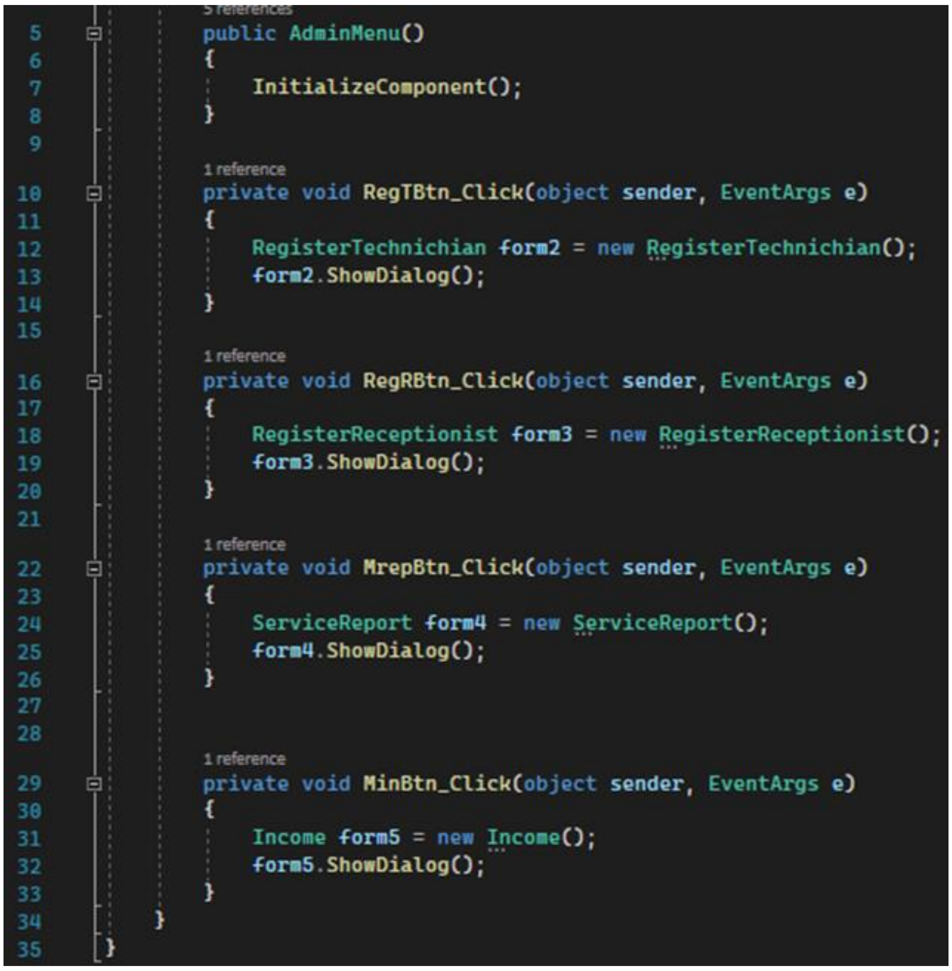
**Admin.cs (Nathaniel)**



```
    internal class Admin
    {
        static SqlConnection con = new SqlConnection(ConfigurationManager.Connecti
        string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFil

        1 reference
        public string income()
        {
            string Total = null;
            con.Open();
            SqlCommand cmd = new SqlCommand("SELECT SUM (fee) FROM Orders", con);
            SqlDataReader reader = cmd.ExecuteReader();
            if (reader.Read())
            {
            Total = reader.GetInt32(0).ToString();
            }
            con.Close();
            return Total;
        }
    }

}
```

| Code Line | Explanation |
|---|---|
| 19 | Create a string total |
| 20 | Open the connection to the database |
| 21 | Create a database command to select sum of database |
| 22 | Read data from database and execute cmd set of rows from database |
| 23-25 | Get a value of column as 32 bit sign string of total |
| 27 | Close the connection to the database |
| 28 | Return a value called “Total” to where this method called |

Admin Menu (Nathaniel)

