Table of Contents

Introduction

Within the sphere of marketing research, trustworthy sources of information are originated form the customers in order to maximize the profit of business. The strategic and smart method of obtaining the information is conducting surveys from customers. We develop numerous survey systems as regards the consumers and have made many notable projects in local market as well as global one. The **Apple** company approached me to construct the program which focus on undertaking surveys of its branded Apple smartwatches and evaluating the responses. The program will allow two characters (Administrators & Customers). Therefore, program will be designed for each role. Users are allowed to create secure accounts, login and perform their respective tasks. For the administrators, efficient pages will be built for creating surveys and analyzing the responses. As regards the customers, attention-grabbing pages will be definitely developed for answering the surveys and reviewing them.

1. Task – 1

1.1 Analysis

In task1, a program which is applicable to two roles (administrators, customers) is created for conducting, responding and reporting surveys according to the roles. When it comes to program, design thinking is vital to capture customers' attention and to arouse the productivity of administrators. Company product has been highlighted for top priority aiming for obtaining an accurate survey report. For the sake of good algorithm, the program steers towards handling a range of input data effectively and taking less amount of memory.

In task2, test plan has to be written with date and identifier in order to keep track of whether a program works systematically. Detailed test scripts with practical test results are accompanied by the test plan.

In task3, class diagram is constructed and each class has been explained why it has to be created. At last, the task 3 ends up with user manual for the purpose of proper use without any risk.

1.2 Assumption

An integrated development environment (IDE) is required for coding so Microsoft Visual Studio has to be installed. Furthermore, Microsoft SQL server is installed to create databases and tables for storing input data. C# language is used throughout the program. Images concerning the company product are applied for background images. User Interfaces of program are taken screenshot for black box testing. As regards white box testing, some pieces of coding have to be taken screenshots. Whenever data have been inputted into database, successful results have to be recorded and incorporated into test results.

1.3 Coding

1. WelcomePage.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class WelcomePage : Form
    {
        public WelcomePage()
        {
            InitializeComponent();
        }
        private void picboxadmin_Click(object sender, EventArgs e)
        {
            AdminLogin al = new AdminLogin();
            al.Show();
            this.Hide();
        }
        private void picboxcustomer_Click(object sender, EventArgs e)
        {
            CustomerLogin cl = new CustomerLogin();
            cl.Show();
            this.Hide();
        }
    }
}
```

2. AdminLogin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class AdminLogin : Form
    {
        AdminInformation ai = new AdminInformation();
        AdminDatasetTableAdapters.AdminTableTableAdapter adminDataset = new
AdminDatasetTableAdapters.AdminTableTableAdapter();
        DataTable adminDataTable = new DataTable();

        public AdminLogin()
        {
            InitializeComponent();
        }

        private void btnlogin_Click(object sender, EventArgs e)
        {
            //receive information
            ai.adminUsername = txtusername.Text;
            ai.adminPassword = txtpassword.Text;

            //login process
            if (txtusername.Text != string.Empty || txtpassword.Text != string.Empty)
            {
                //save process
                adminDataTable = adminDataset.LoginAdmin(ai.adminUsername,
ai.adminPassword);
```

```
                if (adminDataTable.Rows.Count == 1)
                {
                    //analysis process
                    ai.adminID = adminDataTable.Rows[0][0].ToString();
                    ai.adminName = adminDataTable.Rows[0][1].ToString();

                    //output result process
                    MessageBox.Show(" Warmly welcome " + ai.adminName + " ID " +
ai.adminID, "Login successful", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    AdminDashboard admDash = new AdminDashboard();
                    admDash.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Please make sure your adminUsername and password \n
\t Thank you.", "Login fail",MessageBoxButtons.OK,MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("Please completely fill your adminUsername and password
\n \t Thank you.", "Login Incomplete", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void linklableRegister_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            //link to register page
            AdminRegistration admRegtr = new AdminRegistration();
            admRegtr.Show();
            this.Hide();
        }

        private void btncancel_Click(object sender, EventArgs e)
        {
            //close process
            string messege = "Are you sure you want to close the program?";
```

```
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

3. AdminRegistration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class AdminRegistration : Form
    {
        AdminDatasetTableAdapters.AdminTableTableAdapter adminDataset = new
AdminDatasetTableAdapters.AdminTableTableAdapter();
        DataTable adminDataTable = new DataTable();
        AdminInformation ai = new AdminInformation();

        public AdminRegistration()
        {
```

```
            InitializeComponent();
        }

        private void txtGetData()
        {
            //receive process
            ai.adminID = txtid.Text;
            ai.adminName = txtadname.Text;
            ai.adminUsername = txtusername.Text;
            ai.adminPassword = txtpassword.Text;
            ai.adminEmail = txtemail.Text;
        }
        private void txtDataEmpty()
        {
            //empty textbox funtion
            txtadname.Text = string.Empty;
            txtusername.Text = string.Empty;
            txtpassword.Text = string.Empty;
            txtemail.Text = string.Empty;
        }
        private void Auto_ID()
        {
            //admin auto id
            adminDataTable = adminDataset.GetData();
            if ( adminDataTable.Rows.Count==0 )
            {
                txtid.Text = "Admin-001";
            }
            else
            {
                int num = adminDataTable.Rows.Count - 1;
                string oldId = adminDataTable.Rows[num][0].ToString();
                int newId = Convert.ToInt32(oldId.Substring(6,3));

                if( newId>=1 && newId<9 )
                {
                    txtid.Text = "Admin-00"+(newId+1);
                }
                else if( newId >=9&& newId <99 )
```

```
                {
                    txtid.Text = "Admin-0"+(newId+1);
                }
                else if( newId >=99 && newId <= 999 )
                {
                    txtid.Text = "Admin-"+(newId+1);
                }
            }
        }
        private void AdminRegistration_Load(object sender, EventArgs e)
        {
            Auto_ID();
        }
        private void linkLabelLogin_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            //link to login page
            AdminLogin admLgn = new AdminLogin();
            admLgn.Show();
            this.Hide();
        }
        private void btnregister_Click_1(object sender, EventArgs e)
        {
            txtGetData();
            if (txtadname.Text != "" && txtusername.Text != ""&& txtpassword.Text != ""
&& txtemail.Text != "")//testing process
            {
                adminDataset.SaveAdmin(ai.adminID, ai.adminName, ai.adminUsername,
ai.adminPassword, ai.adminEmail);
                //successful message
                MessageBox.Show("The registration has been successful. \n \t Thank you.",
"Registration successful! ", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                Auto_ID();
                txtDataEmpty();
                AdminLogin al = new AdminLogin();
                al.Show();
                this.Close();
            }
            else
```

```
            {
                //error messege
                MessageBox.Show("Please completely fill the form! \n \t Thank you.",
"Registration fail!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                txtDataEmpty();
            }
        }
        private void btncancel_Click(object sender, EventArgs e)
        {
            //close funtion
            string messege = "Are you sure you want to cancel?";
            string title = "Cancelling";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons);
            if (result == DialogResult.OK)
            {
                Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

4. AdminDashboard.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
```

```
{
    public partial class AdminDashboard : Form
    {
        AdminDatasetTableAdapters.AdminTableTableAdapter adminDataset = new
AdminDatasetTableAdapters.AdminTableTableAdapter();
        SurveyDatasetTableAdapters.SurveyTableTableAdapter surveyDataset = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        DataTable adminDataTable = new DataTable();
        DataTable surveyDataTable = new DataTable();
        AdminInformation ai = new AdminInformation();

        public AdminDashboard()
        {
            InitializeComponent();
        }

        private void AdminDashboard_Load(object sender, EventArgs e)
        {
            //Show id,name,email,number of created survey of respective admin
            adminDataTable = adminDataset.GetAdminData(ai.adminID);
            lbladminid.Text = adminDataTable.Rows[0][0].ToString();

            adminDataTable = adminDataset.GetAdminData(ai.adminID);
            lbladminname.Text = adminDataTable.Rows[0][1].ToString();

            adminDataTable = adminDataset.GetAdminData(ai.adminID);
            lbladminemail.Text = adminDataTable.Rows[0][4].ToString();

            surveyDataTable = surveyDataset.GetSurveyData(ai.adminID);
            if (surveyDataTable.Rows.Count >= 1)
            {
                lbladminsurveyid.Text = Convert.ToString(surveyDataTable.Rows.Count);
            }
            else
            {
                lbladminsurveyid.Text = "There is no survey you have created.";
            }
        }
```

```
        private void viewToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //adminview page in menu bar
            AdminViewList alist = new AdminViewList();
            alist.Show();
        }

        private void registerToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //register page in menu bar
            AdminRegistration ar = new AdminRegistration();
            ar.Show();
            this.Close();
        }

        private void loginToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //login page in menu bar
            AdminLogin al = new AdminLogin();
            al.Show();
            this.Close();
        }

        private void closeToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            //close program in menu bar
            string messege = "Do you want to close?";
            string title = "Exit";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Question);
            if (result == DialogResult.OK)
            {
                Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
```

```
        }

        private void viewSurveyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //view created survey page in menu bar
            AdminViewSurvey avs = new AdminViewSurvey();
            avs.Show();
        }

        private void createSurveyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //create survey page in menu bar
            AdminCreateSurvey acs = new AdminCreateSurvey();
            acs.Show();
        }

        private void refreshToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //refresh funtion in menu bar
            Close();
            AdminDashboard adm = new AdminDashboard();
            adm.Show();
            this.Close();
        }
    }
}
```

5. AdminViewList.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class AdminViewList : Form
    {
        public AdminViewList()
        {
            InitializeComponent();
        }

        private void AdminList_Load(object sender, EventArgs e)
        {
            //Show all admins except adminid and password columns
            AdminDatasetTableAdapters.AdminTableTableAdapter adminDataset = new
AdminDatasetTableAdapters.AdminTableTableAdapter();
            DataTable AdminDataTable = new DataTable();
            for (int i = 0; i < 5; i++)
            {
                dgvadminlist.DataSource = adminDataset.GetData();
                dgvadminlist.Columns[i].Visible = true;
                if (i == 0 || i == 3 )
                {
                    dgvadminlist.Columns[i].Visible = false;
                }
                dgvadminlist.Refresh();
            }
        }

        private void btncancel_Click(object sender, EventArgs e)
        {
            //close function process
            string messege = "Do you want to close this window?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result= MessageBox.Show(messege, title,
buttons,MessageBoxIcon.Question);
            if (result == DialogResult.OK)
               {
```

```
                this.Close();
               }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

6. AdminCreateSurvey.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class AdminCreateSurvey : Form
    {
        AdminDatasetTableAdapters.AdminTableTableAdapter adminDataset = new
AdminDatasetTableAdapters.AdminTableTableAdapter();
        SurveyDatasetTableAdapters.SurveyTableTableAdapter surveyDataset = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        AdminInformation ai = new AdminInformation();
        SurveyInformation si = new SurveyInformation();
        DataTable surveyDataTable = new DataTable();

        public AdminCreateSurvey()
        {
            InitializeComponent();
        }
        //Get data from textbox
```

```
        private void txtGetData()
        {
            si.surveyID = txtsurveyid.Text;
            si.surveyName = txtsurveyname.Text;
            si.surveyDate = txtdate.Text;
            si.adminName = ai.adminName;
            si.adminID = ai.adminID;
            si.question1 = txtq1.Text;
            si.question2 = txtq2.Text;
            si.question3 = txtq3.Text;
            si.question4 = txtq4.Text;
            si.question5 = txtq5.Text;
            si.question6 = txtq6.Text;
        }
        //textbox empty
        private void txtDataEmpty()
        {
            txtsurveyname.Text = string.Empty;
            txtq1.Text = string.Empty;
            txtq2.Text = string.Empty;
            txtq3.Text = string.Empty;
            txtq4.Text = string.Empty;
            txtq5.Text = string.Empty;
            txtq6.Text = string.Empty;
        }

        private void AdminCreateSurvey_Load(object sender, EventArgs e)
        {
            //auto show adminname,id, date
            Auto_Id();
            surveyDataTable = adminDataset.GetAdminData(ai.adminID);
            txtadminname.Text = surveyDataTable.Rows[0][3].ToString();
            txtadminid.Text = surveyDataTable.Rows[0][4].ToString();
            txtdate.Text = DateTime.Now.ToString("d-M-yy");
        }

        private void Auto_Id()
        {
            //Survey auto ID
```

```
            surveyDataTable = surveyDataset.GetData();
            if (surveyDataTable.Rows.Count == 0)
            {
                txtsurveyid.Text = "Survey-001";
            }
            else
            {
                int num = surveyDataTable.Rows.Count - 1;
                string oldId = surveyDataTable.Rows[num][0].ToString();
                int newId = Convert.ToInt32(oldId.Substring(7, 3));

                if (newId >= 1 && newId < 9)
                {
                    txtsurveyid.Text = "Survey-00" + (newId + 1);
                }
                else if (newId >= 9 && newId < 99)
                {
                    txtsurveyid.Text = "Survey-0" + (newId + 1);
                }
                else if (newId >= 99 && newId <= 999)
                {
                    txtsurveyid.Text = "Survey-" + (newId + 1);
                }
            }
        }

        private void btncancel_Click(object sender, EventArgs e)
        {
            //close funtion
            string messege = "Are you sure you want to close the program?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
```

```
            {
                this.Show();
            }
        }
        //create surveys
        private void btnconfirm_Click(object sender, EventArgs e)
        {
            txtGetData();
            if (txtsurveyname.Text == string.Empty || txtq1.Text == string.Empty ||
txtq2.Text == string.Empty || txtq3.Text == string.Empty || txtq4.Text == string.Empty ||
txtq5.Text == string.Empty ||
            txtq6.Text == string.Empty)
            {
                MessageBox.Show("Please completely fill the form! ", "Survey
Fail",MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
            {
                string messege = "Do you want to save the survey?";
                string title = "Confirmation";
                MessageBoxButtons buttons = MessageBoxButtons.YesNo;
                DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Question);
                if (result == DialogResult.Yes)
                {
                    surveyDataset.SaveSurvey(si.surveyID, si.surveyName, si.surveyDate,
si.adminName, si.adminID, si.question1, si.question2, si.question3, si.question4,
si.question5, si.question6);
                    Auto_Id();
                    txtDataEmpty();
                }
                else if (result == DialogResult.No)
                {
                    this.Show();
                }
            }
        }
    }
}
```

7. AdminViewSurvey.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class AdminViewSurvey : Form
    {
        SurveyDatasetTableAdapters.SurveyTableTableAdapter surveyDataset = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        AdminInformation ai = new AdminInformation();
        SurveyInformation si = new SurveyInformation();
        DataTable surveyDataTable = new DataTable();
        public AdminViewSurvey()
        {
            InitializeComponent();
        }
        private void txtDataEmpty()
        {
            //To get empty textbox
            txtq1.Text = string.Empty;
            txtq2.Text = string.Empty;
            txtq3.Text = string.Empty;
            txtq4.Text = string.Empty;
            txtq5.Text = string.Empty;
            txtq6.Text = string.Empty;
        }
        private void AdminViewSurvey_Load(object sender, EventArgs e)
        {
            //show surveyId, surveyname, 6 questions in table
```

```
            surveyDataTable = surveyDataset.GetSurveyData(ai.adminID);
            dgvSQ.DataSource = surveyDataTable;
            dgvSQ.Columns[2].Visible = false;
            dgvSQ.Columns[3].Visible = false;
            dgvSQ.Columns[4].Visible = false;
            dgvSQ.Refresh();

            //combobox structure
            surveyDataTable = surveyDataset.GetSurveyData(ai.adminID);
            cboquestions.DataSource = surveyDataTable;
            cboquestions.ValueMember = "SurveyID";
            cboquestions.DisplayMember = "SurveyID";
            cboquestions.SelectedValue = 0;
        }

        private void btnupdate_Click(object sender, EventArgs e)
        {
            //Select data from the textbox
            si.question1 = txtq1.Text;
            si.question2 = txtq2.Text;
            si.question3 = txtq3.Text;
            si.question4 = txtq4.Text;
            si.question5 = txtq5.Text;
            si.question6 = txtq6.Text;
            try
            {
                //Update according To selected item of ComboBox
                si.surveyID = cboquestions.SelectedValue.ToString();
                surveyDataset.UpdateSurvey(si.question1, si.question2, si.question3,
si.question4, si.question5, si.question6, si.surveyID);
                dgvSQ.Refresh();
                MessageBox.Show("Update process has been successful","Update
Complete",MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                txtDataEmpty();
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Please Select the survey first",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            }

        }

        private void btnrefresh_Click(object sender, EventArgs e)
        {
            //program refresh
            this.Hide();
            AdminViewSurvey avs = new AdminViewSurvey();
            avs.Show();
        }

        private void btndelete_Click(object sender, EventArgs e)
        {
            try
            {
                //delete process
                si.surveyID = cboquestions.SelectedValue.ToString();
                surveyDataset.DeleteSurveyQuestions(si.surveyID);
                MessageBox.Show("Delete process has been successful", "Delete Complete",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Please Select the survey first",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void dgvSQ_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            //Show question when clicking On the datagridview cell
            surveyDataTable = surveyDataset.GetSurveyData(ai.adminID);
            int rowNo = dgvSQ.CurrentCell.RowIndex;
            txtq1.Text = surveyDataTable.Rows[rowNo][5].ToString();
            txtq2.Text = surveyDataTable.Rows[rowNo][6].ToString();
            txtq3.Text = surveyDataTable.Rows[rowNo][7].ToString();
            txtq4.Text = surveyDataTable.Rows[rowNo][8].ToString();
            txtq5.Text = surveyDataTable.Rows[rowNo][9].ToString();
```

```
            txtq6.Text = surveyDataTable.Rows[rowNo][10].ToString();
            cboquestions.SelectedValue = surveyDataTable.Rows[rowNo][0].ToString();
        }

        private void btnclose_Click(object sender, EventArgs e)
        {
            //close funtion
            string messege = "Do you want to close this window?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Question);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }

        private void btnreport_Click(object sender, EventArgs e)
        {
            try
            {
                //response of selected survey shown in report page
                int rowNo = dgvSQ.CurrentCell.RowIndex;
                ai.surveyID = surveyDataTable.Rows[rowNo][0].ToString();
                ReportForm rf = new ReportForm();
                rf.Show();
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Please Select the survey first",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
```

```
}
```

8. ReportForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace SurveyForm
{
    public partial class ReportForm : Form
    {
        CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter
customerResponseDataset = new
CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter();
        AdminInformation ai = new AdminInformation();
        public ReportForm()
        {
            InitializeComponent();
        }

        private void ReportForm_Load(object sender, EventArgs e)
        {
            string sid = ai.surveyID;
            txtnor.Text = Convert.ToString(customerResponseDataset.CountResponse(sid));

            lstq1.Items.Add("Yes = " + customerResponseDataset.Q1Yes(sid));
            lstq1.Items.Add("No = " + customerResponseDataset.Q1No(sid));

            lstq2.Items.Add("Yes = " + customerResponseDataset.Q2Yes(sid));
            lstq2.Items.Add("No = " + customerResponseDataset.Q2No(sid));
```

```
            lstq3.Items.Add("Choice 1 = "+ customerResponseDataset.Q3_1(sid));
            lstq3.Items.Add("Choice 2 = " + customerResponseDataset.Q3_2(sid));
            lstq3.Items.Add("Choice 3 = " + customerResponseDataset.Q3_3(sid));

            lstq4.Items.Add("Extremely Unimportant = " +
customerResponseDataset.Q4ExtremelyUnimportant(sid));
            lstq4.Items.Add("Unimportant = " +
customerResponseDataset.Q4Unimportant(sid));
            lstq4.Items.Add("Normal = " + customerResponseDataset.Q4Normal(sid));
            lstq4.Items.Add("Important = " + customerResponseDataset.Q4Important(sid));
            lstq4.Items.Add("Extremely Important  = " +
customerResponseDataset.Q4ExtremelyImportant(sid));

            lstq5.Items.Add("Communication = "+
customerResponseDataset.Q5Communication(sid));
            lstq5.Items.Add("Health & Fitness = " +
customerResponseDataset.Q5HealthAndFitness(sid));
            lstq5.Items.Add("Business = " + customerResponseDataset.Q5Business(sid));
            lstq5.Items.Add("Education = " + customerResponseDataset.Q5Education(sid));
            lstq5.Items.Add("Social = " + customerResponseDataset.Q5Social(sid));

            txtq6.Text = Convert.ToString(customerResponseDataset.Q6Count(sid));

        }

        private void btnprint_Click(object sender, EventArgs e)
        {
            try
            {
                string fileName = txtfilename.Text;
                FileStream fs = new FileStream(fileName + ".txt", FileMode.CreateNew,
FileAccess.Write);

                StreamWriter sw = new StreamWriter(fs);
                sw.WriteLine(lblres.Text + " " + txtnor.Text + "\n");

                sw.WriteLine(lblQ1R.Text + " ");
                foreach (var item in lstq1.Items)
                {
```

```
                    sw.WriteLine(item.ToString());
                }
                sw.WriteLine("\n" + lblQ2R.Text + " ");
                foreach (var item in lstq2.Items)
                {
                    sw.WriteLine(item.ToString());
                }
                sw.WriteLine("\n" + lblQ3R.Text + " ");
                foreach (var item in lstq3.Items)
                {
                    sw.WriteLine(item.ToString());
                }
                sw.WriteLine("\n" + lblQ4R.Text + " ");
                foreach (var item in lstq4.Items)
                {
                    sw.WriteLine(item.ToString());
                }
                sw.WriteLine("\n" + lblQ5R.Text + " ");
                foreach (var item in lstq5.Items)
                {
                    sw.WriteLine(item.ToString());
                }

                sw.WriteLine("\n" + lblQ6R.Text + " " + txtq6.Text);
                sw.Flush();
                sw.Close();
                MessageBox.Show("Customers' Responses have been successfully saved as
textual format!", "Report complete", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                txtfilename.Text = string.Empty;
            }
            catch (IOException ioe)
            {
                MessageBox.Show(ioe.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        private void btnclose_Click(object sender, EventArgs e)
        {
            string messege = "Are you sure you want to close the program?";
```

```
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

9. CustomerLogin.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class CustomerLogin : Form
    {
        CustomerDatasetTableAdapters.CustomerTableTableAdapter customerDataset = new
CustomerDatasetTableAdapters.CustomerTableTableAdapter();
        DataTable customerDataTable = new DataTable();
        CustomerInformation ci = new CustomerInformation();
        public CustomerLogin()
```

```
        {
            InitializeComponent();
        }

        private void btnlogin_Click(object sender, EventArgs e)
        {
            ci.username = txtusername.Text;
            ci.password = txtpassword.Text;
            if (txtusername.Text != string.Empty || txtpassword.Text != string.Empty)
            {
                customerDataTable = customerDataset.LoginCustomer(ci.username,
ci.password);
                if (customerDataTable.Rows.Count == 1)
                {
                    ci.customerID = customerDataTable.Rows[0][0].ToString();
                    ci.customerName = customerDataTable.Rows[0][1].ToString();
                    MessageBox.Show(" Warmly welcome " + ci.customerID + " ID " +
ci.customerName, "Login successful");
                    CustomerDashboard cd = new CustomerDashboard();
                    cd.Show();
                    this.Hide();
                }
                else
                {
                    MessageBox.Show("Please make sure your Username and password \n \t
Thank you.", "Login fail", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
            else
            {
                MessageBox.Show("Please completely fill your Username and password \n \t
Thank you.", "Login Incomplete", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void linklbl_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            CustomerRegistration cr = new CustomerRegistration();
            cr.Show();
```

```
            this.Close();
        }

        private void btncancel_Click(object sender, EventArgs e)
        {
            string messege = "Are you sure you want to close the program?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

10. CustomerRegistration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class CustomerRegistration : Form
    {
```

```
        CustomerDatasetTableAdapters.CustomerTableTableAdapter customerDataset = new
CustomerDatasetTableAdapters.CustomerTableTableAdapter();
        DataTable customerDatatable = new DataTable();
        CustomerInformation ci = new CustomerInformation();
        public CustomerRegistration()
        {
            InitializeComponent();
        }
        private void Auto_ID()
        {
            customerDatatable = customerDataset.GetData();
            if (customerDatatable.Rows.Count == 0)
            {
                txtid.Text = "Customer-001";
            }
            else
            {
                int num = customerDatatable.Rows.Count - 1;
                string oldId = customerDatatable.Rows[num][0].ToString();
                int newId = Convert.ToInt32(oldId.Substring(9, 3));

                if (newId >= 1 && newId < 9)
                {
                    txtid.Text = "Customer-00" + (newId + 1);
                }
                else if (newId >= 9 && newId < 99)
                {
                    txtid.Text = "Customer-0" + (newId + 1);
                }
                else if (newId >= 99 && newId <= 999)
                {
                    txtid.Text = "Customer-" + (newId + 1);
                }
            }
        }
        private void txtGetData()
        {
            ci.customerID = txtid.Text;
            ci.customerName = txtname.Text;
```

```
            ci.dob = txtdob.Text;
            ci.country = txtcountry.Text;
            ci.city = txtcity.Text;
            ci.address = txtaddress.Text;
            ci.phnumber = txtphnumber.Text;
            ci.email = txtemail.Text;
            ci.username = txtusername.Text;
            ci.password = txtpassword.Text;
            ci.gender = null;
            if (rdomale.Checked == true)
            {
                ci.gender = "M";
            }
            else if(rdofemale.Checked == true)
            {
                ci.gender = "F";
            }
        }

        private void CustomerRegister_Load(object sender, EventArgs e)
        {
            Auto_ID();
        }

        private void btnregister_Click(object sender, EventArgs e)
        {
            if ( txtname.Text != string.Empty || txtdob.Text != string.Empty ||
txtcountry.Text != string.Empty || txtcity.Text != string.Empty || txtaddress.Text !=
string.Empty || txtphnumber.Text != string.Empty || txtemail.Text != string.Empty ||
txtusername.Text != string.Empty || txtpassword.Text != string.Empty)
            {
                txtGetData();
                if (ci.gender == null)
                {
                    ci.gender = "other";
                }
                customerDataset.SaveCustomer(ci.customerID, ci.customerName, ci.dob,
ci.gender, ci.country, ci.city, ci.address, ci.phnumber, ci.email, ci.username,
ci.password);
```

```
                MessageBox.Show("The registeration has been successful. \n \t Thank
you.", "Registeration successful! ", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                CustomerLogin cl = new CustomerLogin();
                cl.Show();
                this.Close();
            }
            else
            {
                MessageBox.Show("Please Completely fill in the form! ", "Registration
fail", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void chkaccept_CheckedChanged(object sender, EventArgs e)
        {
            btnregister.Enabled = true;
        }

        private void btncancel_Click(object sender, EventArgs e)
        {
            string messege = "Do you want to exit?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }

        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs
e)
        {
            CustomerLogin cl = new CustomerLogin();
```

```
            cl.Show();
            this.Hide();
        }

    }
}
```

11. CustomerDashboard.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class CustomerDashboard : Form
    {
        CustomerDatasetTableAdapters.CustomerTableTableAdapter customerDataset = new
CustomerDatasetTableAdapters.CustomerTableTableAdapter();
        SurveyDatasetTableAdapters.SurveyTableTableAdapter surveyDataset = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter
customerResponseDataset = new
CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter();
        DataTable customerDataTable = new DataTable();
        DataTable customerResponseDataTable = new DataTable();
        DataTable surveyDataTable = new DataTable();
        CustomerInformation ci = new CustomerInformation();
        SurveyInformation si = new SurveyInformation();
        public CustomerDashboard()
        {
            InitializeComponent();
        }
```

```
private void GenderCount()
{
    //get all customer data
    customerDataTable = customerDataset.GetData();
    int maleCount = 0;
    int femaleCount = 0;

    //count number of male and female
    for (int i = 0; i < customerDataTable.Rows.Count; i++)
    {
        string tempGender = customerDataTable.Rows[i][3].ToString();
        if (tempGender == "M")
        {
            maleCount++;
        }
        else
        {
            femaleCount++;
        }
    }
    lblmalecount.Text = Convert.ToString(maleCount);
    lblfemalecount.Text = Convert.ToString(femaleCount);

    //count number of respondant
    lblgendercount.Text = Convert.ToString(maleCount + femaleCount);
}
private void CustomerDashboard_Load(object sender, EventArgs e)
{
    GenderCount();
    customerDataTable = customerDataset.GetCustomerData(ci.customerID);
    lblname.Text = customerDataTable.Rows[0][1].ToString();
    lblusername.Text = customerDataTable.Rows[0][9].ToString();

    customerDataTable = customerResponseDataset.GetResponse();
    lblresponsecount.Text = customerDataTable.Rows.Count.ToString();

    surveyDataTable = surveyDataset.GetData();
    dgvsurveys.DataSource = surveyDataTable;
    for (int i = 4; i < 11 ; i++)
```

```
            {
                dgvsurveys.Columns[i].Visible = false;
            }
            dgvsurveys.Refresh();
        }

        private void dgvsurveys_CellContentClick(object sender, DataGridViewCellEventArgs
e)
        {
            int rowNo = dgvsurveys.CurrentCell.RowIndex;
            string surveyid = surveyDataTable.Rows[rowNo][0].ToString();
            ci.surveyID = surveyid;

            lblsurveyid.Text = surveyDataTable.Rows[rowNo][0].ToString();
            lbladminname.Text = surveyDataTable.Rows[rowNo][3].ToString();
        }

        private void picboxyes_Click(object sender, EventArgs e)
        {
            if (ci.surveyID != null)
            {
                CustomerAnswerSurvey cas = new CustomerAnswerSurvey();
                cas.Show();
                this.Hide();
            }
            else
            {
                MessageBox.Show("Please Select the Item! ", "Fail", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void picno_Click(object sender, EventArgs e)
        {
            lblsurveyid.Text = "";
            lbladminname.Text = "";
            ci.surveyID = null;
        }
```

```
        private void historyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CustomerHistory ch = new CustomerHistory();
            ch.Show();
        }

        private void picboxenter_Click(object sender, EventArgs e)
        {
            panelinfo.Show();
            dgvsurveys.Show();
        }

        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            string messege = "Are you sure you want to close the program?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

12. CustomerAnswerSurvey.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections;

namespace SurveyForm
{
    public partial class CustomerAnswerSurvey : Form
    {
        SurveyDatasetTableAdapters.SurveyTableTableAdapter SurveyDataSet = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter
CustomerResponseDataset = new
CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter();
        DataTable surveyDataTable = new DataTable();
        CustomerInformation ci = new CustomerInformation();
        string q1, q2, q3, q4, q5;
        int q6;
        string datetime = DateTime.Now.ToString("d-M-yy");
        public CustomerAnswerSurvey()
        {
            InitializeComponent();
        }

        private void CustomerAnswerSurvey_Load(object sender, EventArgs e)
        {

                surveyDataTable = SurveyDataSet.ShowSurveyQuestions(ci.surveyID);
                lbldate.Text = datetime;
                lblsurveyid.Text = ci.surveyID;
                lblcustomername.Text = ci.customerName;
                lblsurveyname.Text = surveyDataTable.Rows[0][1].ToString();
                gbq1.Text = surveyDataTable.Rows[0][5].ToString();
                gbq2.Text = surveyDataTable.Rows[0][6].ToString();
                gbq3.Text = surveyDataTable.Rows[0][7].ToString();
                gbq4.Text = surveyDataTable.Rows[0][8].ToString();
                gbq5.Text = surveyDataTable.Rows[0][9].ToString();
                gbq6.Text = surveyDataTable.Rows[0][10].ToString();
        }
```

```
private string Saveq1()
{
    q1 = "";
    if (rdoyesq1.Checked == true)
    {
        q1 = rdoyesq1.Text;
        return q1;
    }
    else
    {
        q1 = rdonoq1.Text;
        return q1;
    }
}
private string Saveq2()
{
    q2 = "";
    if (rdoyesq1.Checked == true)
    {
        q2 = rdoyesq2.Text;
        return q2;
    }
    else
    {
        q2 = rdonoq2.Text;
        return q2;
    }
}
private string Saveq3()
{
    q3 = "";
    if (rdo1q3.Checked == true)
    {
        q3 = rdo1q3.Text;
        return q3;
    }
    else if (rdo2q3.Checked == true)
    {
```

```
            q3 = rdo2q3.Text;
            return q3;
        }
        else
           {
            q3 = rdo3q3.Text;
            return q3;
           }
    }
    private string Saveq4()
    {
        q4 = "";
        if (rdo1q4.Checked == true)
        {
            q4 = rdo1q4.Text;
            return q4;
        }
        else if (rdo2q4.Checked == true)
        {
            q4 = rdo2q4.Text;
            return q4;
        }
        else if (rdo3q4.Checked == true)
        {
            q4 = rdo3q4.Text;
            return q4;
        }
        else if(rdo4q4.Checked == true)
        {
            q4 = rdo4q4.Text;
            return q4;
        }
        else
        {
            q4 = rdo5q4.Text;
            return q4;
        }
    }
    private string Saveq5()
```

```
        {
            q5 = cbochoose.SelectedItem.ToString();
            return q5;
        }
        private int Saveq6()
        {
            if (rdo0q6.Checked == true)
            {
                q6 = 0;
                return q6;
            }
            else if (rdo1q6.Checked == true)
            {
                q6 = 1;
                return q6;
            }
            else if (rdo2q6.Checked == true)
            {
                q6 = 2;
                return q6;
            }
            else if (rdo3q6.Checked == true)
            {
                q6 = 3;
                return q6;
            }
            if (rdo4q6.Checked == true)
            {
                q6 = 4;
                return q6;
            }
            else if (rdo5q6.Checked == true)
            {
                q6 = 5;
                return q6;
            }
            else if (rdo6q6.Checked == true)
            {
                q6 = 6;
```

```
                return q6;
            }
            else if (rdo7q6.Checked == true)
            {
                q6 = 7;
                return q6;
            }
            else if (rdo8q6.Checked == true)
            {
                q6 = 8;
                return q6;
            }
            else if (rdo9q6.Checked == true)
            {
                q6 = 9;
                return q6;
            }
            else
            {
                q6 = 10;
                return q6;
            }
        }
        private void btnsave_Click(object sender, EventArgs e)
        {
            try
            {
                //receiving responses
                Saveq1();
                Saveq2();
                Saveq3();
                Saveq4();
                Saveq5();
                Saveq6();
                //transfering the response to database
                CustomerResponseDataset.SaveResponse(ci.surveyID, ci.customerName,
ci.customerID, datetime, q1, q2, q3, q4, q5, q6);
                MessageBox.Show("Your response is successfully saved! ", "Successful",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
```

```
                CustomerDashboard cd = new CustomerDashboard();
                cd.Show();
                this.Close();
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Please fill the form completely!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnback_Click(object sender, EventArgs e)
        {
            //return back process
            string messege = "Do you want to exit?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                CustomerDashboard cd = new CustomerDashboard();
                cd.Show();
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
    }
}
```

13. CustomerHistory.cs

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

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace SurveyForm
{
    public partial class CustomerHistory : Form
    {
        CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter
customerResponseDataset = new
CustomerDatasetTableAdapters.CustomerResponseTableTableAdapter();
        SurveyDatasetTableAdapters.SurveyTableTableAdapter surveyDataset = new
SurveyDatasetTableAdapters.SurveyTableTableAdapter();
        DataTable surveyTable = new DataTable();
        DataTable responseTable = new DataTable();
        CustomerInformation ci = new CustomerInformation();
        public CustomerHistory()
        {
            InitializeComponent();
        }

        private void CustomerHistory_Load(object sender, EventArgs e)
        {
            //show responses
            string CusID = ci.customerID;
            responseTable = customerResponseDataset.GetCustomerResponse(CusID);
            dgvhistory.DataSource = responseTable;
            dgvhistory.Columns[1].Visible = false;
            dgvhistory.Columns[2].Visible = false;
            dgvhistory.Refresh();

            // searching
            int rowsCount = responseTable.Rows.Count;
            for (int i = 0; i < rowsCount; i++)
            {
                cboSurvey.Items.Add(responseTable.Rows[i][0].ToString());
            }
```