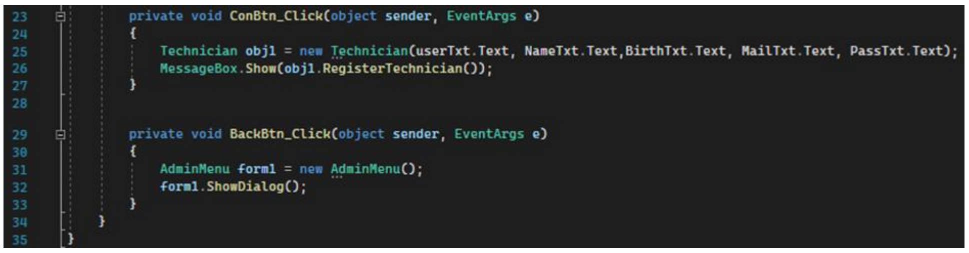


```
          5 references
 5        public AdminMenu()
 6        {
 7            InitializeComponent();
 8        }
 9
          1 reference
10        private void RegTBtn_Click(object sender, EventArgs e)
11        {
12            RegisterTechnichian form2 = new RegisterTechnichian();
13            form2.ShowDialog();
14        }
15
          1 reference
16        private void RegRBtn_Click(object sender, EventArgs e)
17        {
18            RegisterReceptionist form3 = new RegisterReceptionist();
19            form3.ShowDialog();
20        }
21
          1 reference
22        private void MrepBtn_Click(object sender, EventArgs e)
23        {
24            ServiceReport form4 = new ServiceReport();
25            form4.ShowDialog();
26        }
27
28
          1 reference
29        private void MinBtn_Click(object sender, EventArgs e)
30        {
31            Income form5 = new Income();
32            form5.ShowDialog();
33        }
34    }
35 }
```

| Code Line | Explanation |
|---|---|
| 12 | Open the RegisterTechnician form |

| | |
|---|---|
| 13 | Show a RegisterTechnician form |
| 18 | Open the RegisterReceptionist form |
| 19 | Show a RegisterReceptionsit form |
| 24 | Open the ServiceReport form |
| 25 | Show a ServiceReport form |
| 31 | Open the Income form |
| 32 | Show an income form |

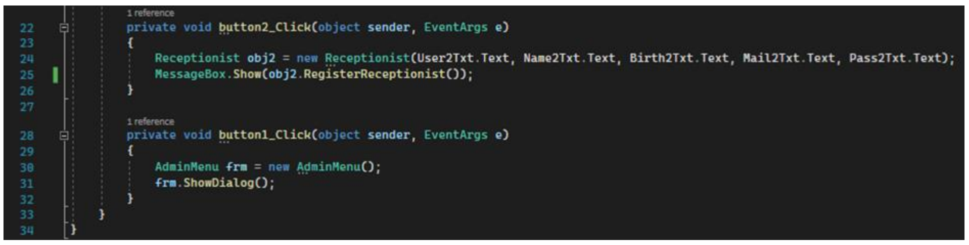
Registration Technician Form (RegisterTechnician.cs) (Nathaniel)



```
23          private void ConBtn_Click(object sender, EventArgs e)
24          {
25              Technician obj1 = new Technician(userTxt.Text, NameTxt.Text,BirthTxt.Text, MailTxt.Text, PassTxt.Text);
26              MessageBox.Show(obj1.RegisterTechnician());
27          }
28
29          private void BackBtn_Click(object sender, EventArgs e)
30          {
31              AdminMenu form1 = new AdminMenu();
32              form1.ShowDialog();
33          }
34      }
35  }
```

| Code Line | Explanation |
|---|---|
| 25 | Insert new data to Technician/user database |
| 26 | A message box to show status |
| 31 | Open an AdminMenu form |
| 32 | Show a AdminMenu form |

Registration Receptionist Form (RegisterReceptionist.cs) (Nathaniel)

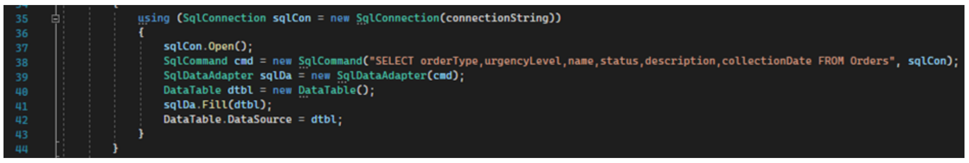


```
            1 reference
22          private void button2_Click(object sender, EventArgs e)
23          {
24              Receptionist obj2 = new Receptionist(User2Txt.Text, Name2Txt.Text, Birth2Txt.Text, Mail2Txt.Text, Pass2Txt.Text);
25              MessageBox.Show(obj2.RegisterReceptionist());
26          }
27
            1 reference
28          private void button1_Click(object sender, EventArgs e)
29          {
30              AdminMenu frm = new AdminMenu();
31              frm.ShowDialog();
32          }
33      }
34  }
```