```
        }
        private void btnsearch_Click(object sender, EventArgs e)
        {
            try
            {
                //search answered surveys
                string sid = cboSurvey.SelectedItem.ToString();
                responseTable = customerResponseDataset.SearchSurvey(sid);
                dgvhistory.DataSource = responseTable;
                dgvhistory.Refresh();
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Select item before searching",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
            }
        }
        private void picboxclose_Click(object sender, EventArgs e)
        {
            string messege = "Are you sure you want to close the program?";
            string title = "Confirmation";
            MessageBoxButtons buttons = MessageBoxButtons.OKCancel;
            DialogResult result = MessageBox.Show(messege, title, buttons,
MessageBoxIcon.Exclamation);
            if (result == DialogResult.OK)
            {
                this.Close();
            }
            else if (result == DialogResult.Cancel)
            {
                this.Show();
            }
        }
        private void picboxclick_Click(object sender, EventArgs e)
        {
            dgvhistory.Show();
        }
    }
}
```

1.4 Design and Justification of Design

Being on the cutting edge of technology, the Apple company focuses on details from administrative pages to user interfaces.

1. Welcome page

The first and foremost page for the users is the welcome page which is the common page for asking the options. The pictures act like buttons and the labels explain which picture is for admin or customer at the bottom of each picture.

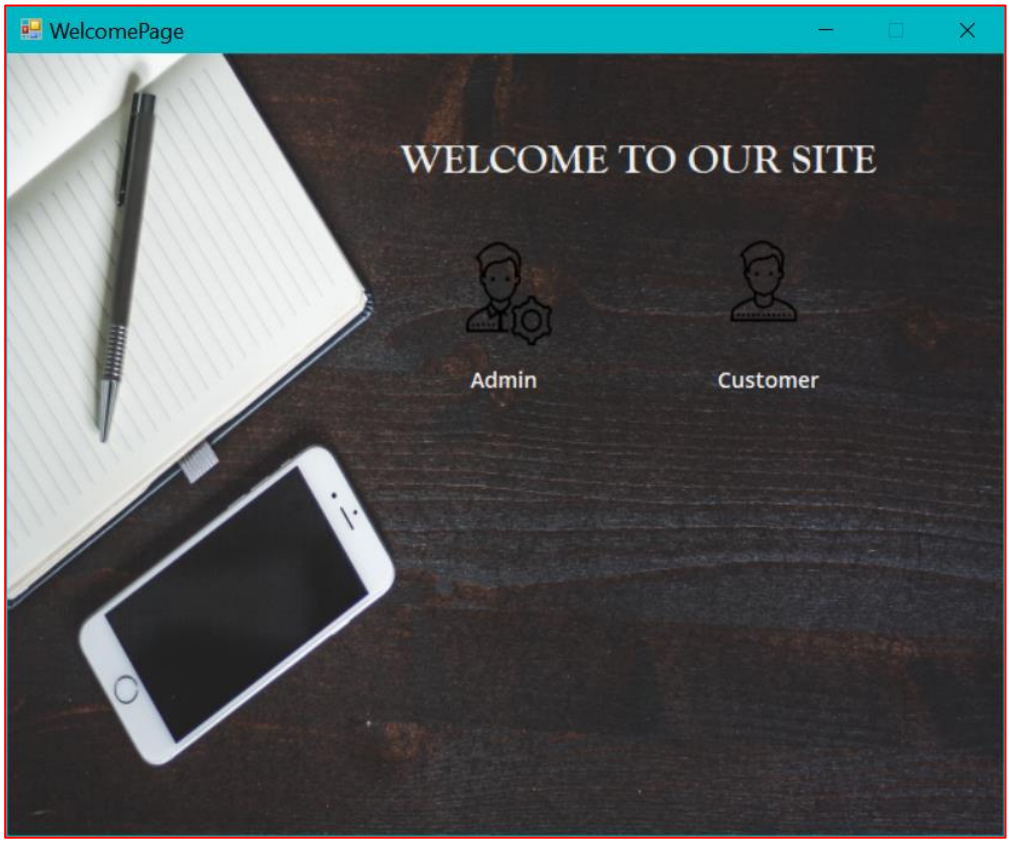


Figure 1

2. Admin Registration page

The Dark gray color is applied to the heading to conform to the background color. Each label describes what must be filled in the textbox. The white-colored labels are suitable for smoky background image. The password characters are not shown for providing the security.

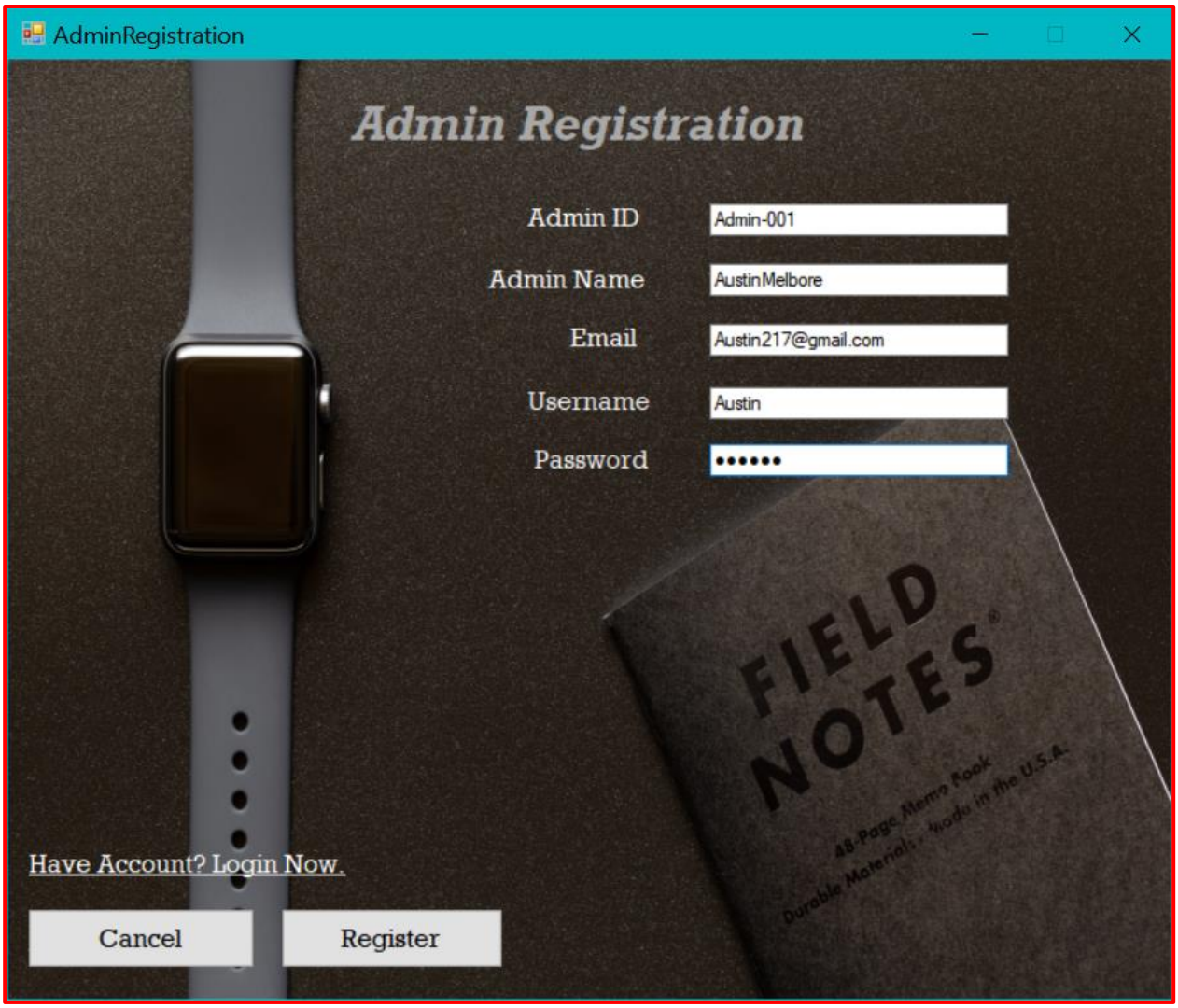


Figure 2

3. Admin Login page

The background image of Admin login bears a significant resemblance to that of Admin Registration page. The bold italic heading is used to highlight. Password characters are not shown in the textbox to prioritize security.

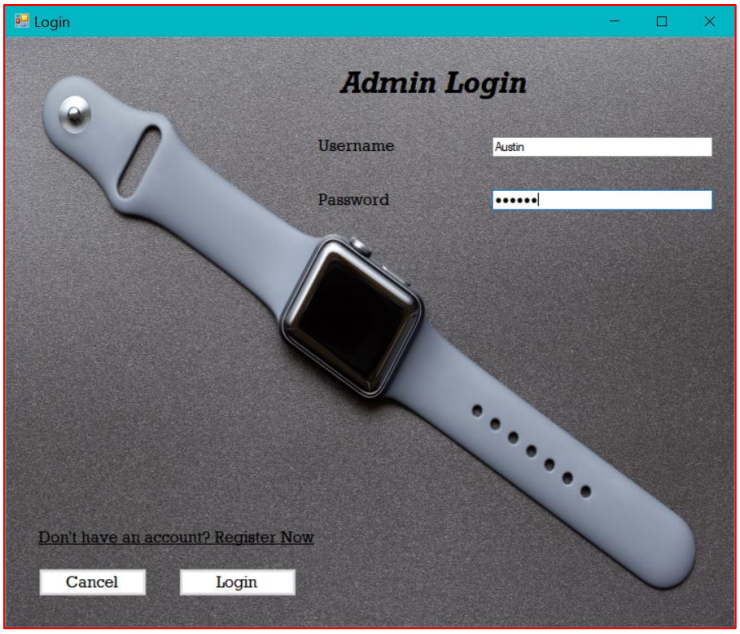


Figure 3

4. Admin Dashboard page

The left labels are just for description but the right ones are varying in terms of admins login. The black and white photographs are utilized to prioritize simplicity and novelty. The pop-up Datetime picker is placed at the top-right corner to ease the taking hand off the mouse to type in a date. It is the major home page linking to all the pages. The menu bar indicates the available page to browse.

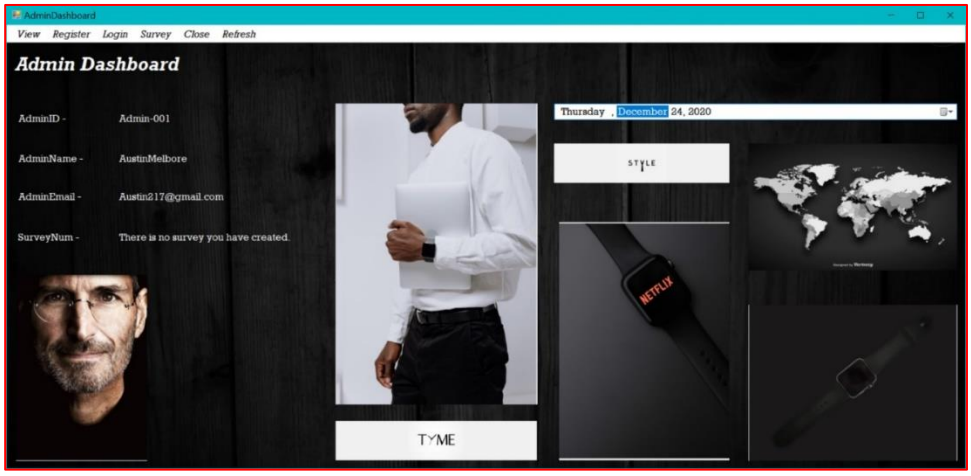


Figure 4

5. Admin View List page

The table describes the particular information at the top-left corner of the page. The information source is all about admin name, username and email of registered admins. The green-colored button is used in accordance with the background image. The table is possible way to visualize the information more clearly.

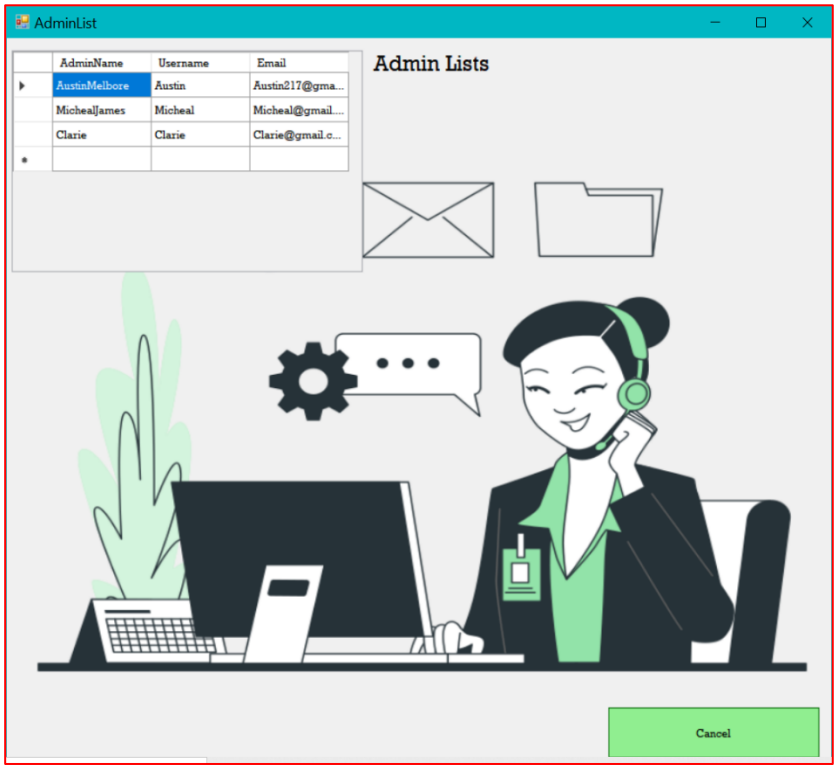


Figure 5

6. Admin Create Survey page

The page can be divided into two sections, survey information and survey questions. In the first section, the labels are just representations of what types of information are in the textboxes. The textboxes are not required to be filled in manually as they automatically express when the page is loaded. In the second section, the label positions are contrary to the first ones to show distinctive design. The textboxes are a little bit larger than that of first section.

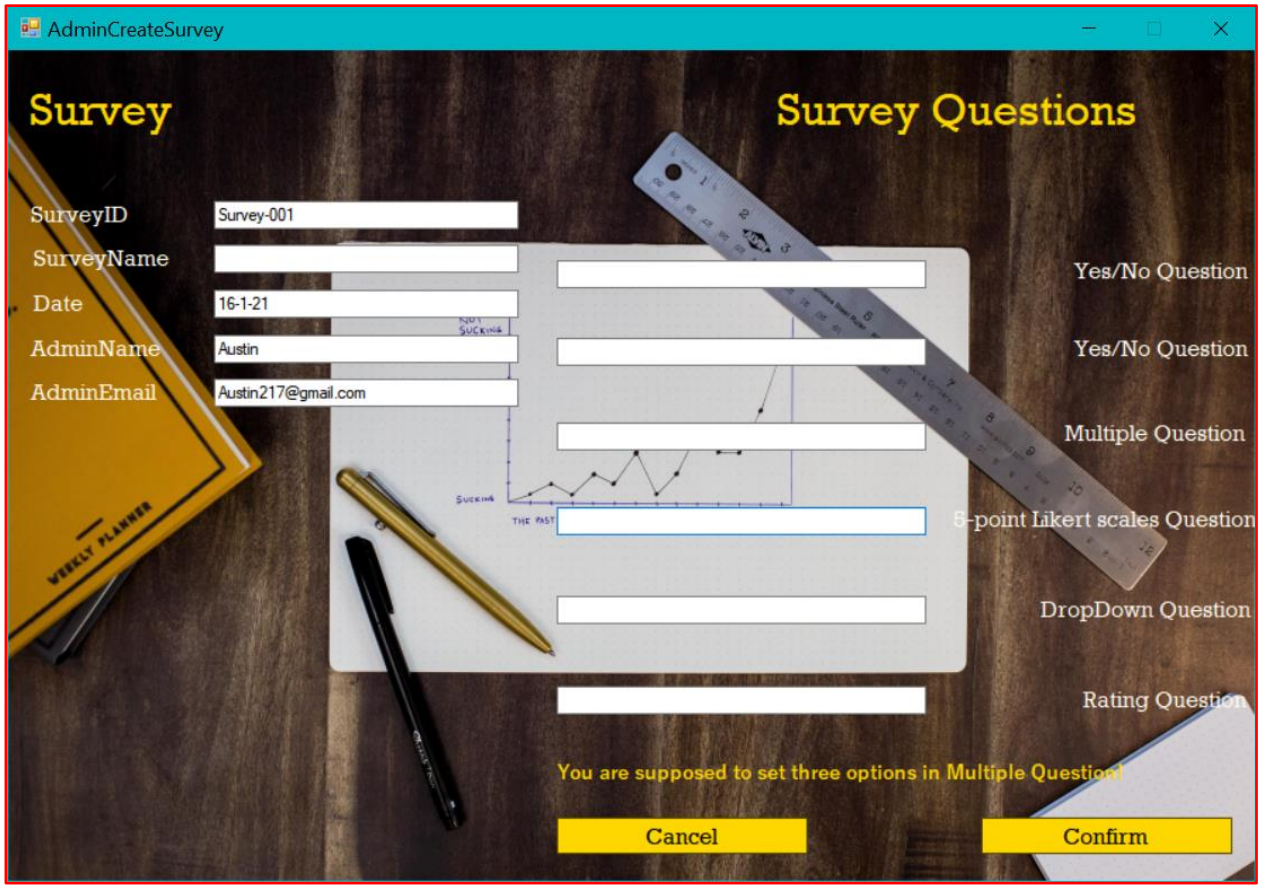


Figure 6

7. Admin View Survey page

The survey table covers a significant top area to illustrate the survey questions. The dropdown is placed at the left and the textboxes at the right. The underlined black-label is an instruction before processing. Close button is created at the bottom-left corner not to confuse with others.

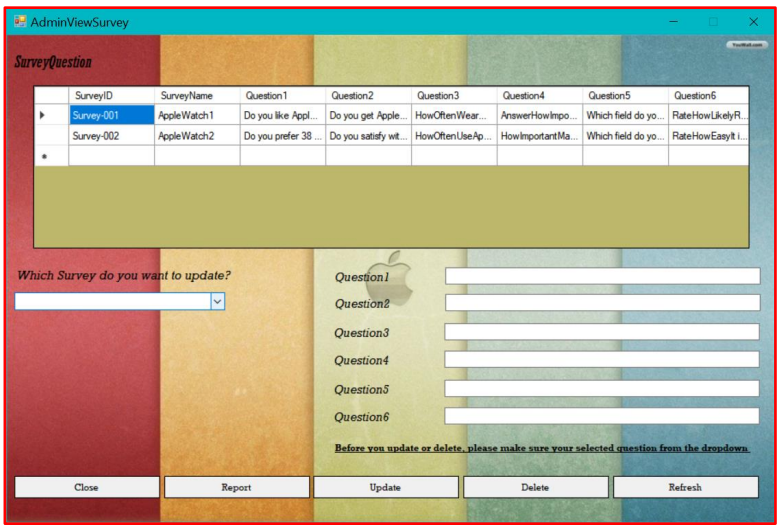


Figure 7

8. Report form page

The list boxes are placed differently to illustrate the company logo. The yellow green color is used to show the feeling of freshness. The labels are yellow which is complementary to green color. The long textbox and brown “Print” button are positioned at the bottom of the screen.

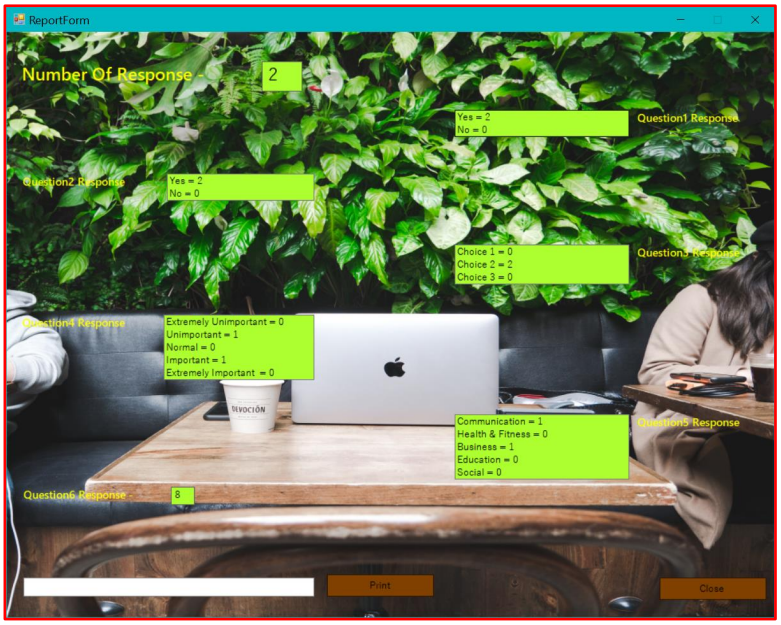


Figure 8

9. Customer Login page

The left labels show what to fill in the respective textboxes. The black labels are set to emphasize.

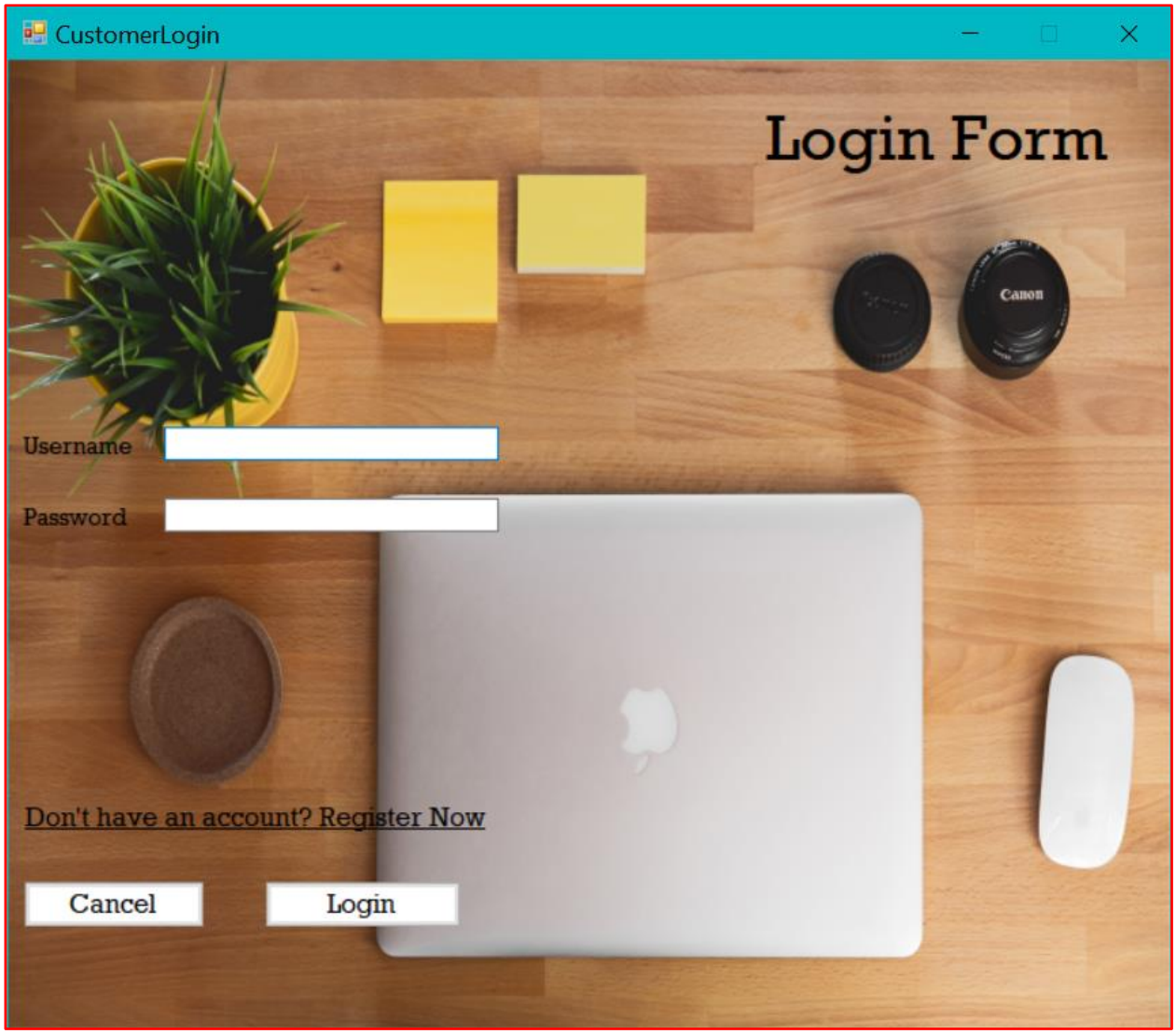


Figure 9

10. Customer registration page

The font and size of heading is different from those of labels. The bold labels are placed at the left side while the textboxes are beside those. The checkbox at the bottom is to make confirmation and the blue link label is beside it.

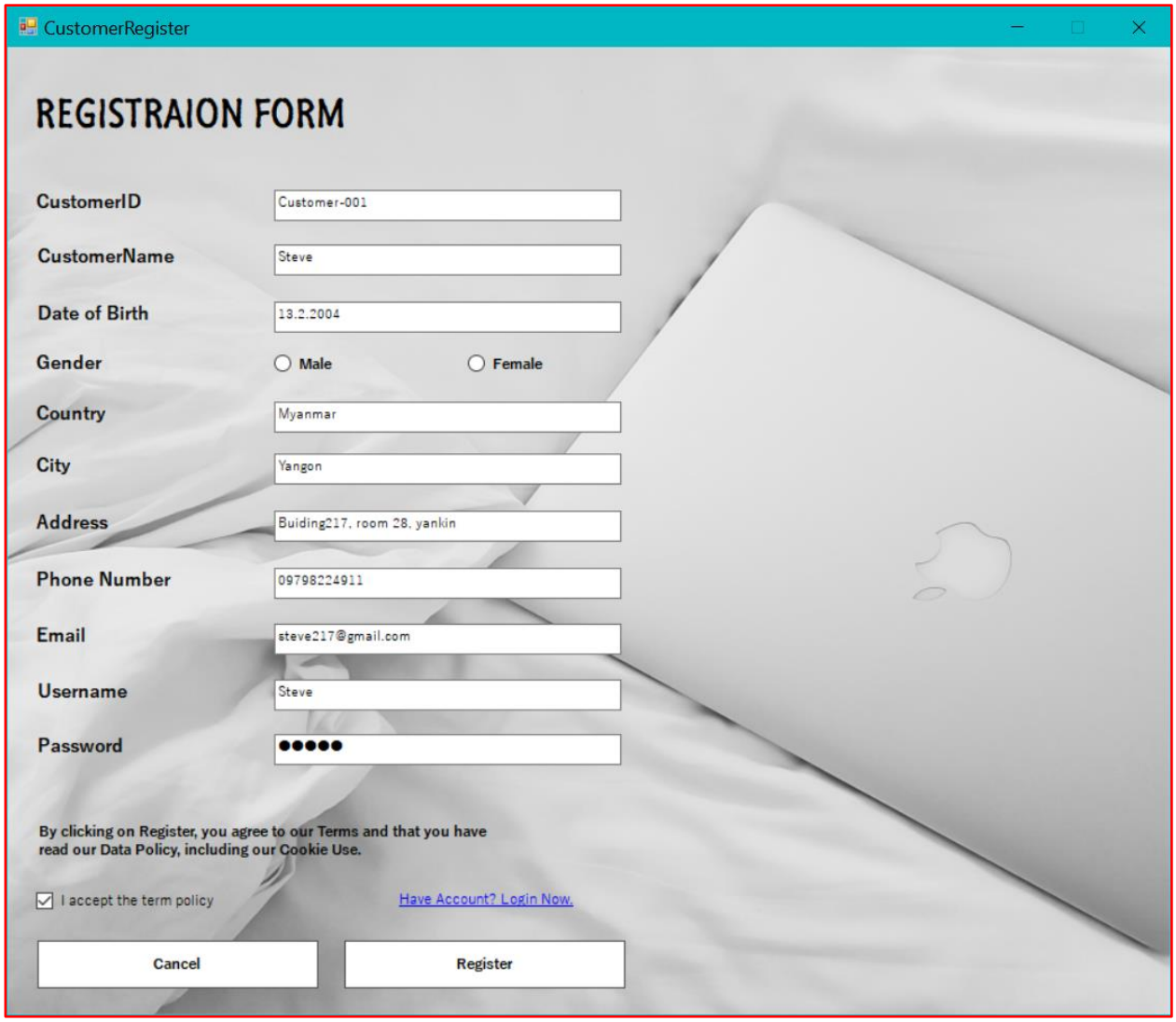


Figure 10

11. Customer dashboard page

The customer dashboard is created with embedded designs emphasizing for attention-grabbing. The top-right picture and left-bottom table will appear only if clicking on the smart watch which is at the middle of the screen. The gold-colored labels are only descriptions and instructions. The black menu bar has two options. The spaces are created to show flexible design in the question sentence. Pictures are utilized to perform Yes and No functions.

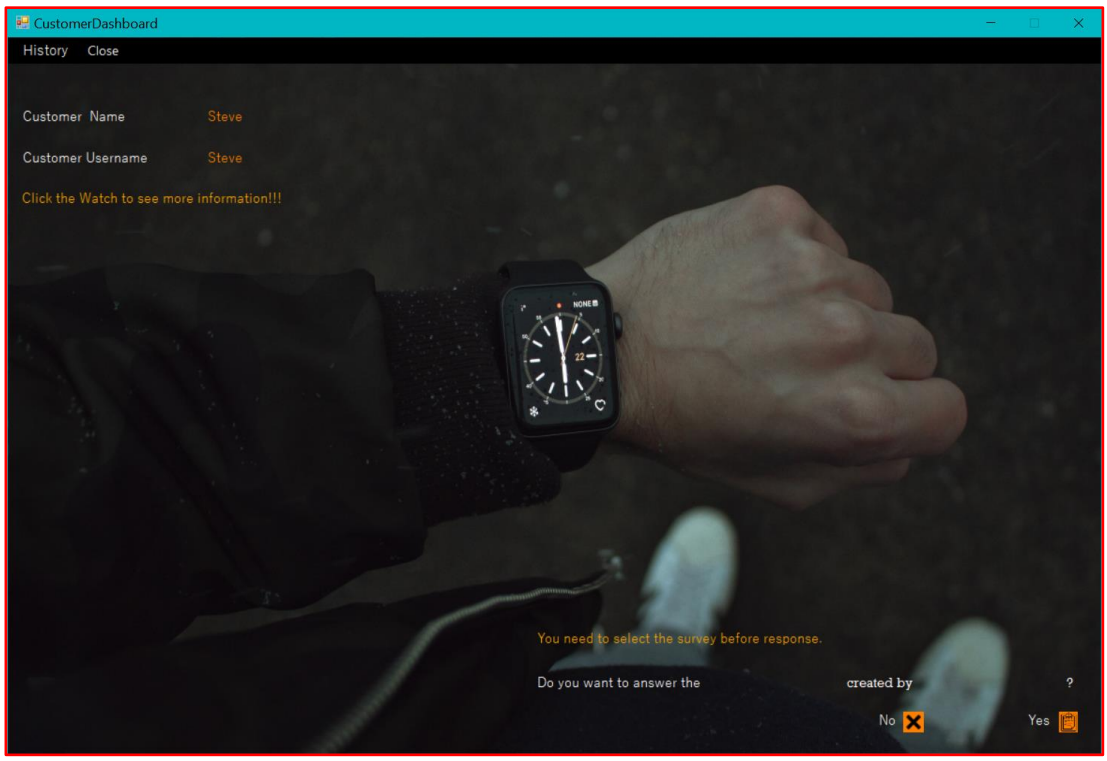


Figure 11 Before clicking the picture

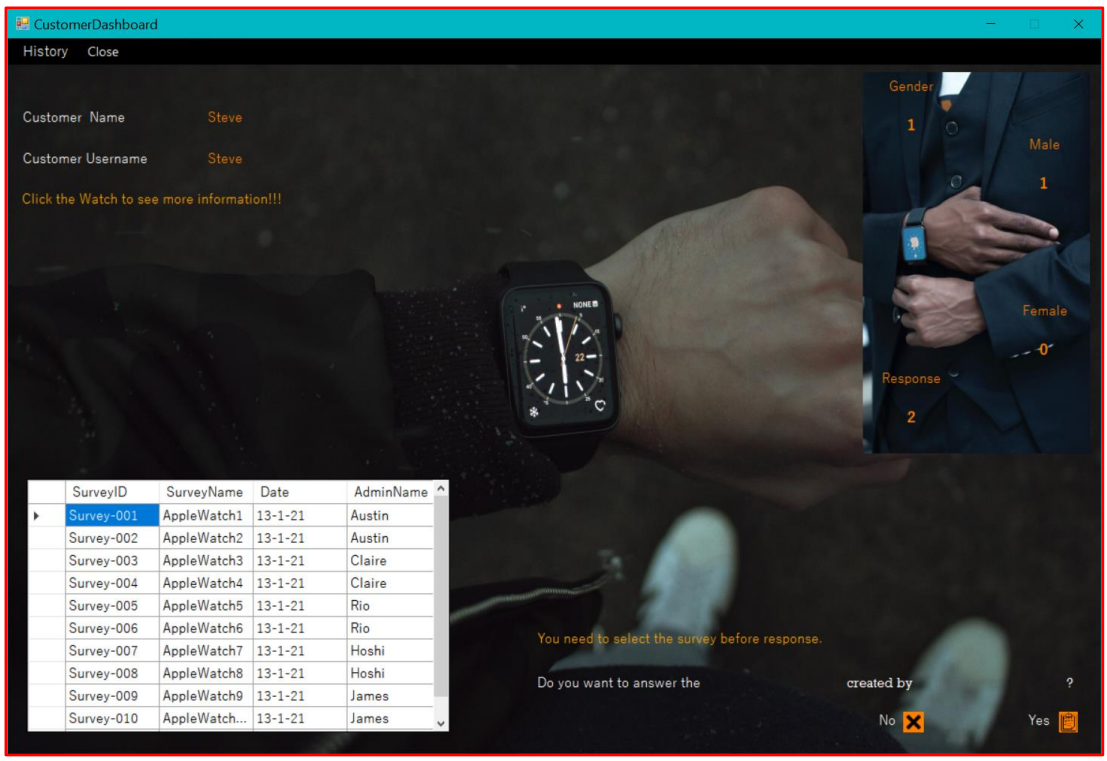


Figure 12 After clicking the picture

12. Customer history page

The white-colored search box is placed at the top of screen and the red-colored label is instruction. “Search” button is beside it for instant use. Click the picture to show up the customer’s answered surveys table. Red cross is designed for closing function.

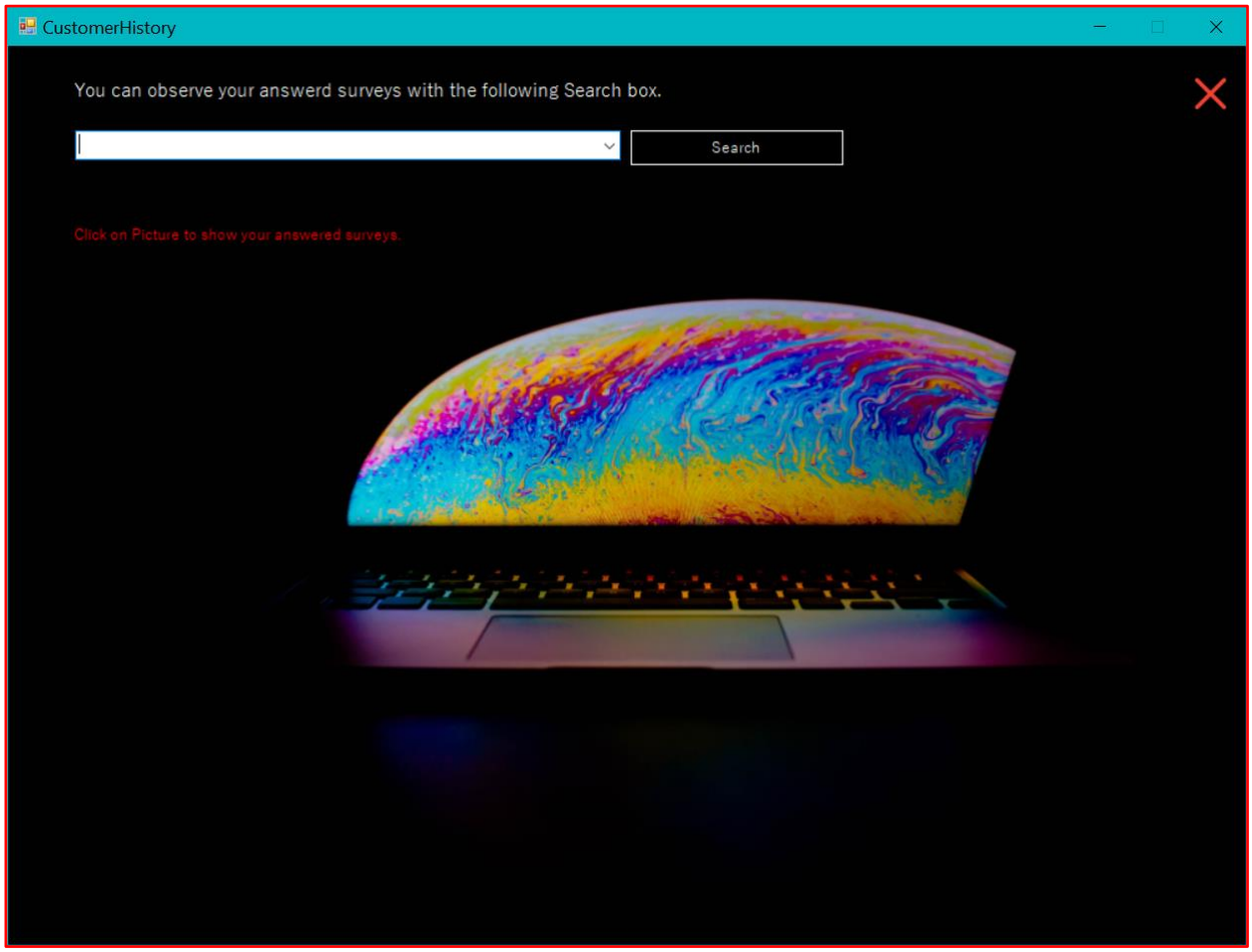


Figure 13 Before clicking on the picture

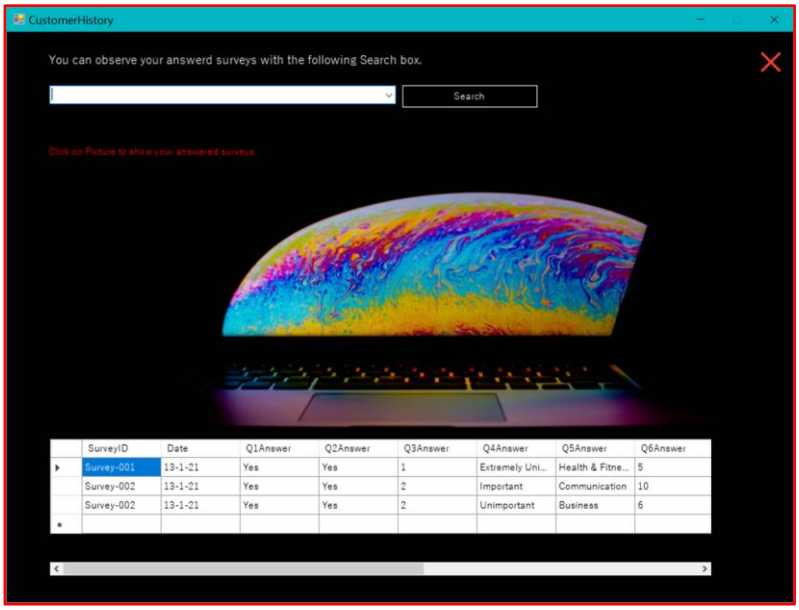


Figure 14 After clicking on the picture

13. Customer Answer Survey page

Each question of selected survey will be placed in each group box. Red-colored buttons are used to match with the background image. Radio buttons are presented for options and drop-down list provides more choices. The customer name, answering survey name and date are designed at the top-right of the screen.

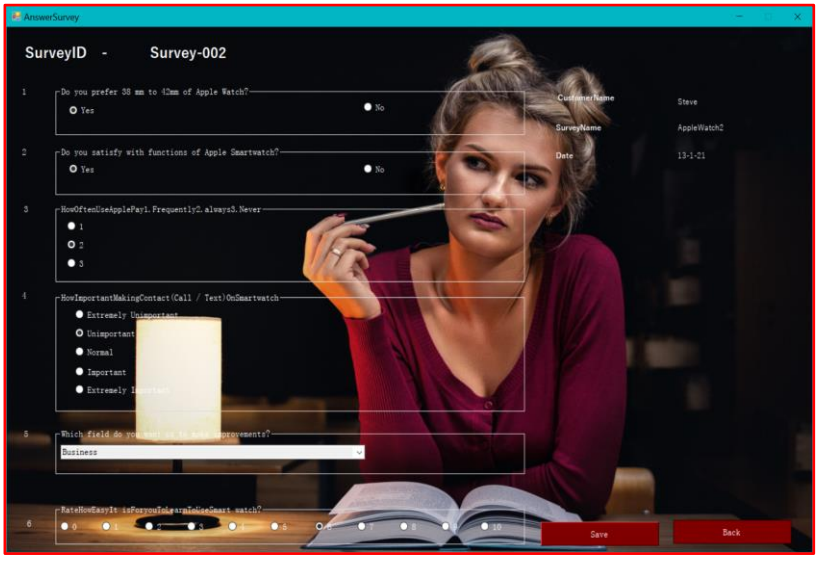


Figure 15

1.5Program Structure

As soon as the program starts, it will ask users for two options. The user must choose one option according to his role to continue. Clicking on the picture will allow to go to the respective pages.

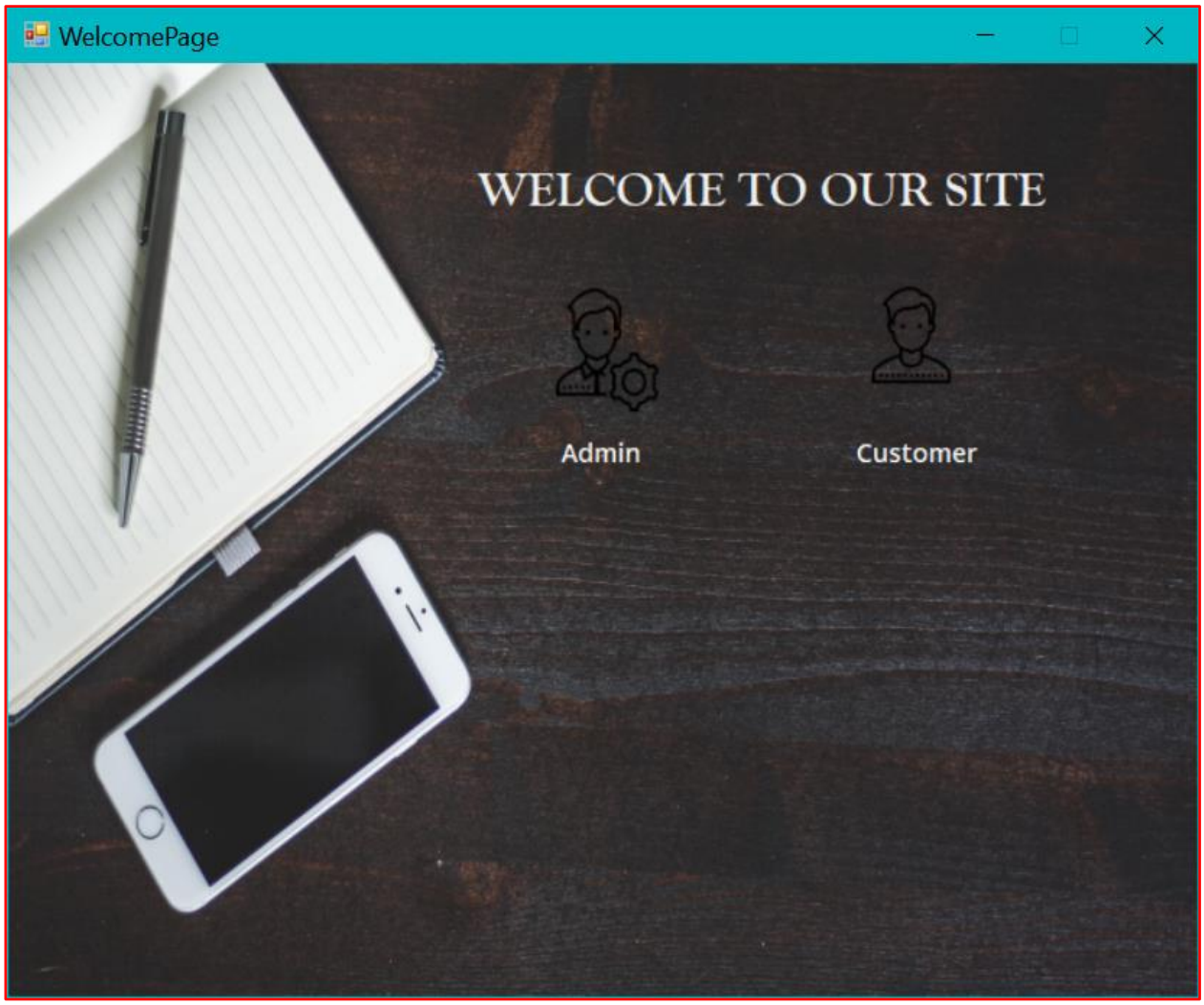


Figure 16

For Administrators

Administrative registration page is the initial one that accumulates admin information to conduct surveys. This process needs to be accomplished so that the admin is capable of creating survey with his secure account. But the registered admin does not require to sign up again. Even one incomplete fill-in raises null reference exception.

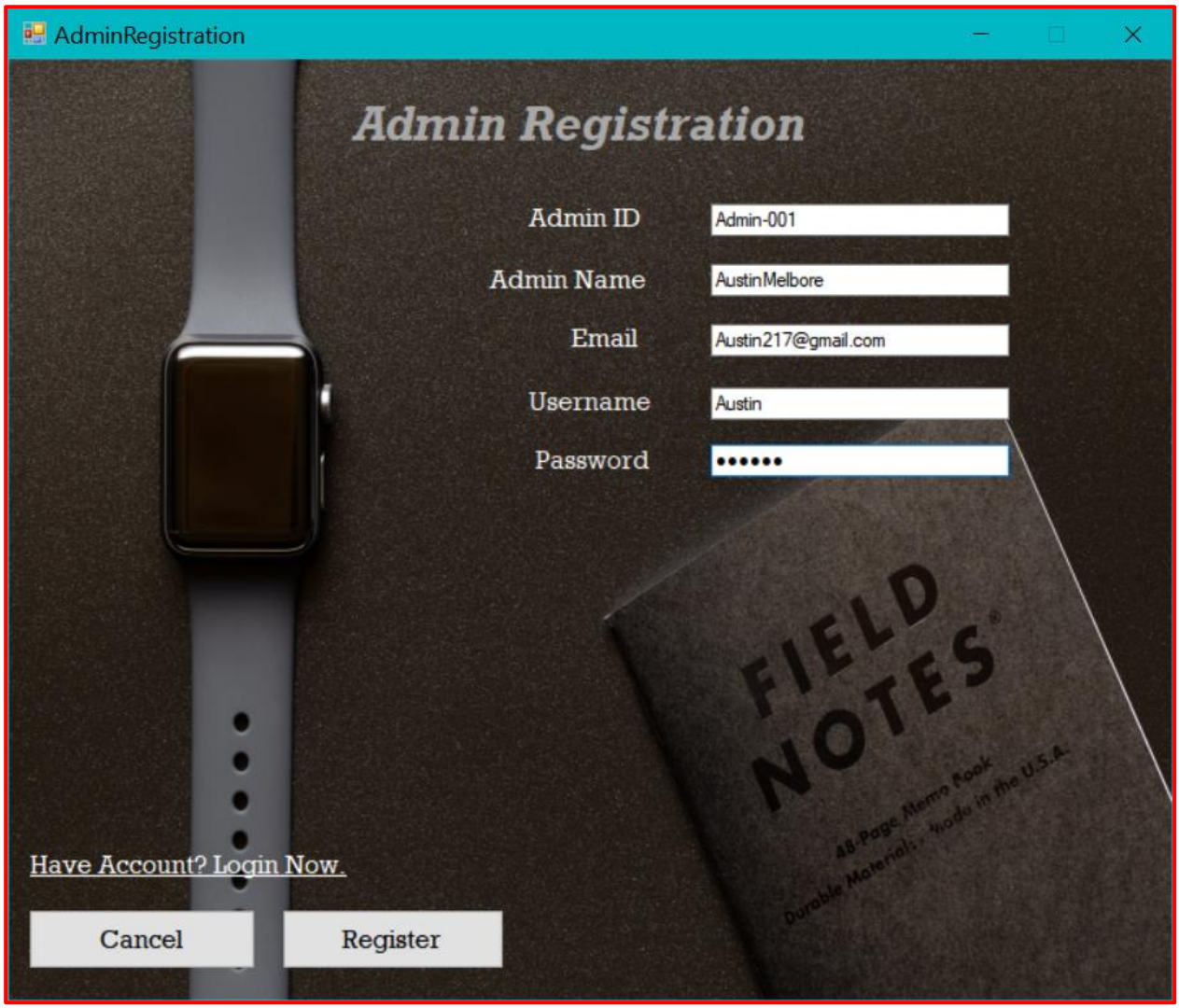


Figure 17

The admin Login page will also throw null reference exception when one of the information is forgotten to fill up. Whenever clicking on buttons, there will always be dialog boxes to express confirmation and successful message.

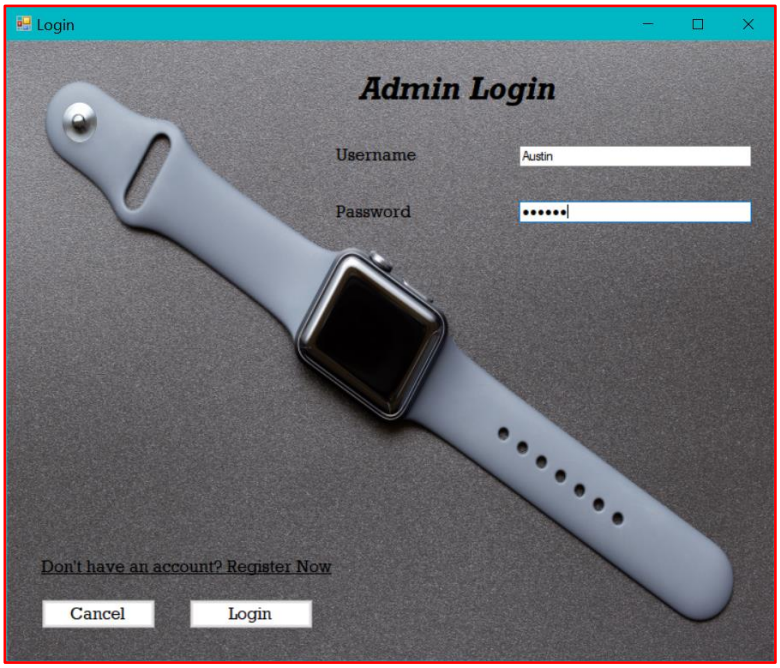


Figure 18

As soon as the admin login process has been accomplished, the Administrative Dashboard provides administrators direct access to manually upload new surveys to the sever, to edit them and to delete the unwanted ones. It serves as an administrative home page for the purposes of faster and more compatible community management.

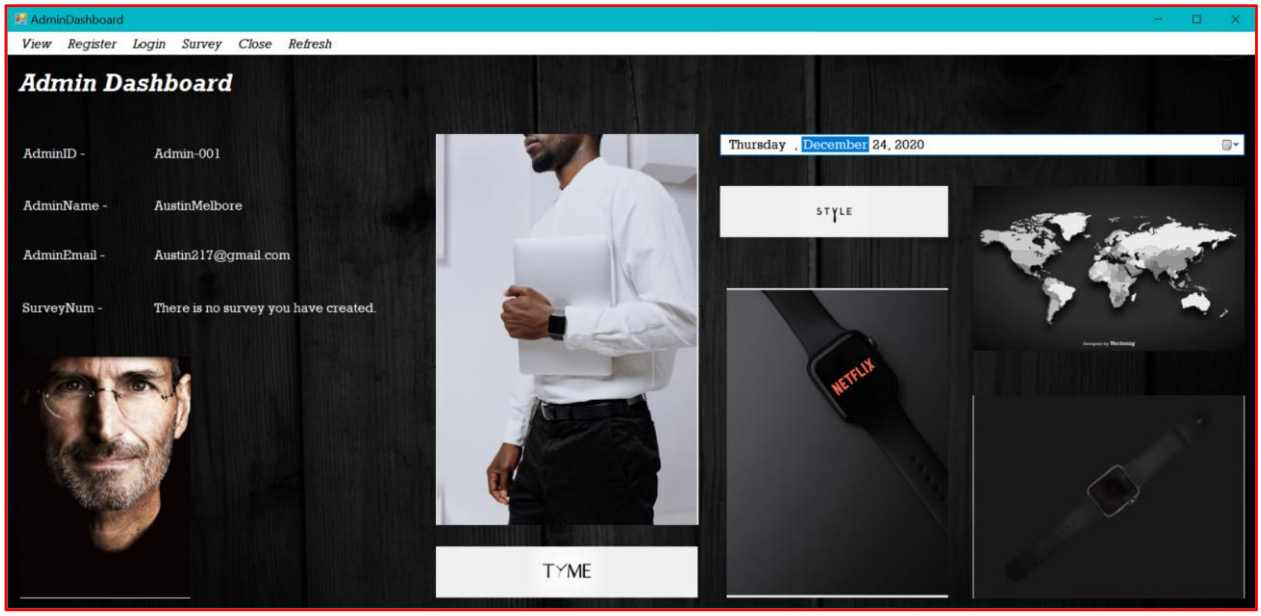


Figure 19

Admin View List page is designed only for viewing registered admins and no specific activity would be able to be carried out here.

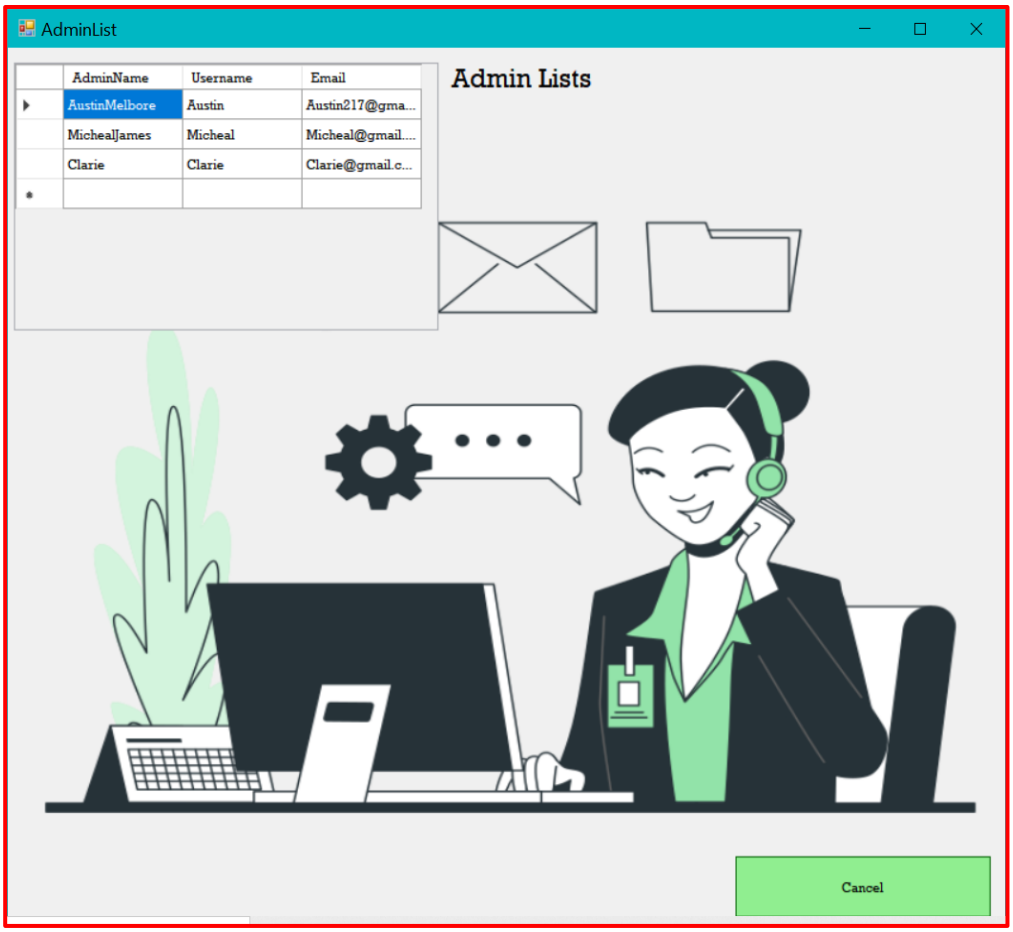


Figure 20

The following Create Survey page gives access to create surveys about the apple watch for the aim of customer feedback. Customer surveys are essential to comprehend expectation, perceptions, satisfaction and areas for improvement in order that the business can make a quick response to satisfy the customers. The textboxes of survey ID, date, admin Name and admin Email are automatically filled up as soon as the page loads. The input data will be received by the textboxes and transferred to the database after the “Confirm” button being hit.