| Code Line | Explanation |
|---|---|
| 24 | Insert new data to Receptionist/User database |
| 25 | A message box to show status |

| | |
|---|---|
| 30 | Open an AdminMenu form |
| 31 | Show a AdminMenu form |

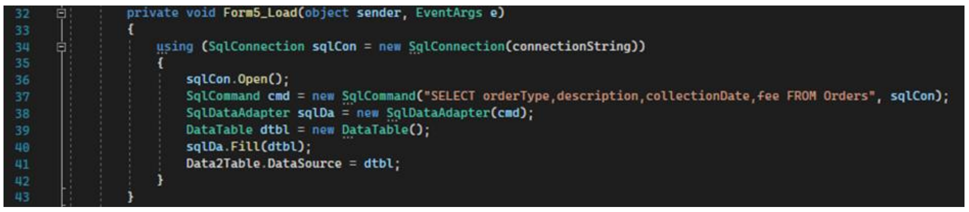
Service Report Form (ServiceReport.cs) (Nathaniel)



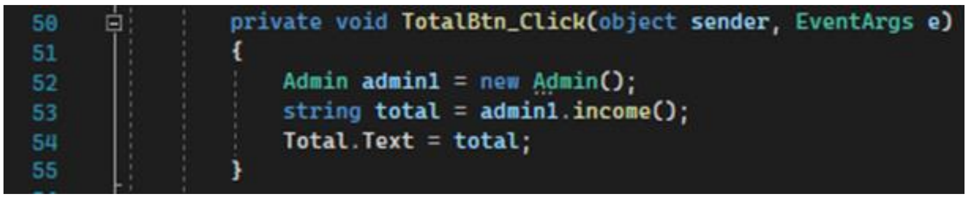
```
35          using (SqlConnection sqlCon = new SqlConnection(connectionString))
36          {
37              sqlCon.Open();
38              SqlCommand cmd = new SqlCommand("SELECT orderType,urgencyLevel,name,status,description,collectionDate FROM Orders", sqlCon);
39              SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);
40              DataTable dtbl = new DataTable();
41              sqlDa.Fill(dtbl);
42              DataTable.DataSource = dtbl;
43          }
44      }
```

| Code Line | Explanation |
|---|---|
| 37 | Open the connection to the database |
| 38 | Create sql command to select required data from orders database |
| 39 | Represent sql commands and database connection |
| 40 | Represent database table from order |
| 41 | To fill the row of data table |
| 42 | Result of data table |

Income Form (Income.cs) (Nathaniel)



```
32      private void Form5_Load(object sender, EventArgs e)
33      {
34          using (SqlConnection sqlCon = new SqlConnection(connectionString))
35          {
36              sqlCon.Open();
37              SqlCommand cmd = new SqlCommand("SELECT orderType,description,collectionDate,fee FROM Orders", sqlCon);
38              SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);
39              DataTable dtbl = new DataTable();
40              sqlDa.Fill(dtbl);
41              Data2Table.DataSource = dtbl;
42          }
43      }
```



```
50      private void TotalBtn_Click(object sender, EventArgs e)
51      {
52          Admin admin1 = new Admin();
53          string total = admin1.income();
54          Total.Text = total;
55      }
```

| Code Line | Explanation |
|---|---|
| 36 | Open the connection to the database |
| 37 | Create sql command to select required data from orders database |
| 38 | Represent sql commands and database connection |
| 39 | Represent database table from order |

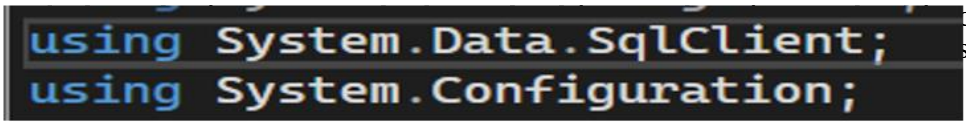
| | |
|---|---|
| 40 | To fill the row of data table |
| 41 | Result of data table |
| 52 | open admin1 from admin class |
| 53 | Create string total income |
| 54 | Show result of total income |

Customers.cs (Owen)

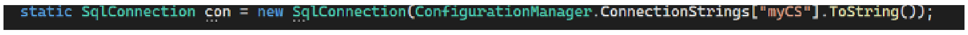
The Customers.cs is a class file inside of our program that contains every single constructor and

tion. Methods from

system connects the

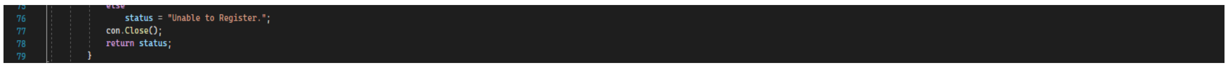


```
using System.Data.SqlClient;
using System.Configuration;
```



```
static SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["myCS"].ToString());
```

1. Register customer (Receptionist Menu) (Mikael Owen Kartika)

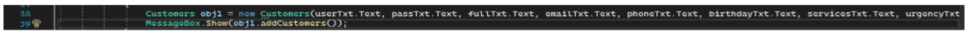


```
75              else
76                  status = "Unable to Register.";
77              con.Close();
78              return status;
79          }
```

-To register new customers and choose requested services. (Owen)

| Code Line | Code Explanation |
|---|---|
| 46 | Create string to check the status of the form |
| 47 | Open the connection to the database |
| 50-53 | Create a database command to insert values to 3 different tables |
| 54-69 | Assigning values to a SQL parameter with selected data |
| 70-71 | Execute the “cmd2” and “cmd3” to insert the selected database |
| 72-76 | Execute the “cmd” to insert the selected database and gives value to “status” |
| 77 | Close the connection to the database |
| 78 | Return a value called “status” to where this method called |

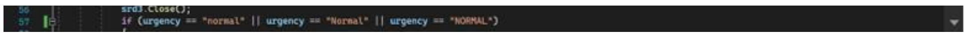
This method is called in the RegisterCustomer Form



```
38              Customers obj1 = new Customers(userTxt.Text, passTxt.Text, fullTxt.Text, emailTxt.Text, phoneTxt.Text, birthdayTxt.Text, servicesTxt.Text, urgencyTxt
39              MessageBox.Show(obj1.addCustomers());
```

| Code Line | Code Explanation |
|---|---|
| 32 | Create an if statement which true when there are empty textbox and do the following task |
| 33 | Show message box of “please fill all required fields” when if is true |
| 38 | Create an object by assigning a constructor “Customers” into created object “obj1” |
| 39 | Call the method of “addCustomers” and show the returned value which is “status” |

2. Generate Receipt for customers (Receptionist Menu) (Mikael Owen Kartika)



```
56          srd3.Close();
57          if (urgency == "normal" || urgency == "Normal" || urgency == "NORMAL")
```

| Code Line | Code Explanation |
|---|---|
| 25-28 | Create required variable or in this case (servID,urgency,servName,fee) |
| 30 | Establish the connection with the database |
| 31 | Create sql command to select required data from customers table where the username are inserted by users. |
| 32 | Create a SQL Data Reader with variable “srd” and assign the value from executing reader from the ‘command’ |
| 33 | Assign a while loop to read the database line by line with the condition (rd.read()) and create data collection from the rows |
| 35 | Assign the selected value with index 0 (fullname) to fullname textbox |
| 36 | Assign the selected value with index 1(services) to servID |
| 37 | Assign the selected value with index 2 (description) to urgency |
| 41 | Close the reader of the database |
| 42 | Create an object ‘command3’ and assign the SQL Command to select servicename from table service where service ID equals to the value of servID |
| 43 | Create a SQL Data Reader with variable “srd3” and assign the value from executing reader from the ‘command’ |
| 44 | Create a while loop to read the database line with the condition (srd3.Read()) and store value from the database into variable |
| 46 | Get value index 0 from reader srd3 inserted to the textbox servicesTxt |
| 47 | Get value index 0 from reader srd3 stored in variable servName |

| | |
|---|---|
| 50 | Close the reader of the database |
| 51 | Create sql command named command5 to insert data in orders table where values are before stored in variable servName |
| 52 | Assigning values to a SQL parameter with selected data |
| 53 | Execute the “command5” to INSERT the selected data |
| 56 | Close the reader of the database |