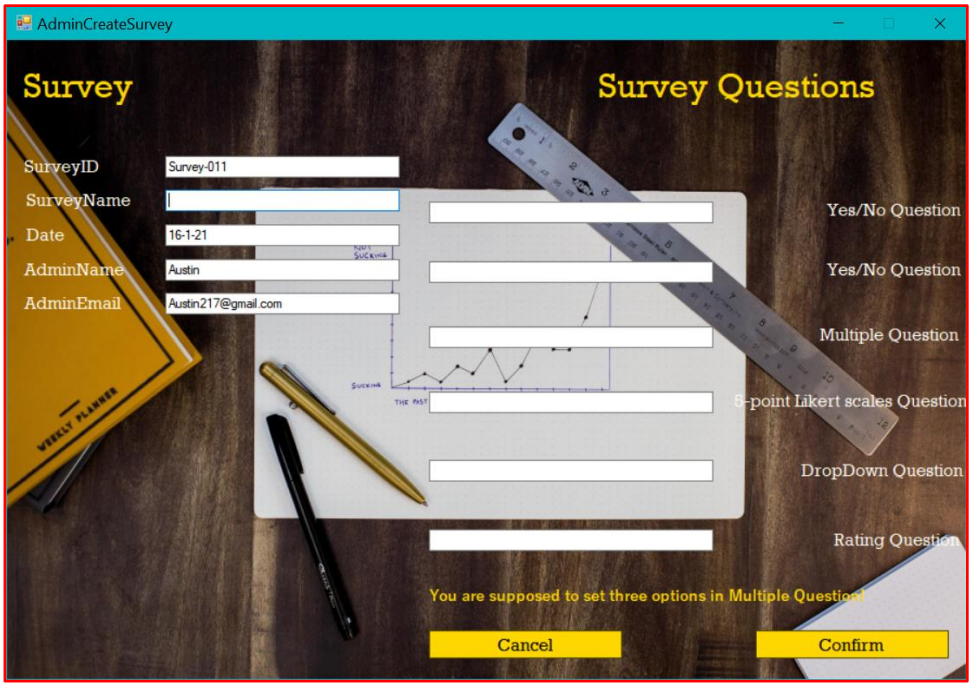


Figure 21

The next following page is for updating the created surveys to keep abreast of clients' requirement and to delete them in case they are ineffective in asking for customer experience.

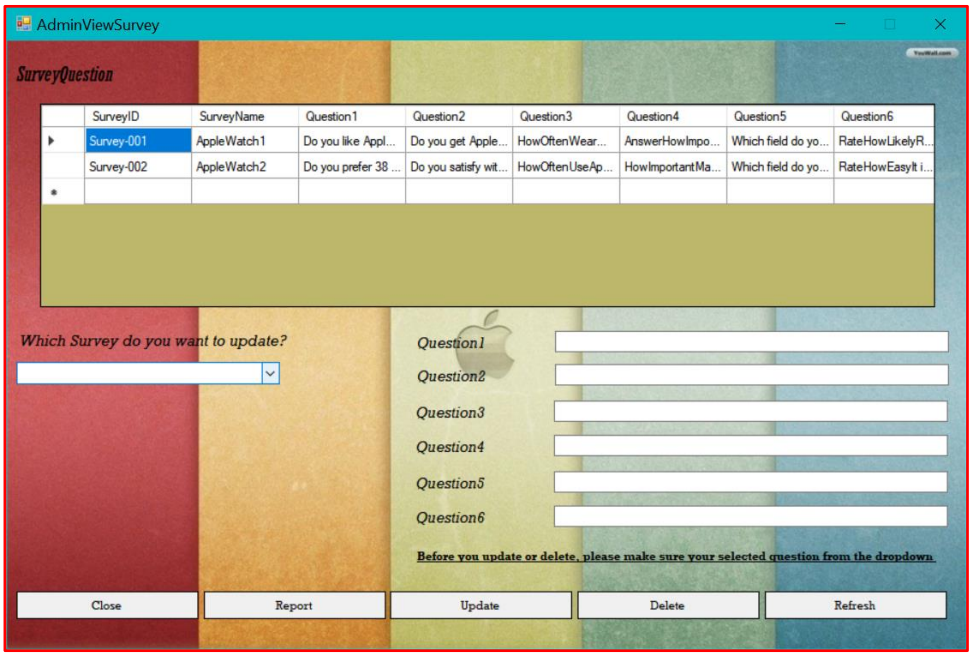


Figure 22

In this page, only clicking on the Data-grid cell of survey question causes the respective questions to show in the respective textboxes. And the survey question has been selected in the drop-down. This is to make changes in survey easily rather than the manual process.

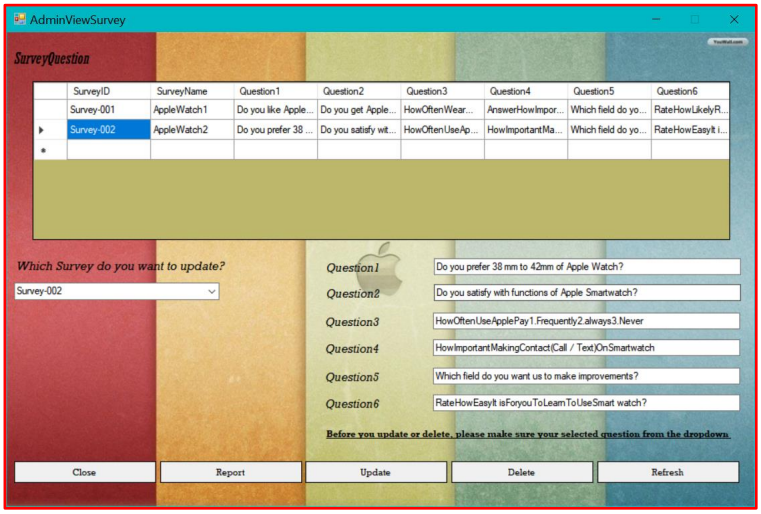


Figure 23

After the delete and update processes have been accomplished, clicking on “Refresh” button generates the latest survey questions instead of closing the page. Administrators are capable of reporting surveys so report form page is constructed. To perform the report task, go to the report form page by clicking on the “Report” button.

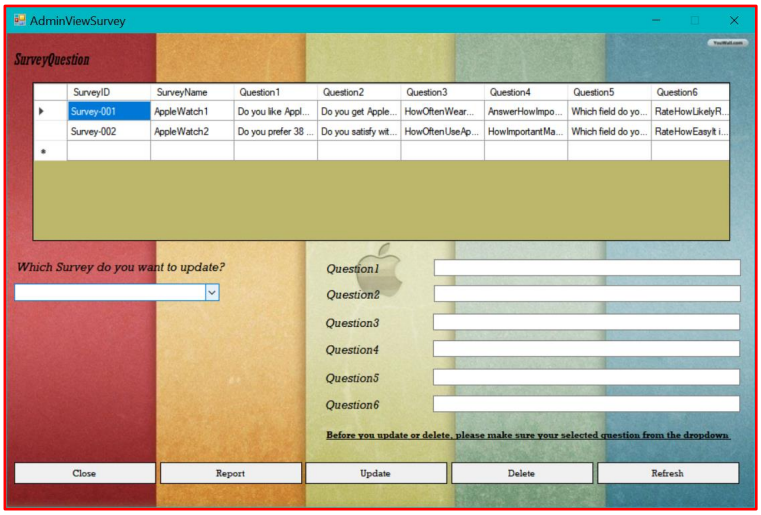


Figure 24

In order to report the survey responses in textual format, the report should be named first. Enter the report name in the textbox which is at the bottom of the page and hit the “Print” button. Then, “successful” will pop up and the tasks of administrators end up in this stage.

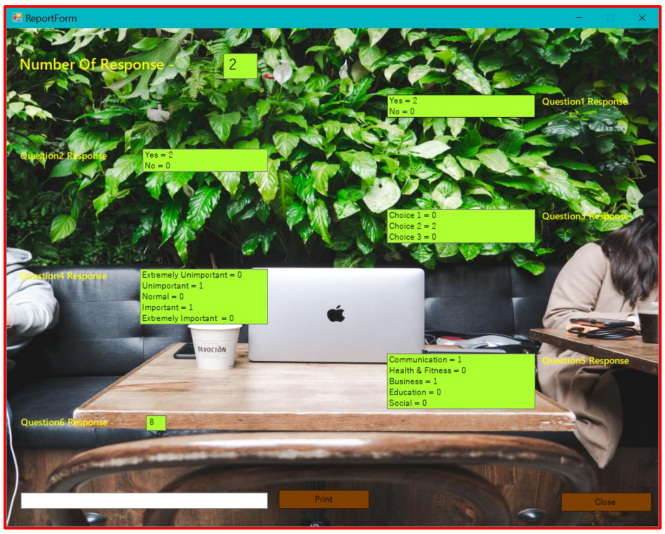


Figure 25

For customers

The customer login page is the initial one for clients to answer surveys through this account. Error handling is constructed to avoid from incomplete fill-in. The underlined label links the page with the registration page. The textboxes receive the input data.

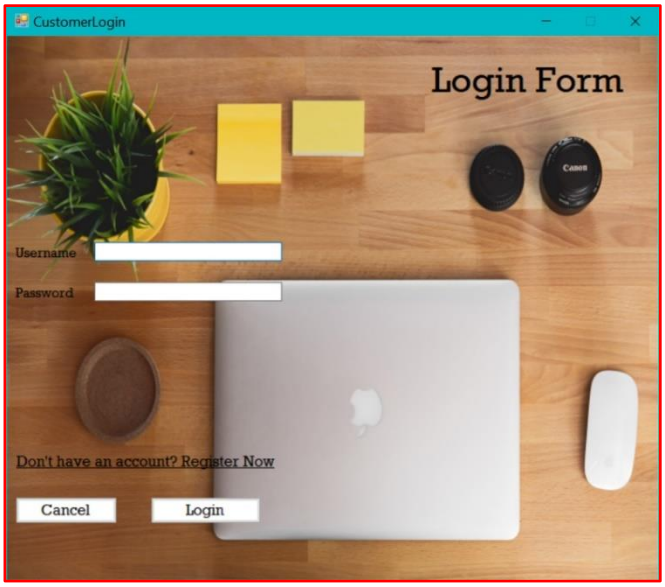


Figure 26

The customer registration page is for accumulating the customers information and give secure accounts to them in order to answer surveys through these accounts. Only after the customers have read the term policy and enabled the checked box, register button enables to work. Registration process will show null reference exception when some pieces of insufficient information are filled up.

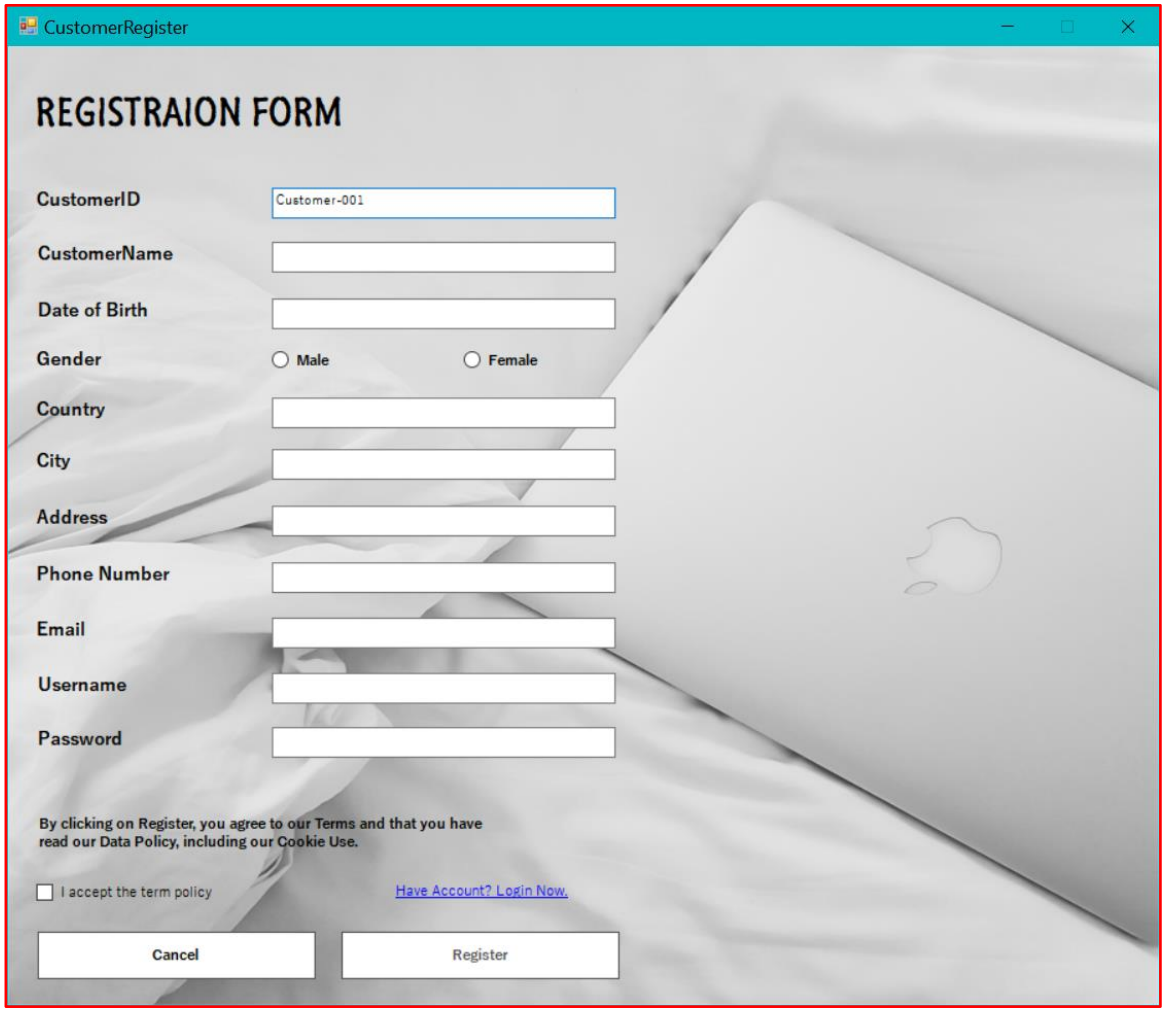


Figure 27

The customer dashboard has an embedded function which shows table and numbers of customers only if the customer clicks on the smart watch. The client can select the desired survey by clicking on the cell in table. And then, the survey ID and admin name of the selected survey will appear on the space. Forgetting to select the survey will not make the program continues. Having selecting, click on the picture beside “Yes” label.

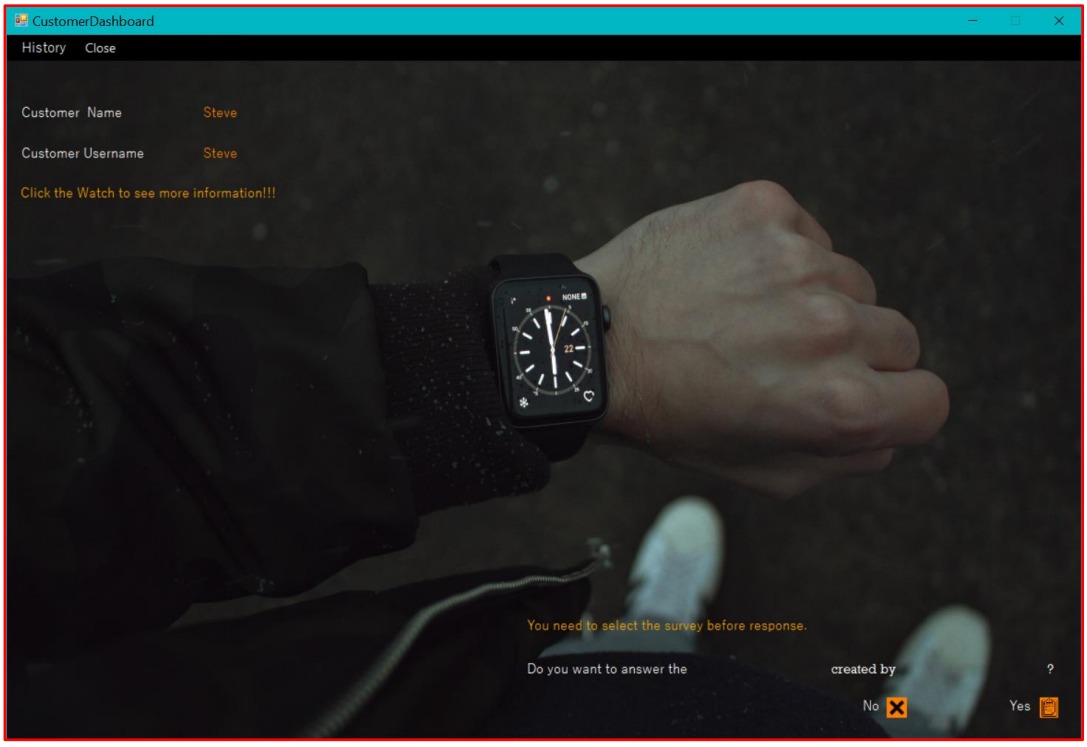


Figure 28 before clicking on the picture

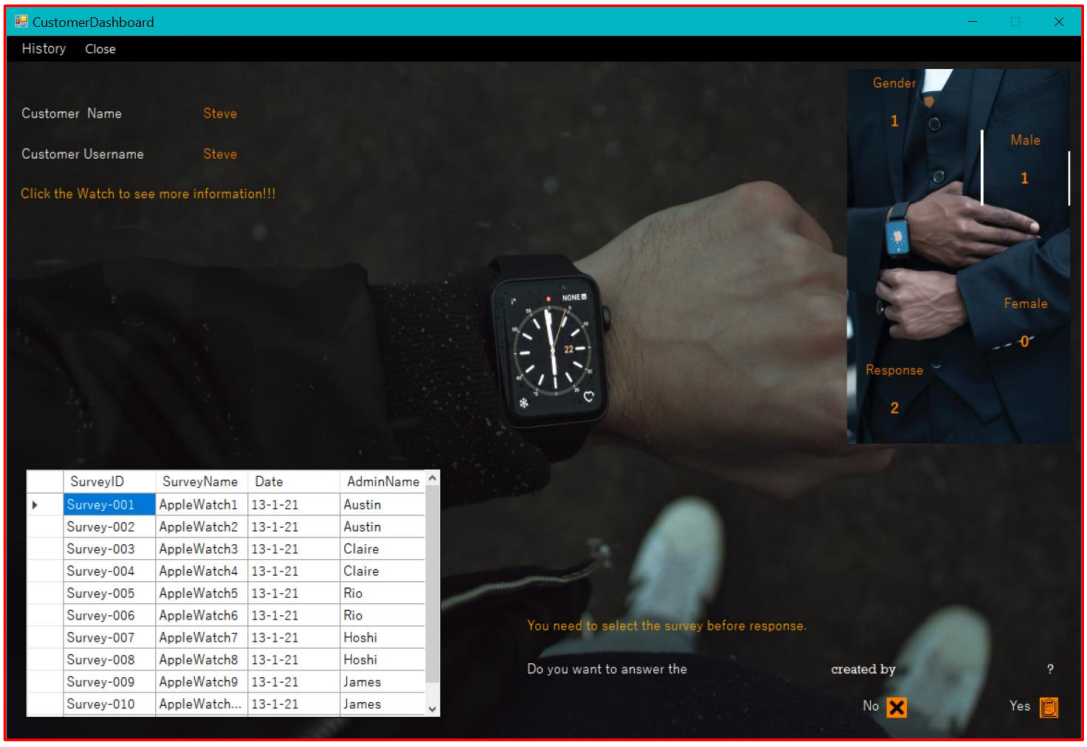


Figure 29 before clicking on the picture

Each survey question is placed within the group box not to confuse with each other. The customer must answer all the six questions otherwise the response won't be saved and the error message will show. After finishing, clicking on "Save" button will make the clients' response save in the database.

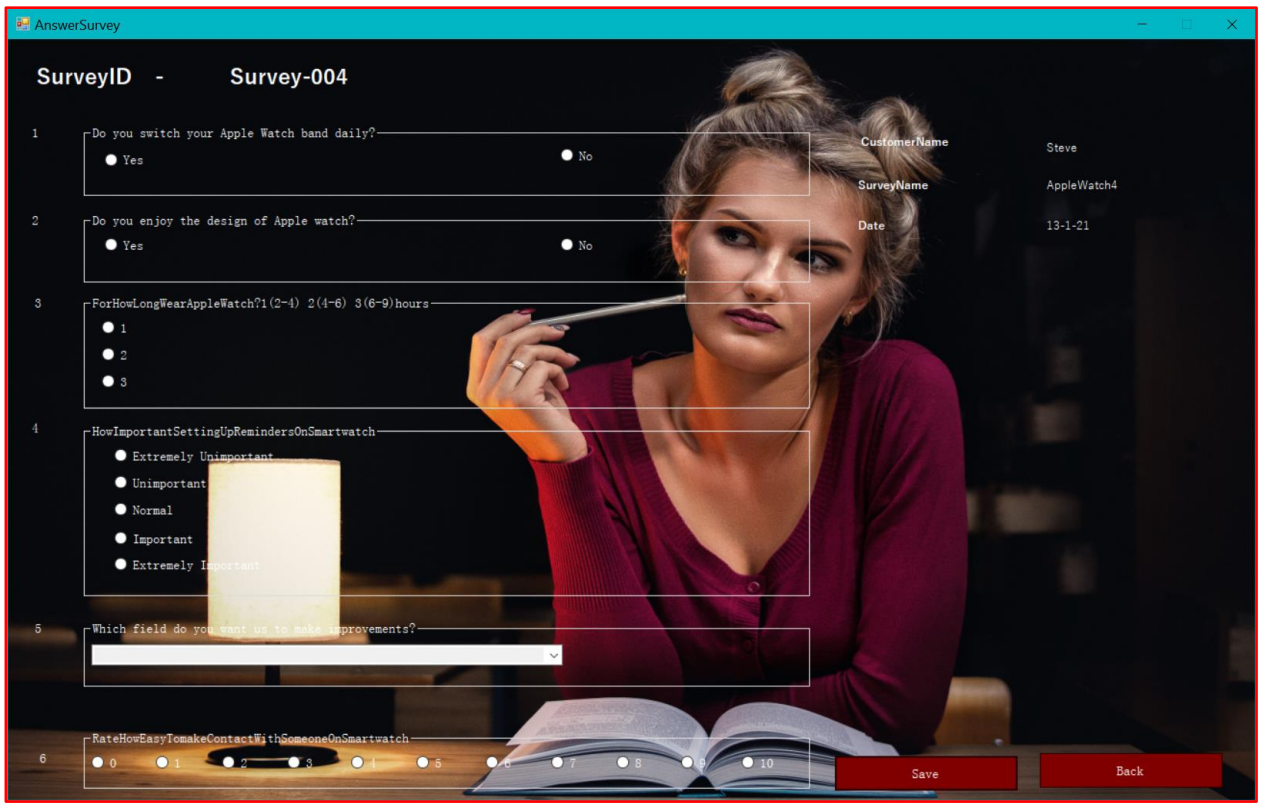


Figure 30

The following page has been created for reviewing for the sake of customers. But the responses won't be shown up as soon as the page is loaded. Clicking on the picture will make response table illustrate below the picture. If the customer is not able to observe his survey, he can utilize the search box which is constructed in order to ease browsing surveys. After selecting the responded surveys, don't forget to hit the "Search" button. To be concluded, the role of customer ends up in here.

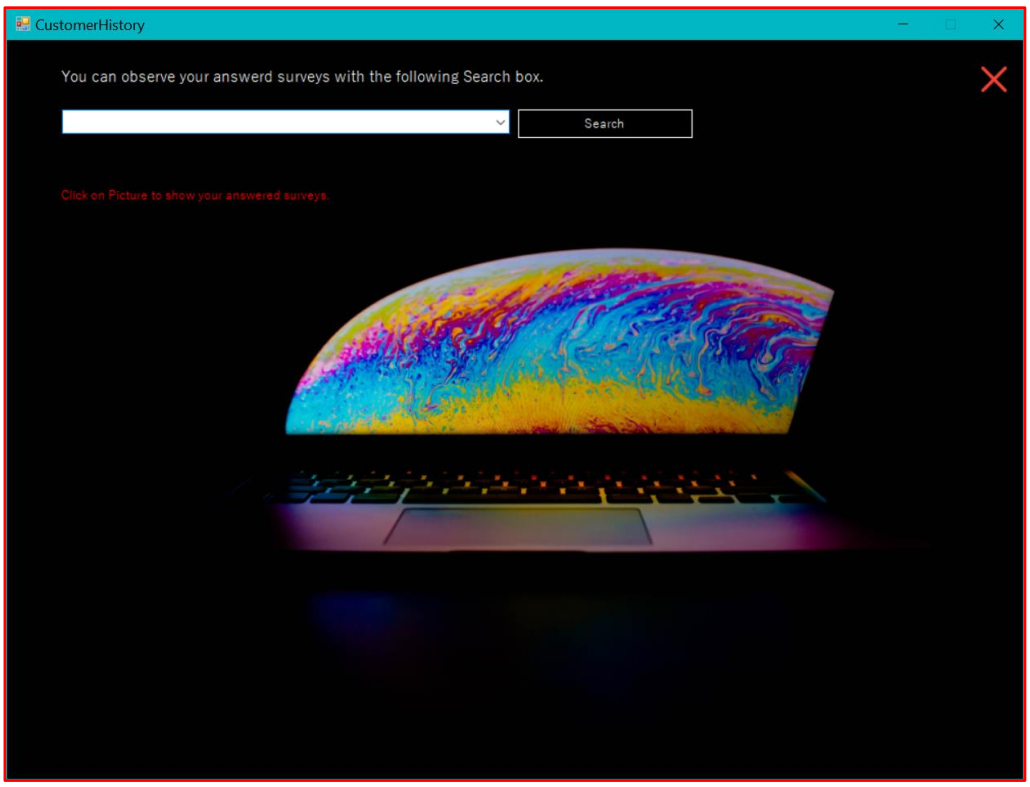


Figure 31 before clicking on the picture

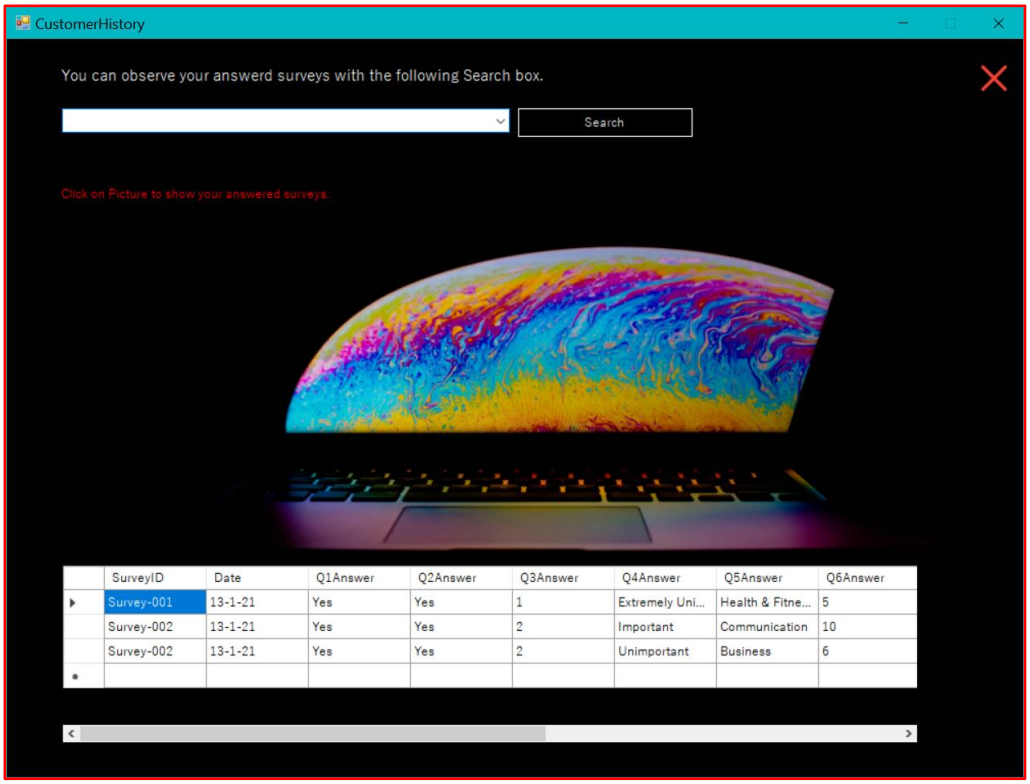


Figure 32 after clicking on the picture

1.6 Quality of Algorithms

The program includes a variety of different algorithm such as functions, iteration, selection, classes, objects, encapsulation, error handling, event handling and inheritance.

Function

```
private void txtGetData()
{
    ai.adminID = txtid.Text;
    ai.adminName = txtadname.Text;
    ai.adminUsername = txtusername.Text;
    ai.adminPassword = txtpassword.Text;
    ai.adminEmail = txtemail.Text;

}
```

Selection

```
if (result == DialogResult.OK)
{
    this.Close();
}
else if (result == DialogResult.Cancel)
{
    this.Show();

}
```

Iteration

```
dgvadminlist.DataSource = adminDataset.GetData();
dgvadminlist.Columns[i].Visible = true;
if (i == 0 || i == 3 )
{
    dgvadminlist.Columns[i].Visible = false;
}

dgvadminlist.Refresh();
```

Class

```
    public partial class WelcomePage : Form
    {
        public WelcomePage()
        {
            InitializeComponent();
        }
        private void picboxadmin_Click(object sender, EventArgs e)
        {
            AdminLogin al = new AdminLogin();
            al.Show();
            this.Hide();
        }
        private void picboxcustomer_Click(object sender, EventArgs e)
        {
            CustomerLogin cl = new CustomerLogin();
            cl.Show();
            this.Hide();
        }
    }
```

Object

```
            AdminLogin al = new AdminLogin();

            al.Show();
```

Inheritance

```
public partial class CustomerLogin : Form
    {

    }
```

Encapsulation

```
        private string SurveyID;
        public string  surveyID
        {
            get { return SurveyID; }
            set { SurveyID = value; }

        }
```

Error handling

```
            try
            {
                //Update according to selected item of Combo Box
                si.surveyID = cboquestions.SelectedValue.ToString();
                surveyDataset.UpdateSurvey(si.question1, si.question2, si.question3,
                si.question4, si.question5, si.question6, si.surveyID);
                dgvSQ.Refresh();
                MessageBox.Show("Update process has been successful","Update
                Complete",MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
                txtDataEmpty();
            }
            catch (NullReferenceException nre)
            {
                MessageBox.Show(nre.Message, "Please Select the survey first",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

Event handling

```
        private void btnrefresh_Click(object sender, EventArgs e)
        {
            this.Hide();
            AdminViewSurvey avs = new AdminViewSurvey();
            avs.Show();
        }
```

1.7 Readability

Meaningful and specific names are used to boost the readability in codebase. In this program, abstract names are not utilized to refrain from poor and unmaintainable codebase.

Variables concerning administrators are written within the class “AdminInformation”.

```
    class AdminInformation
    {
        private static string AdminUsername, AdminPassword, AdminID, AdminName,
AdminEmail, SurveyID;

        public string adminUsername
        {
            get { return AdminUsername; }
            set { AdminUsername = value; }
        }
        public string adminPassword
        {
            get { return AdminPassword; }
            set { AdminPassword = value; }
        }
        public string adminID
        {
            get { return AdminID; }
            set { AdminID = value; }
        }
        public string adminName
        {
            get { return AdminName; }
            set { AdminName = value; }
        }
        public string adminEmail
        {
            get { return AdminEmail; }
            set { AdminEmail = value; }
        }
        public string surveyID
```

```
        {
            get { return SurveyID; }
            set { SurveyID = value; }
        }
```

Variables concerning customers are written within the class “CustomerInformation”.

```
    class CustomerInformation
    {
        private static string CustomerID, CustomerName, Dob, Gender, Country, City,
        Address, Phnumber, Email, Username, Password, SurveyID;
        public string customerID
        {
            get { return CustomerID; }
            set { CustomerID = value; }
        }
        public string customerName
        {
            get { return CustomerName; }
            set { CustomerName = value; }
        }
        public string dob
        {
            get { return Dob; }
            set { Dob = value; }
        }
        public string gender
        {
            get { return Gender; }
            set { Gender = value; }
        }
        public string country
        {
            get { return Country; }
            set { Country = value; }
        }
        public string city
```

```
        {
            get { return City; }
            set { City = value; }
        }
        public string address
        {
            get { return Address; }
            set { Address = value; }
        }
        public string phnumber
        {
            get { return Phnumber; }
            set { Phnumber = value; }
        }
        public string email
        {
            get { return Email; }
            set { Email = value; }
        }
        public string username
        {
            get { return Username; }
            set { Username = value; }
        }
        public string password
        {
            get { return Password; }
            set { Password = value; }
        }
        public string surveyID
        {
            get { return SurveyID; }
            set { SurveyID = value; }
        }
    }
```

Variables concerning surveys are written within the class “SurveyInformation”.

```
    class SurveyInformation
    {
        private string SurveyID, SurveyName, SurveyDate, AdminID, AdminName, Question1,
Question2, Question3, Question4, Question5, Question6;
        public string  surveyID
        {
            get { return SurveyID; }
            set { SurveyID = value; }
        }
        public string surveyName
        {
            get { return SurveyName; }
            set { SurveyName = value; }
        }
        public string surveyDate
        {
            get { return SurveyDate; }
            set { SurveyDate = value; }
        }
        public string adminID
        {
            get { return AdminID; }
            set { AdminID = value; }
        }
        public string adminName
        {
            get { return AdminName; }
            set { AdminName = value; }
        }
        public string question1
        {
            get { return Question1; }
            set { Question1 = value; }
        }
        public string question2
        {
```

```
            get { return Question2; }
            set { Question2 = value; }
        }
        public string question3
        {
            get { return Question3; }
            set { Question3 = value; }
        }
        public string question4
        {
            get { return Question4; }
            set { Question4 = value; }
        }
        public string question5
        {
            get { return Question5; }
            set { Question5 = value; }
        }
        public string question6
        {
            get { return Question6; }
            set { Question6 = value; }
        }
    }
```

2. Task – 2

2.1 Types of Testing

Software Testing is the process of checking whether or not the actual software meets expected requirements. And it also checks the reasonableness of output process. It is implemented during software development. The objective of software testing is to identify errors, weaknesses and missing requirements in contrast to actual requirements. **(What Is Software Testing | Everything You Should Know, 2021), (Types of Software Testing: Different Testing Types with Details, 2021)**

Software testing can be categorized into two parts:

Black Box Testing is a behavioral, specification-based or input-output testing method without caring about the internal structures and logic of the program. It is the outer or external software testing. It is beneficial to the high-level testing and least time consuming. **(Writer, 2021), (Differences between Black Box Testing vs White Box Testing - GeeksforGeeks, 2021)**

White Box Testing is a structural testing method based on the knowledge about the Internal software and code working. It is mostly performed by software developers. It is said to be the inner or the internal software testing. It can be applicable to the low-level testing and most time consuming. **(What Is White Box Testing | Types & Techniques for Code Coverage | Imperva, 2021), (What is WHITE Box Testing? Techniques, Example & Types, 2021)**

In the test plan, Admin Registration page is performed by white-box testing. However, the rest of the program is tested by black-box method.

2.2 Testing Plans

Black box testing

Test script	Description	Date	Tester
1.1	Verify if the registration page will continue when there are some pieces of incomplete information.	Dec 14	Ye Thu Hlaing
1.2	Verify if the registration page will continue when the information is completely filled in.	Dec 15	Ye Thu Hlaing
1.3	Verify if the registration page will continue to the login page by clicking the link-label.	Dec 15	Ye Thu Hlaing
1.4	Verify if registration page will close when the Cancel button is click.	Dec 15	Ye Thu Hlaing
2.1	Verify if login page will continue when the accurate information is filled in.	Dec 16	Ye Thu Hlaing
2.2	Verify if login page will continue when the incorrect username or password is filled in	Dec 16	Ye Thu Hlaing
2.3	Verify if login page will continue when nothing is filled in	Dec 16	Ye Thu Hlaing
2.4	Verify if login page will continue to Registration page after clicking the link label	Dec 17	Ye Thu Hlaing
2.5	Verify if login page will continue after clicking the “Cancel” button	Dec 17	Ye Thu Hlaing
3.1	Verify if administrative home page will offer Admin view list page when clicking on the “View”	Dec 17	Ye Thu Hlaing
3.2	Verify if administrative home page will offer Registration page when clicking on the “Register”.	Dec 18	Ye Thu Hlaing
3.3	Verify if administrative home page will offer Login page when clicking on the “Login”.	Dec 18	Ye Thu Hlaing

3.4	Verify if administrative home page will offer Create Survey page when selecting Survey=>Create Survey.	Dec 19	Ye Thu Hlaing
3.5	Verify if administrative home page will offer View Survey page when selecting Survey=Create Survey.	Dec 19	Ye Thu Hlaing
3.6	Verify if administrative home page will offer View Survey page when selecting Survey=View Survey.	Dec 19	Ye Thu Hlaing
4.1	Verify if the Admin view list page will close	Dec 20	Ye Thu Hlaing
5.1	Verify if survey is created when clicking on the “confirm” button in the Create survey page	Dec 21	Ye Thu Hlaing
5.2	Verify if Create survey page continues when incompletely fill in.	Dec 22	Ye Thu Hlaing
5.3	Verify if Create survey page will close when clicking on the “Cancel” button.	Dec 23	Ye Thu Hlaing
6.1	Verify if items will be dropped when clicking on the arrow of dropdown box.	Dec 24	Ye Thu Hlaing
6.2	Verify if the respective survey questions will appear in the respective textboxes after clicking on the survey ID.	Dec 25	Ye Thu Hlaing
6.3	Verify if the update process will work when clicking on the “Update” button in the “View Survey” Page	Dec 25	Ye Thu Hlaing
6.4	Verify if the delete process will work when clicking on the “Delete” button in the “View Survey” Page	Dec 26	Ye Thu Hlaing
6.5	Verify if the View Survey page will close	Dec 27	Ye Thu Hlaing
6.6	Verify if the View Survey page will offer the Report form page after clicking on “Report” button	Dec 28	Ye Thu Hlaing
7.1	Verify if the welcome page will continue to admin site.	Dec 29	Ye Thu Hlaing

7.2	Verify if the welcome page will continue to customer site.	Dec 29	Ye Thu Hlaing
8.1	Verify if the register button will enable when checkbox is checked	Dec 30	Ye Thu Hlaing
8.2	Verify if registration page will continue when there are some pieces of incomplete information.	Dec 31	Ye Thu Hlaing
8.3	Verify if registration page will continue when the information is completely filled in.	Jan 1	Ye Thu Hlaing
8.4	Verify if registration page will continue to the login page by clicking the link-label.	Jan 2	Ye Thu Hlaing
8.5	Verify if registration page will close when the Cancel button is clicked.	Jan 2	Ye Thu Hlaing
9.1	Verify if login page will continue when the accurate information is filled in.	Jan 3	Ye Thu Hlaing
9.2	Verify if login page will continue when the incorrect username or password is filled in	Jan 4	Ye Thu Hlaing
9.3	Verify if login page will continue when nothing is filled in	Jan 5	Ye Thu Hlaing
9.4	Verify if login page will continue to Registration page after clicking the link label	Jan 6	Ye Thu Hlaing
9.5	Verify if login page will continue after clicking the “Cancel” button	Jan 7	Ye Thu Hlaing
10.1	Verify if other features will show after clicking on the picture	Jan 8	Ye Thu Hlaing
10.2	Verify if the survey ID and Admin Name of selected survey will appear in the blank space by clicking on the table cell	Jan 9	Ye Thu Hlaing
10.3	Verify if the survey ID and Admin Name of selected survey will disappear in blank spaces by clicking on the cross sign	Jan 9	Ye Thu Hlaing
10.4	Verify if the dashboard will continue when nothing has been selected	Jan 10	Ye Thu Hlaing

10.5	Verify if customer dashboard will go to the History page after clicking the “History” button in menu bar	Jan 10	Ye Thu Hlaing
10.6	Verify if the dashboard will close	Jan 11	Ye Thu Hlaing
10.7	Verify if the dashboard will continue to “Customer Answer Survey” page after clicking on picture beside “Yes” label	Jan 12	Ye Thu Hlaing
11.1	Verify if the “Customer Answer Survey” page will transfer responses to databases by clicking “Save” button	Jan 13	Ye Thu Hlaing
11.2	Verify if the “Customer Answer Survey” page will close	Jan 13	Ye Thu Hlaing
12.1	Verify if the responses table will show when clicking on the picture	Jan 14	Ye Thu Hlaing
12.2	Verify if only selected survey will show in the table after hitting the “Search” button	Jan 14	Ye Thu Hlaing
12.3	Verify if the whole page closes after clicking on the red cross	Jan 15	Ye Thu Hlaing
13.1	Verify if the report will save as txt file format after naming the report and clicking on the “Print”	Jan 15	Ye Thu Hlaing
13.2	Verify if the “Report form” page will close	Jan 16	Ye Thu Hlaing

White box testing

Test script	Description	Date	Tester
14.3	White box testing of “Admin Registration” page	Jan 16	Ye Thu Hlaing

2.3 Test Scripts

Black box testing

Test Case	Description	Test Procedure	Expected Result	Actual results
1.1	Testing the registration process	Click on the "Register" with incomplete information	Should not allow to continue when the information is not completely filled in	The "Registration successful" dialog box appears and give back the registration form again.