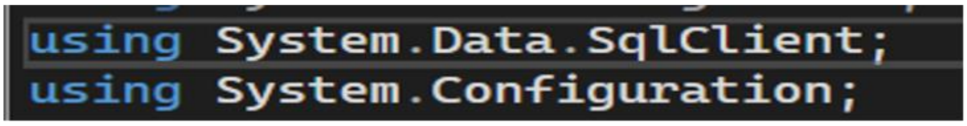
```
}
```

| Code Line | Code Explanation |
|---|---|
| 57 | Create an IF Statement to check the condition whether the urgency normal or urgent and then determine the price. Here, the condition is true if it’s normal |
| 59 | Create an object “command2” and assign an SQL Command SELECT values from the column “CostNormal” WHERE it meets the condition. |
| 60 | Create a SQL Data Reader with variable “srd2” and select the value from executing reader from the ‘command2’ |
| 61 | Create a while loop to read the database line with the condition (srd2.Read()) and store value from the database into variable |
| 63 | Store the value from index 0 of database read on srd2 to textbox feeTxt |
| 64 | Assign the value from index 0 of database read 2 on variable fee |
| 68 | Close the reader of the database |
| 69 | Create an object “command6” and assign an SQL Command INSERT values into table orders with the values stored in variable “fee” |
| 70 | Assigning values to a SQL Parameter with selected data |
| 71 | Execute the “command6” to insert the selected data |
| 73 | Create an ELSE IF Statement with true condition if the urgency is “urgent” |
| 75 | Create an object “command4” and assign an SQL Command SELECT values from the column “CostUrgent” WHERE it meets the condition. |
| 76 | Create a SQL Data Reader with variable “srd2” and select the value from executing reader from the ‘command4’. |
| 77 | Create a while loop to read the database line with the condition (srd4.Read()) and store value from the database into variable. |
| 79 | Assign the value of srd4 from index 0 to textbox feeTxt. |
| 82 | Close the reader of the database. |

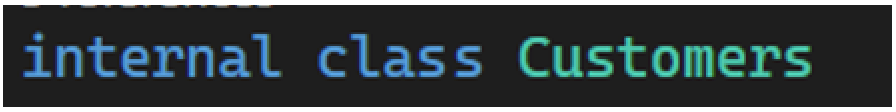
| | |
|---|---|
| 83 | Create an object "command7" and assign an SQL Command INSERT values to table orders the values of variable "fee". |
| 84 | Assigning values to a SQL Parameter with selected data. |
| 85 | Execute the "command6" to insert the selected data. |
| 87 | Close the connection to the database. |

**Customers.cs (Stanley)**

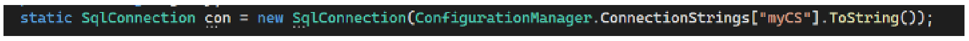
Customers.cs is a class method in the program that contains all the field and function in your program that have spatial functions. Contains methods from the form of our program.



```
using System.Data.SqlClient;
using System.Configuration;
```

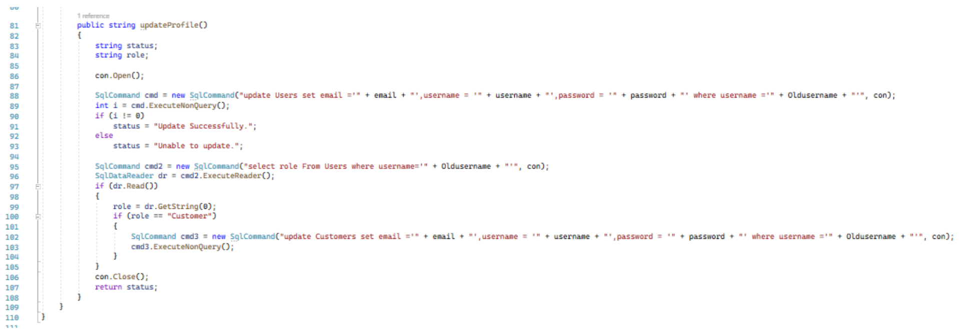


```
internal class Customers
```



```
static SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["myCS"].ToString());
```