Before testing

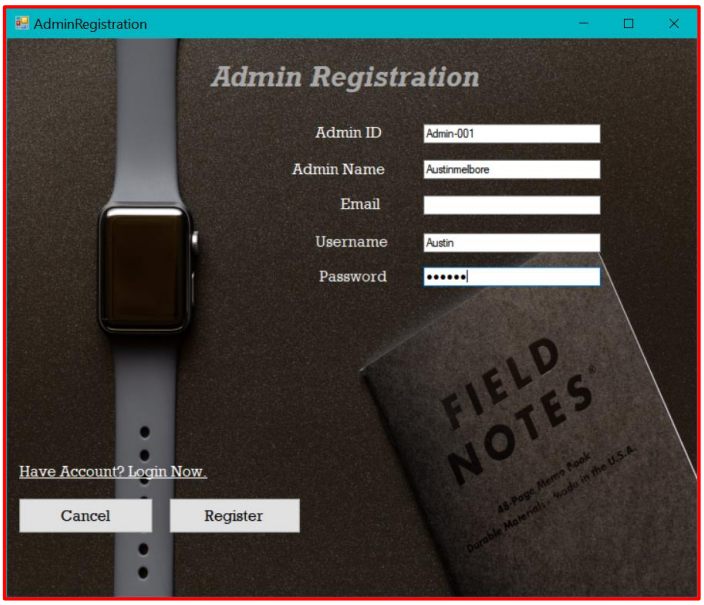


Figure 33

After testing

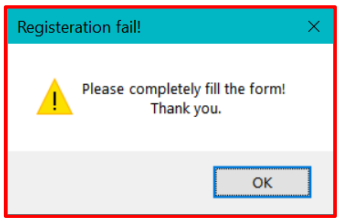


Figure 34

Test Case	Description	Test Procedure	Expected Result	Actual results
1.2	Testing the registration process	Enter valid information, then click on the "Register" button.	Registered data should be saved in the Database and dialog box should also be shown for confirmation	For client side, the successful dialog box appears and returns back to the Login Page when clicking on "OK". For sever side, new account must has been created.

Before testing

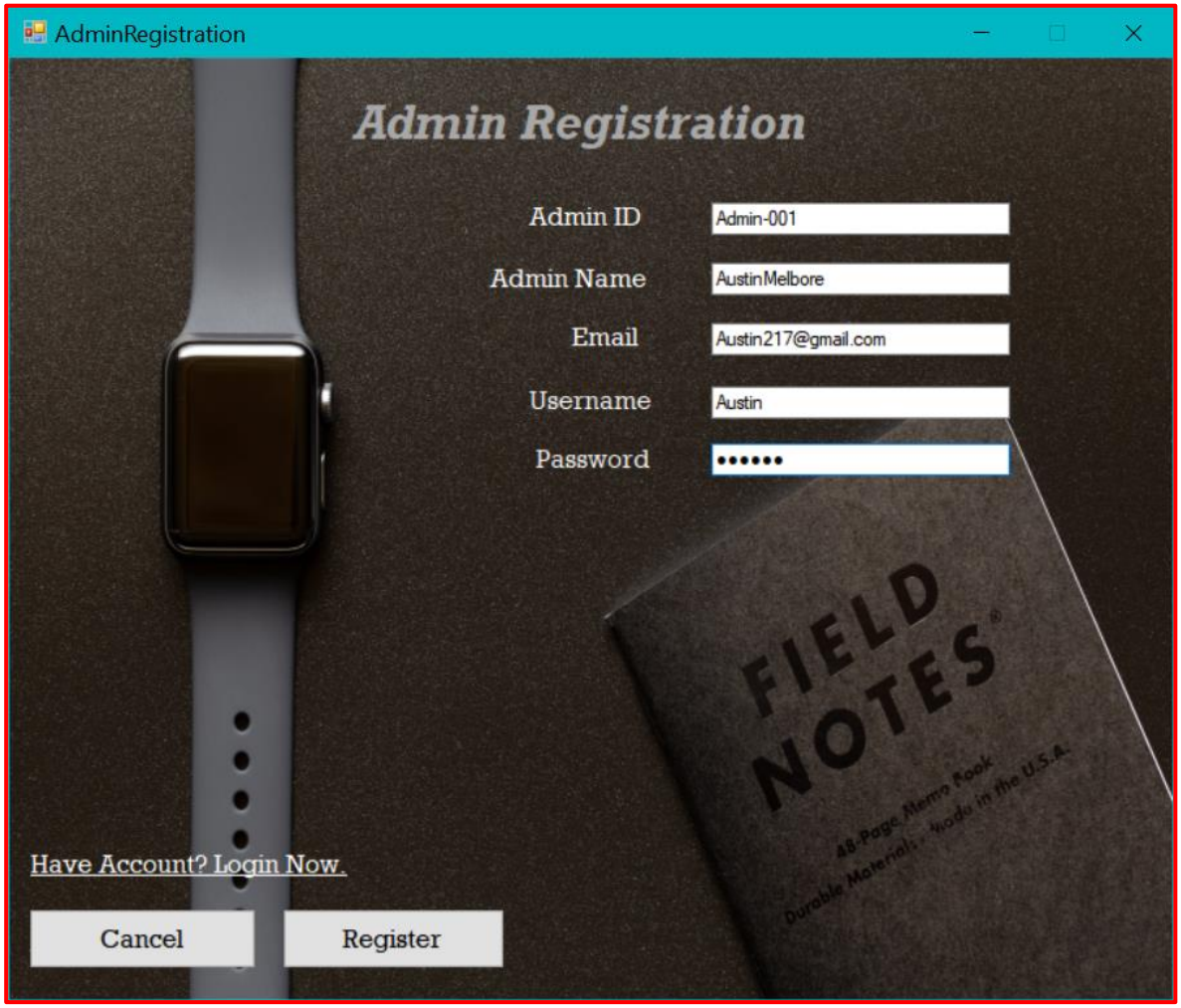


Figure 35 User interface before creating account

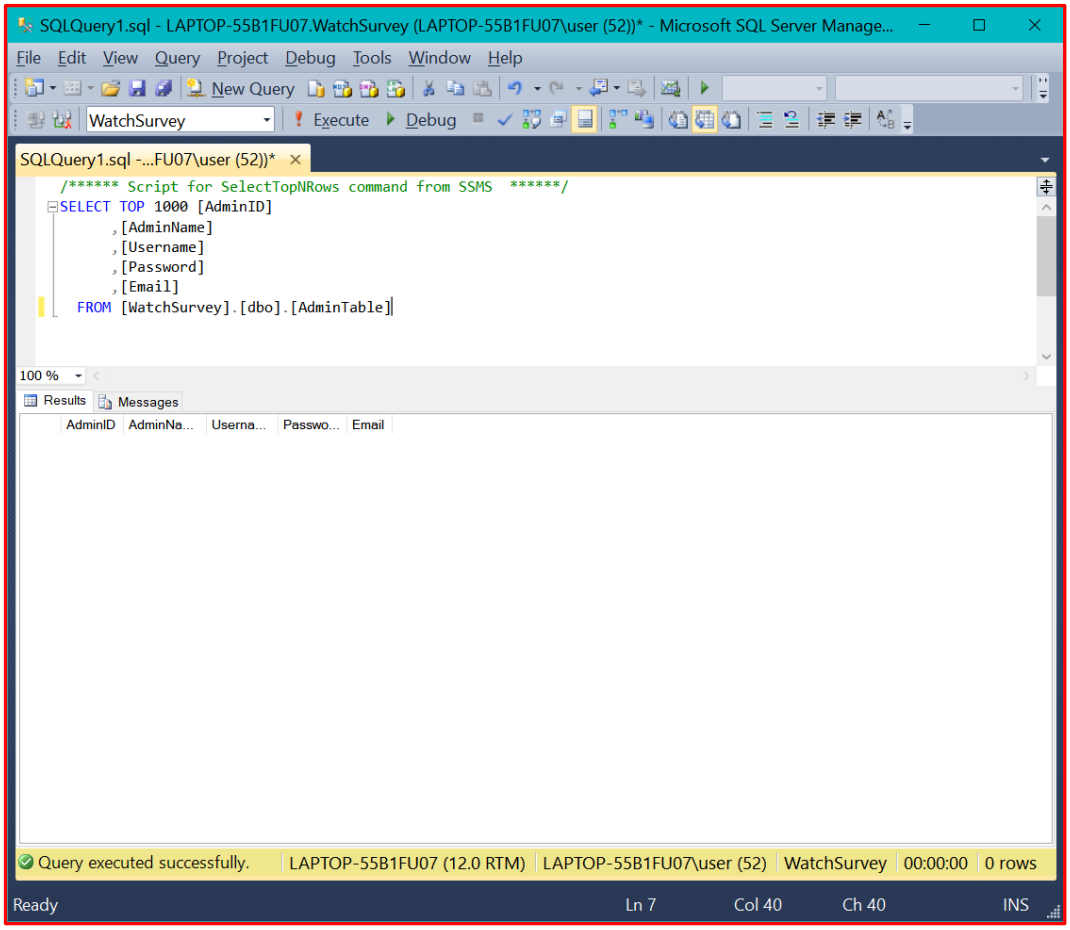


Figure 36 Database before creating admin account

After testing

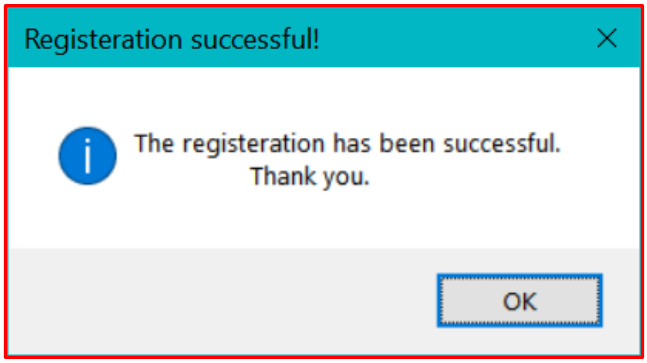


Figure 37 User interface after creating account

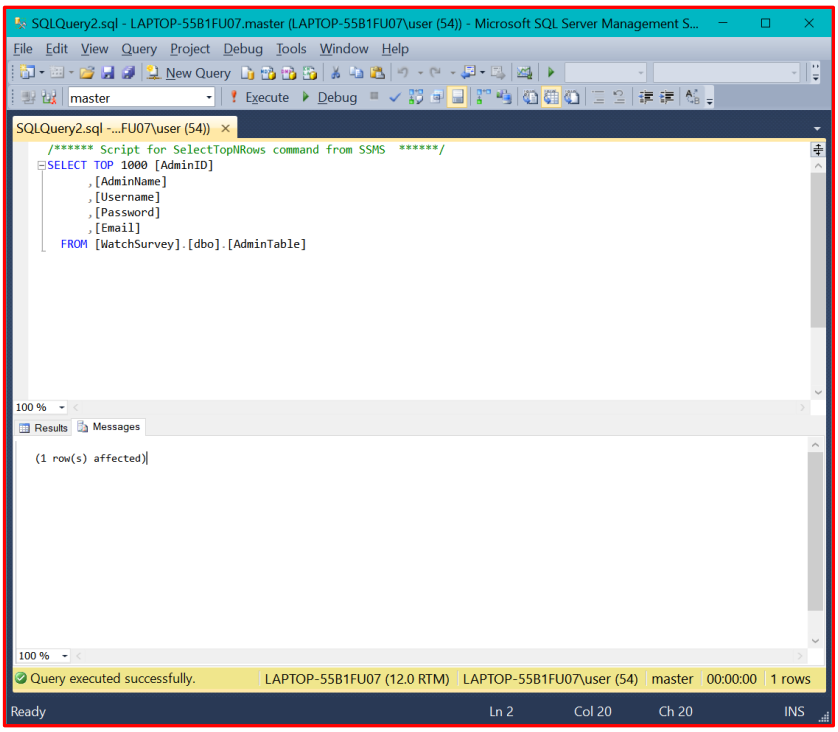


Figure 38 message of successful process in Database after creating admin account

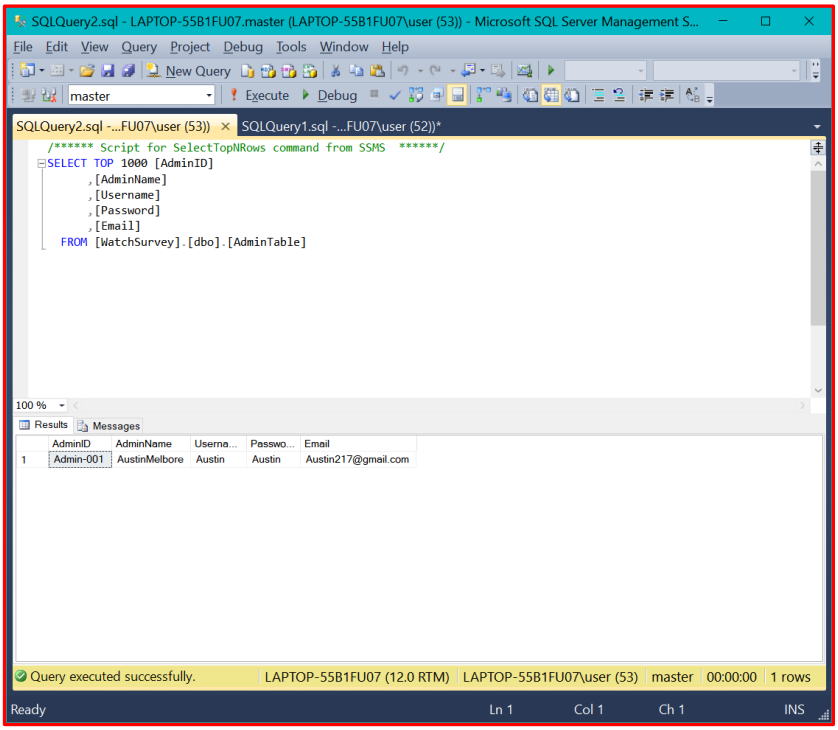


Figure 39 Result of successful process in Database after creating admin account

Test Case	Description	Test Procedure	Expected Result	Actual results
1.3	Testing the white-colored label	Click on the white-colored link label	Able to go to the Login Page	The login page appears.

Before testing

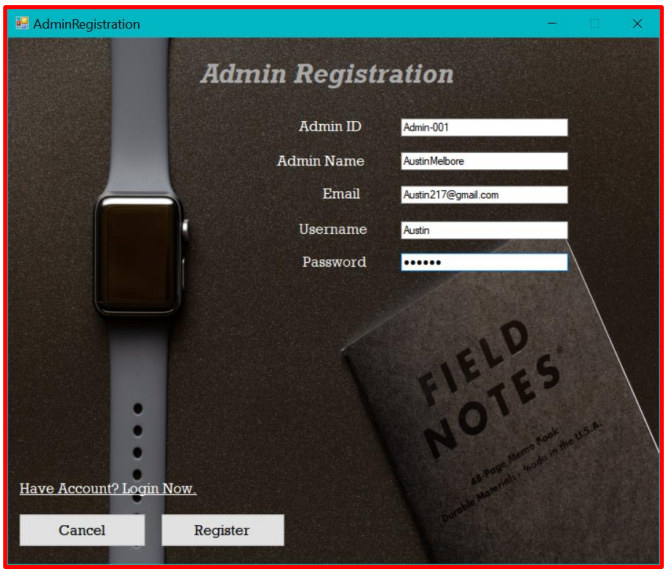


Figure 40

After testing

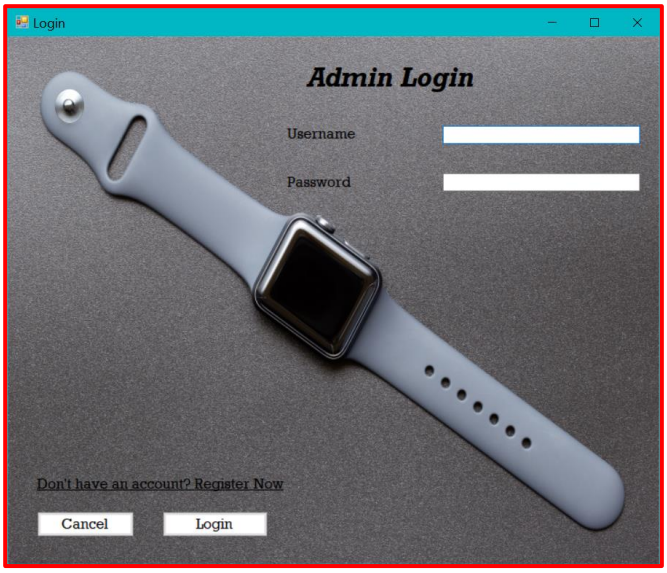


Figure 41

Test Case	Description	Test Procedure	Expected Result	Actual results
1.4	Testing the close button	Click on the “Cancel” button.	Able to close the program	Confirmation box shows to close the program

Before testing

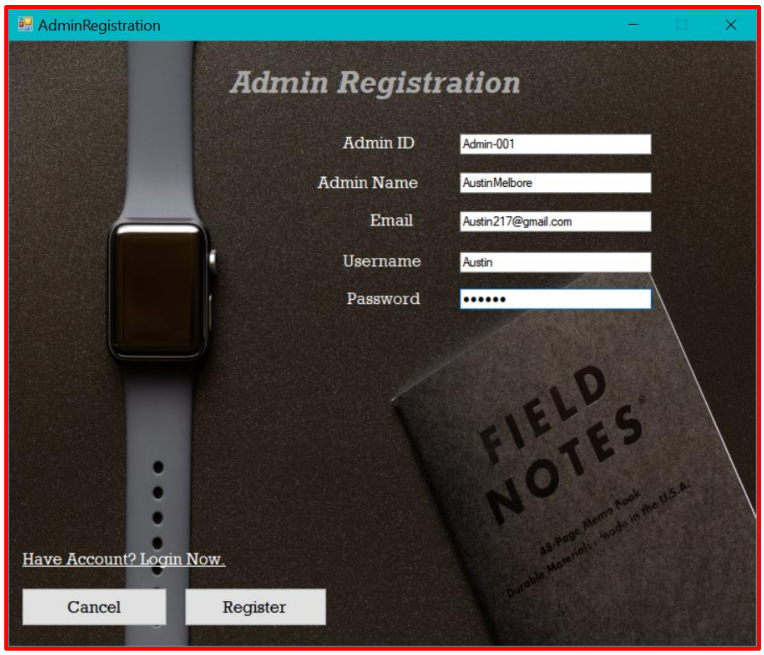


Figure 42

After testing

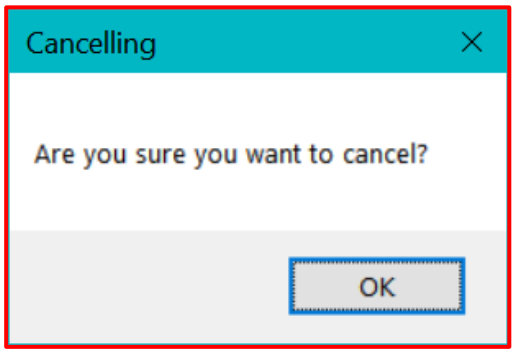


Figure 43

Test Case	Description	Test Procedure	Expected Result	Actual results
2.1	Testing the login	Fill the username and password and then click on the “Login” button.	Should allow login for correct information	“Login successful” dialog box appears and admin Dashboard page has to come up.

Before testing

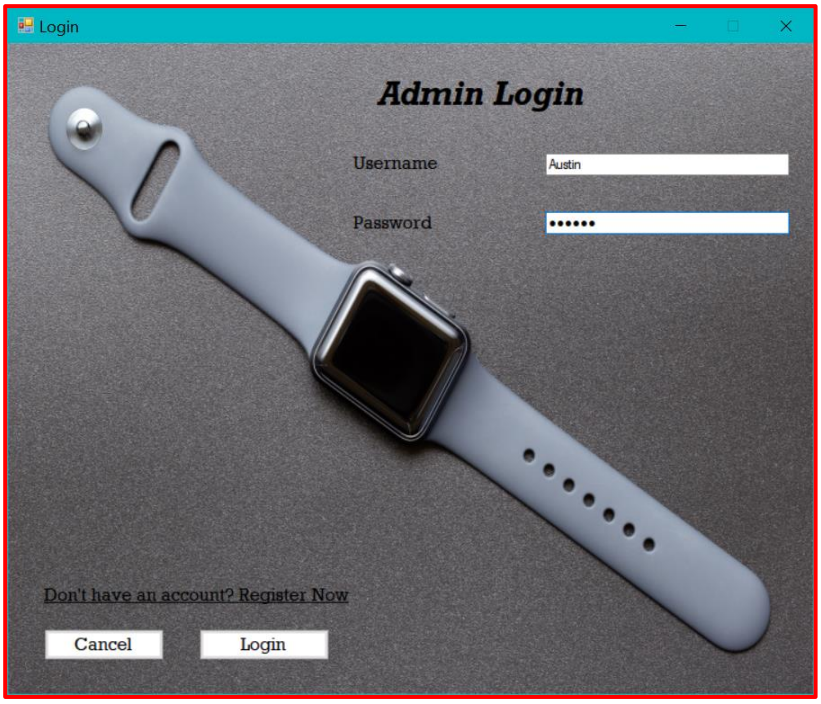


Figure 44

After testing

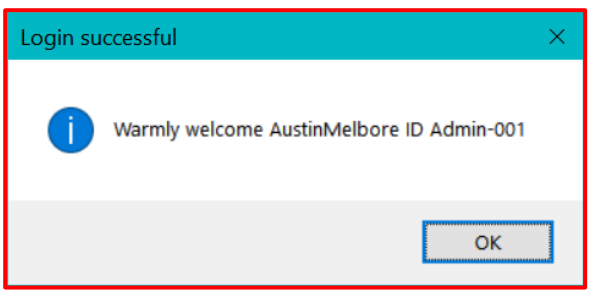


Figure 45

Test Case	Description	Test Procedure	Expected Result	Actual results
2.2	Testing the invalid login	Fill in the blank and enter Login.	Should not allow Login with invalid username or password	“Login fail” dialog box appears.