1. Edit Profile customer (Customer Menu) (Stanley Lais)

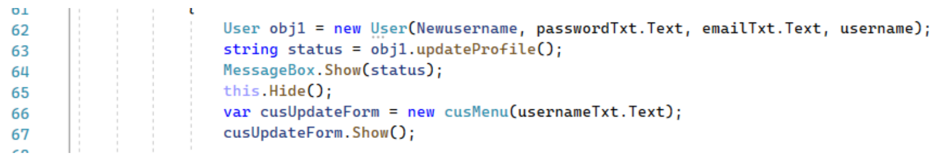


```
            1 reference
81          public string updateProfile()
82          {
83              string status;
84              string role;
85
86              con.Open();
87
88              SqlCommand cmd = new SqlCommand("update Users set email ='" + email + "',username = '" + username + "',password = '" + password + "' where username ='" + Oldusername + "'", con);
89              int i = cmd.ExecuteNonQuery();
90              if (i != 0)
91                  status = "Update Successfully.";
92              else
93                  status = "Unable to update.";
94
95              SqlCommand cmd2 = new SqlCommand("select role From Users where username='" + Oldusername + "'", con);
96              SqlDataReader dr = cmd2.ExecuteReader();
97              if (dr.Read())
98              {
99                  role = dr.GetString(0);
100                 if (role == "Customer")
101                 {
102                     SqlCommand cmd3 = new SqlCommand("update Customers set email ='" + email + "',username = '" + username + "',password = '" + password + "' where username ='" + Oldusername + "'", con);
103                     cmd3.ExecuteNonQuery();
104                 }
105             }
106             con.Close();
107             return status;
108         }
109     }
110 }
```

To edit customer profile (stanley)

| Code Line | Code Explanation |
|---|---|
| 83 - 84 | Create string to check the status and role of the form |
| 86 | Open the connection to the database |
| 88 - 93 | Create a database command to update values to users tables |
| 95 - 101 | Create a database command to Select role values from users tables |
| 102 - 103 | Create a database command to update values to customers tables |
| 106 | Close the connection to the database |
| 107 | Return a value called "status" to where this method called |

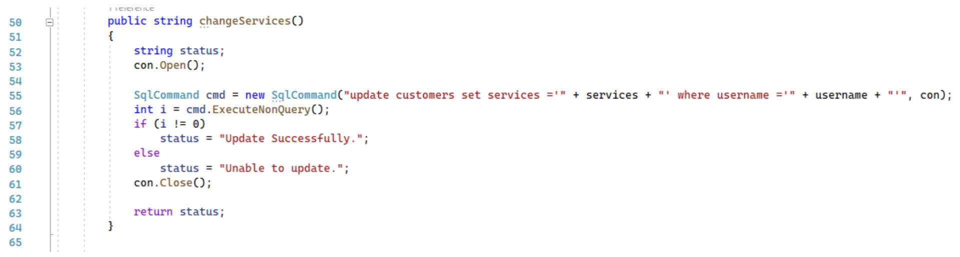
This method is called in the cusProfile Form



```
62              User obj1 = new User(Newusername, passwordTxt.Text, emailTxt.Text, username);
63              string status = obj1.updateProfile();
64              MessageBox.Show(status);
65              this.Hide();
66              var cusUpdateForm = new cusMenu(usernameTxt.Text);
67              cusUpdateForm.Show();
```

| Code Line | Code Explanation |
|---|---|
| 62 | Create an object by assigning a constructor “user” into created object “obj1” |
| 63 | Create a string variable by assigning an object “obj1” into created variable “status” |
| 64 | Show the returned value “status” |
| 65 | Hide this form |
| 66 | Create a variable by assigning a new form “cusMenu” into created variable “cusUpdateForm” |
| 67 | Execute variable “cusUpdateForm” |

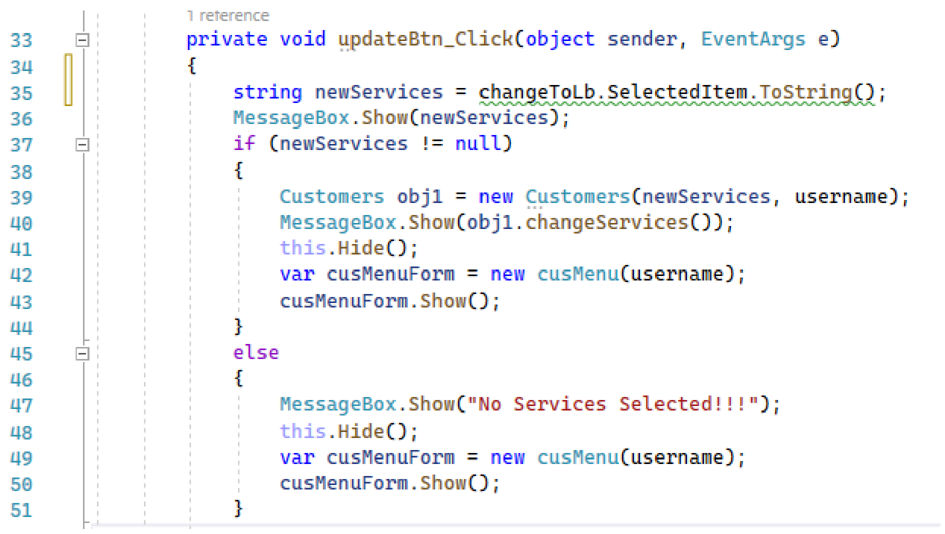
To Chage customer services (stanley)



```
50      public string changeServices()
51      {
52          string status;
53          con.Open();
54
55          SqlCommand cmd = new SqlCommand("update customers set services ='" + services + "' where username ='" + username + "'", con);
56          int i = cmd.ExecuteNonQuery();
57          if (i != 0)
58              status = "Update Successfully.";
59          else
60              status = "Unable to update.";
61          con.Close();
62
63          return status;
64      }
65
```

| Code Line | Code Explanation |
|---|---|
| 52 | Create string to check the status of the form |
| 53 | Open the connection to the database |
| 55 | Create a database command to update values to Customers tables |
| 56 | Execute the database command |
| 57 - 60 | Create an IF Statement to check the condition whether the int I not equal to zero or equal to zero and then determine the status. |
| 61 | Close the connection to the database |
| 63 | Return a value called “status” to where this method called |

This method is called in the cusChangeServices Form



```
            1 reference
33          private void updateBtn_Click(object sender, EventArgs e)
34          {
35              string newServices = changeToLb.SelectedItem.ToString();
36              MessageBox.Show(newServices);
37              if (newServices != null)
38              {
39                  Customers obj1 = new Customers(newServices, username);
40                  MessageBox.Show(obj1.changeServices());
41                  this.Hide();
42                  var cusMenuForm = new cusMenu(username);
43                  cusMenuForm.Show();
44              }
45              else
46              {
47                  MessageBox.Show("No Services Selected!!!");
48                  this.Hide();
49                  var cusMenuForm = new cusMenu(username);
50                  cusMenuForm.Show();
51              }
```

| Code Line | Code Explanation |
|---|---|
| 35 | Select item From the listboxt and assigned it into string variable "newServices" |
| 36 | Show messagebox "newServices" for confirmation |
| 64 | Create an IF Statement to check the condition whether the newServices is null or not and then call the method. |

## Test Plan

| Test Case | Function Name | Test Objective | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|---|
| 1 | Register Technician | -To input data of technician is clear | -Display not successful message box if there are blank input | -Display successful to register for technician if there are no blank input | Need to be improve the code to further development in form |
| 2 | Register Technician | -To input data of receptionist | -Display not successful message box if there are blank input | -Display successful to register for receptionist if there are not blank input | Need to be improve the code to further development in form |

| | | | | | |
|---|---|---|---|---|---|
| 3 | Service Report | -To show report | -Show the require data of order history | -Display necessary things for service report | none |
| 4 | Income | -To show description and total income | -Show a sum monthly total fee of order | -Show all sum total fee of order | Research more to improve the code and form |
| 5 | Register Customers | -To input data from customers | -Display “Please fill mandatory fields” it means all field have input<br><br>-Successful when all fields have input and recorded in the customers table on database | -Display successful to register customer if there are no blank input | Less usage of listbox |
| 6 | Generate Receipt | -To generate receipt contains name, service, fee, | -Customers can have their receipt containing their name,service and fee.<br><br>-Successful when all fields show up and recorded in the customers table on database | -Successfully generate receipt that contains name,service and fee | None |
| 7 | Update customers profile | -To update information on the database to the new one | -Update data on the database to the new one except role, | -Successfully update data on the users database to the new oe | None |
| 8 | View pending orders | -To view orders that are still ongoing / not yet completed | -View pending orders based on status | -Successfully view pending orders based on status between ongoing/completed | None |
| 9 | Update service orders | -To change service orders status | -Change service orders description | -Successfully change service orders status from | |

| | | | | | |
|---|---|---|---|---|---|
| | | | according to the condition of the laptop and show collection date | ongoing to completed and show collection date | |
| 10 | Update technician profile | -To update information on the database to the new one | -Update data on the database to the new one except role, | -Successfully update data on the users database to the new one | None |
| 11 | Update receptionist profile | -To update information on the database to the new one | -Update data on the database to the new one except role, | -Successfully update data on the users database to the new one | None |
| 12 | View service | -To view service type ordered by customers | -View service type ordered when customers register | -Successfully view service type | None |
| 13 | Receptionist Main Menu | -To create the main menu for admins to choose the menu | -The receptionist will go the desired menu depends on the selected menu. | -The receptionist can choose which menu want to be selected and directed to the selected menu. | None |
| 14 | Login | To test whether<br><br>The users able to login using the SQL Database and the system able to automaticall y detect which role | -The student able to login the correct credentials and proceed to the student menu<br><br>-The users able to login the correct credentials and proceed to the next menu<br><br>-If the users, username or password is incorrect, a message box will show incorrect credentials | -Users  able to login successfully and continue to each role menu<br><br>-Error Message Box will show if the username or password is incorrect | None |

| | | | | | |
|---|---|---|---|---|---|
| | | | -The users will automatically login into their user's role which is receptionist, customers, technician, or admin | | |
| 15 | Yes or no confirmation message box | -To make the users give confirmation before updating an order | -the user will receive and asked to choose from the message box when they click the update button | -the message box successfully popped out and users can only update if they picked yes | |

## Conclusion

From creating the laptop service system we learnt that one are connected to another in the database and to be able to works well we have to discuss in team with each member shares how their program will works and connect it to each other.  Those complex things are getting simpler by doing it using class , methods, object. In brief, Collaboration and coding are two skills a successful programmer seamlessly combine  together to design high quality programs. The team has done a quite good job to make the program to make an electronic shop system using a GUI based application. However, there are still a lot of shortfalls, with lack of exception handlings, naming conventions, etc. Hopefully the team will be able to improve in the future and fix the problems and learn from the flaws of this system.

## References

. (2015, January). *C# Properties (Get and Set)*. W3schools. https://www.w3schools.com/cs/cs_properties.php

*How can I get column names from a table in SQL Server?* (2009, June 28). Stack Overflow. https://stackoverflow.com/questions/1054984/how-can-i-get-column-names-from-a-table-in-sql-server#:%7E:text=USE%20db_name%3B%20DESCRIBE%20table_name%3B,column%20names%20with%20the%20type.

Love, C. (2021, February 17). *Collaboration and Coding is Fun and Fosters Team Work*. TechnoKids Blog. https://www.technokids.com/blog/teaching-strategies/collaboration-and-coding-is-fun-and-fosters-team-work/

*System.StackOverflowException was unhandled*. (2013, March 5). Stack Overflow. https://stackoverflow.com/questions/15221268/system-stackoverflowexception-was-unhandled

*What is Use Case Diagram?* (2016, June). Visual-Paradigm. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

## Workload Matrix

| | | | |
|---|---|---|---|
| 1. | -Creating Storyboard<br>-Creating Form Design<br>-Code Update Own Profiile (Receptionist)<br>-Code Generate Receipt (Receptionist)<br>-Code Register Customers (Receptionist)<br>-Fix Bugs<br>-Creating the Document<br>-Writing the Test Plan | Mikael Owen Kartika | Full Completion |
| 2. | -Creating Storyboard<br>-Creating Form Design<br>-Creating Database Frontend and Backend<br>-Code Register new Technician (Admin)<br>-Code Register new Receptionist(Admin)<br>-Code the view service report and total income (Admin)<br>-Writing the Test Plan<br>-Fix Bugs | Nathaniel Sudiono | Full Completion |
| 3. | -Creating Storyboard<br>-Creating Form Design<br>-Code Update Own Profile (Technician)<br>-Code view service request (Technician<br>-Code add description and add laptop collection date (Technician)<br>-Cleaning and Simplify Backend Code | Ariel Amerigo Joe Banua | Full Completion |

| | | | |
|---|---|---|---|
| | -Create the creative design of the forms<br><br>-Fix Bugs | | |
| 4 | -Creating Storyboard<br><br>-Creating Form Design<br><br>-Code Update Own Profile (Customers)<br><br>-Code view description and add laptop collection date (Customers)<br><br>-Code change requested service (Customers)<br><br>-Cleaning and Simplify Backend Code<br><br>-Create the creative design of the forms<br><br>-Fix Bugs | Stanley Lais | Full Completion |