Before testing

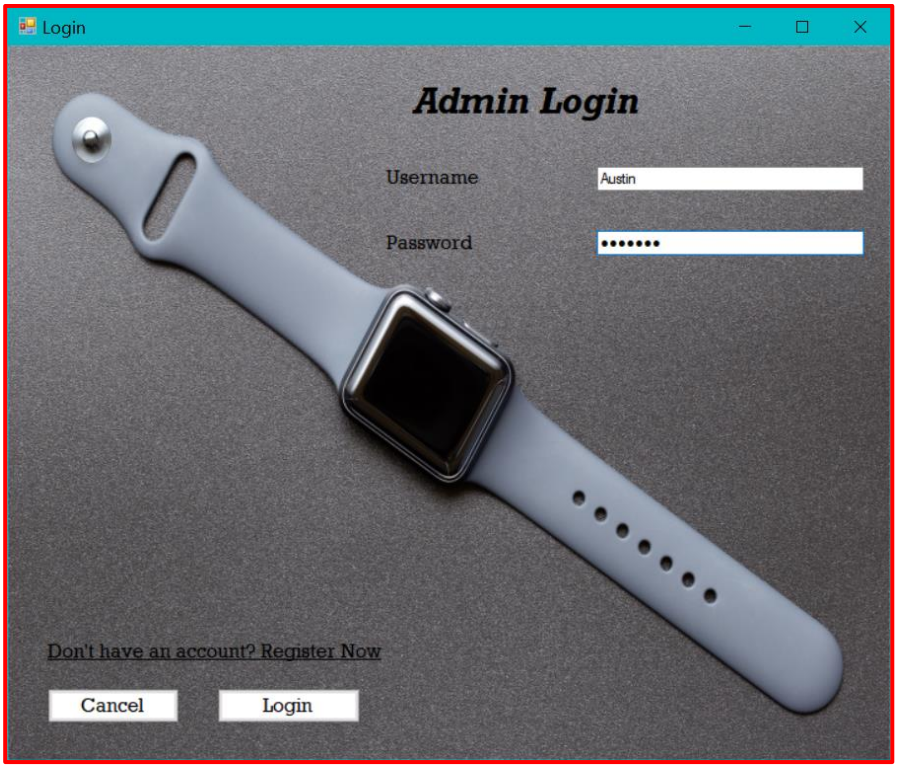


Figure 46

After testing

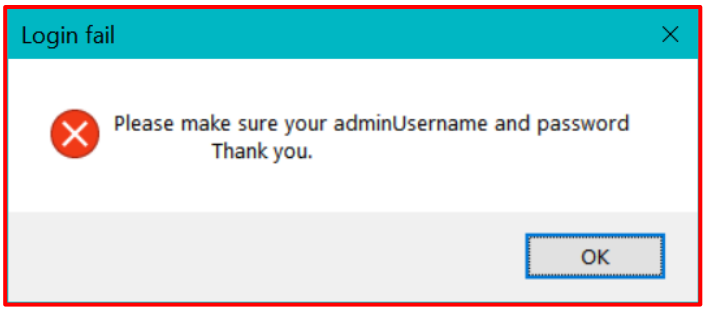


Figure 47

Test Case	Description	Test Procedure	Expected Result	Actual results
2.3	Testing the invalid login	Incomplete form fill-in	Should not allow Login with incomplete information	“Login Incomplete” dialog box appears.

Before testing

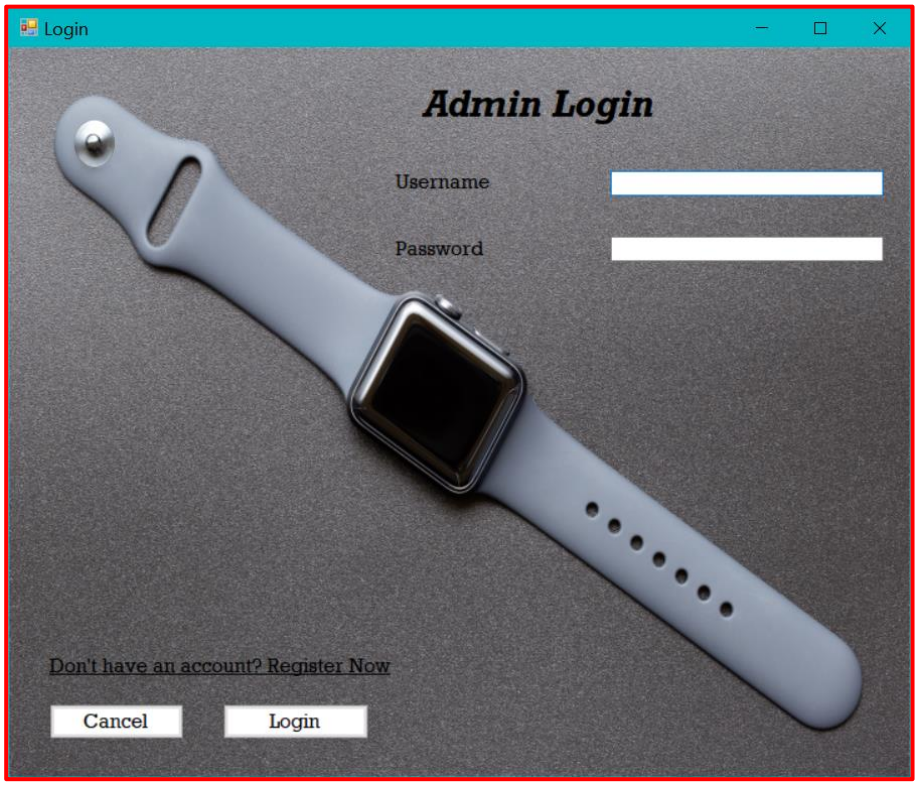


Figure 48

After testing

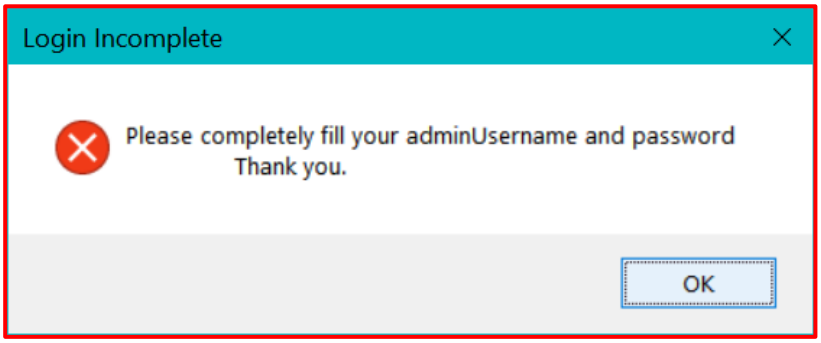


Figure 49

Test Case	Description	Test Procedure	Expected Result	Actual results
2.4	Testing the link label of returning back to the registration page	Click on the “Don’t have an account? Register Now” link Label.	Able to enter the Registration page.	Registration page appears.

Before testing

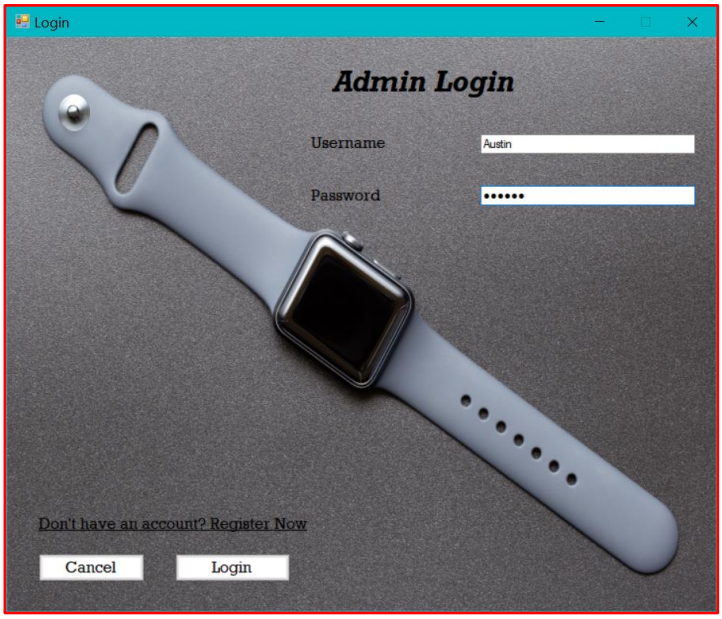


Figure 50

After testing

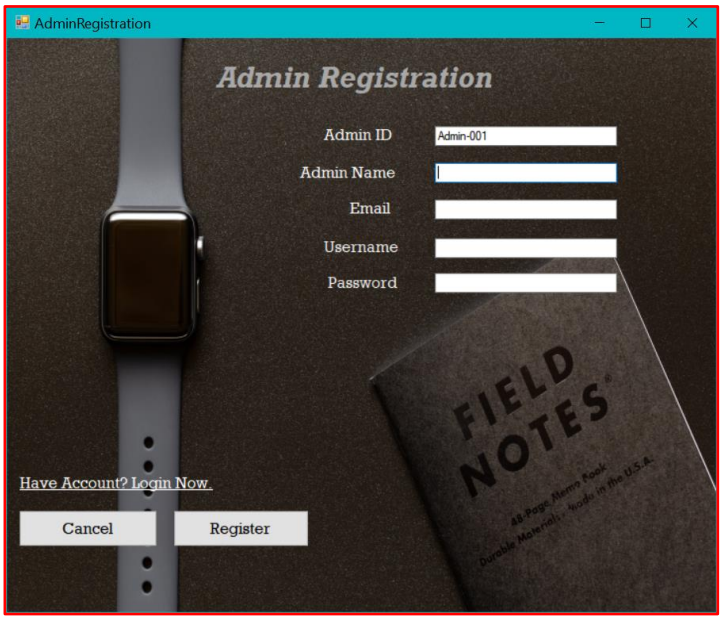


Figure 51

Test Case	Description	Test Procedure	Expected Result	Actual results
2.5	Testing the cancel button	Click on the Cancel button.	Able to close the entire program	Cancelling box appear and the program closes.

Before testing

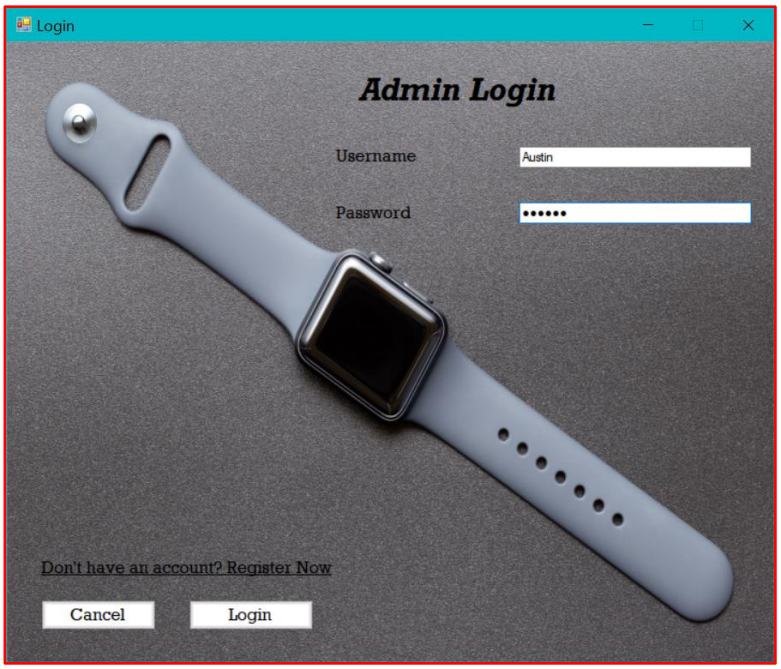


Figure 52

After testing

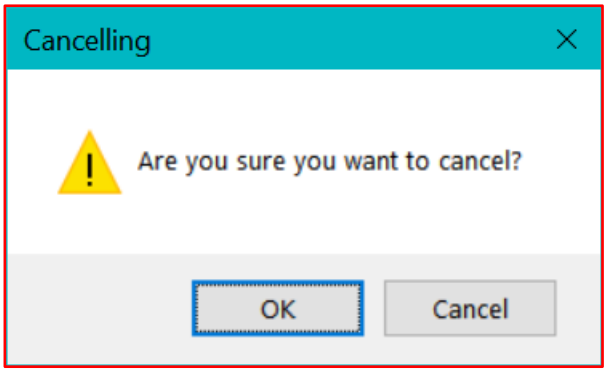


Figure 53

Test Case	Description	Test Procedure	Expected Result	Actual results
3.1	Testing the “View” in the menu bar	Click on the “View” in the menu bar.	Able to jump on the Admin view list page	Admin list page materializes.

Before testing

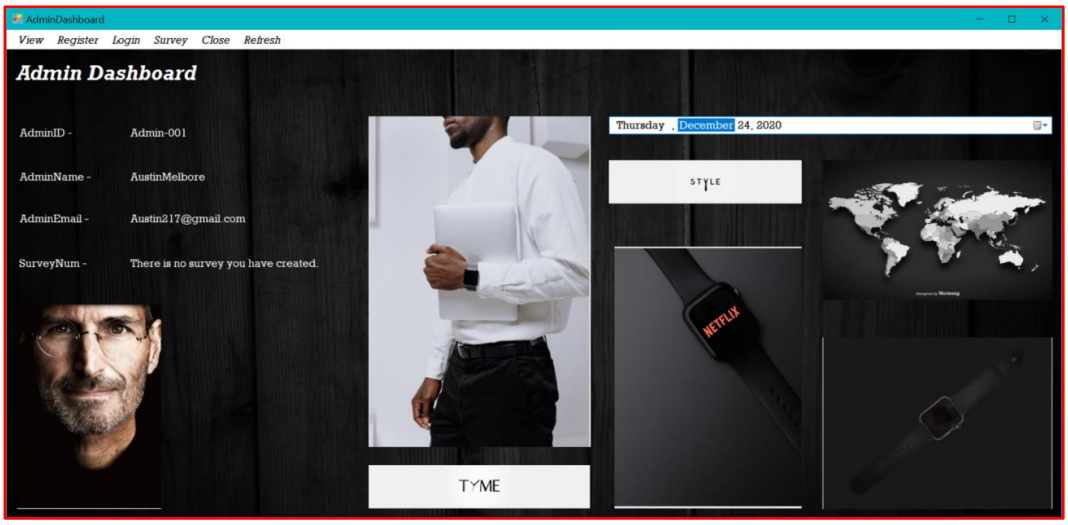


Figure 54

After testing

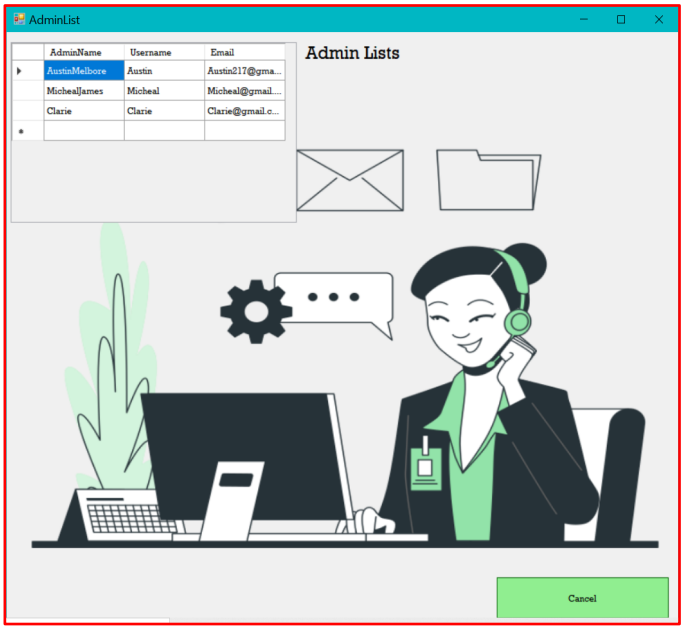


Figure 55

Test Case	Description	Test Procedure	Expected Result	Actual results
3.2	Testing the “Register” button	Click on the “Register” in the menu bar.	Able to go to the registration page	Registration page appears.

Before testing

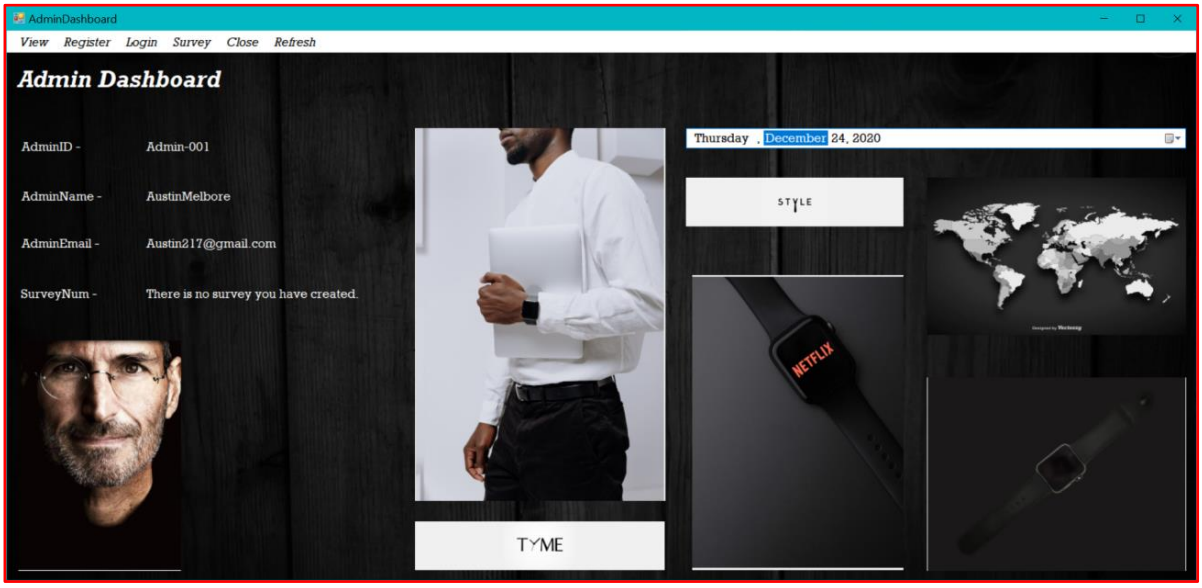


Figure 56

After testing

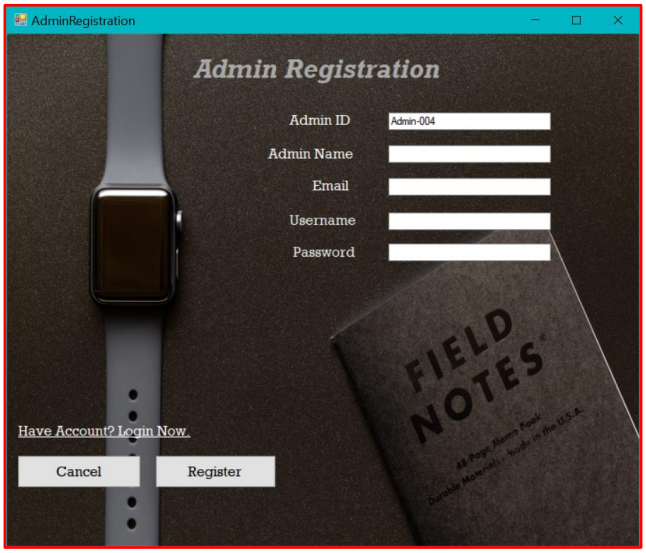


Figure 57

Test Case	Description	Test Procedure	Expected Result	Actual results
3.3	Testing the “Login” button	Click on the “Login” in the menu bar.	Able to go to the login page	Login page must appear.

Before testing

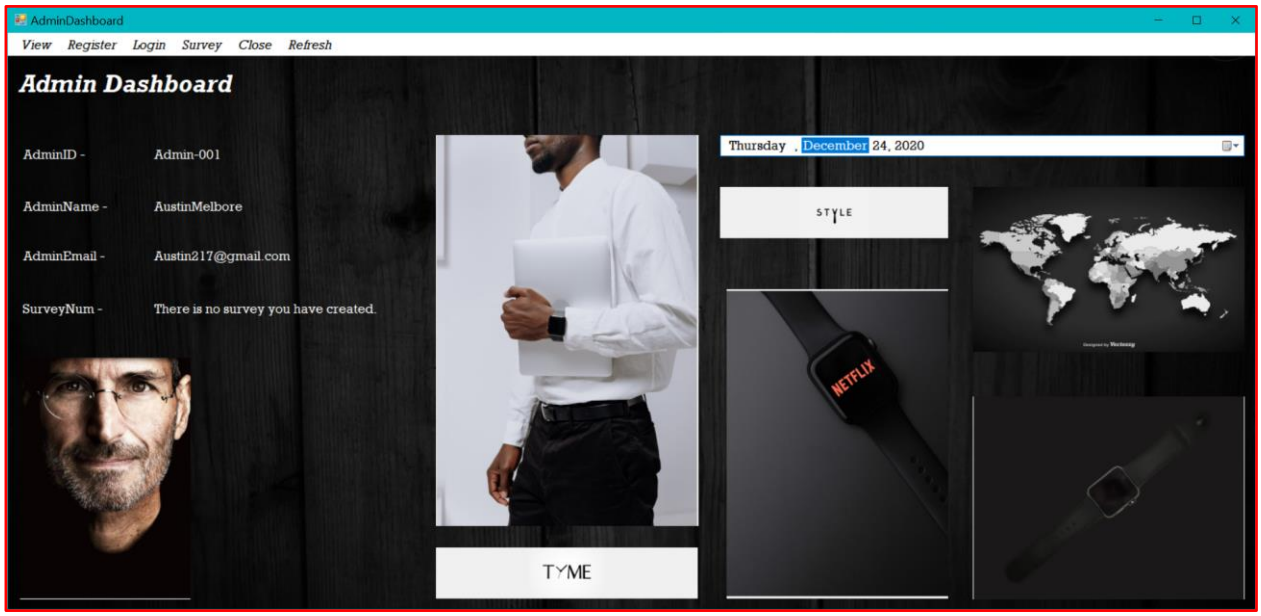


Figure 58

After testing

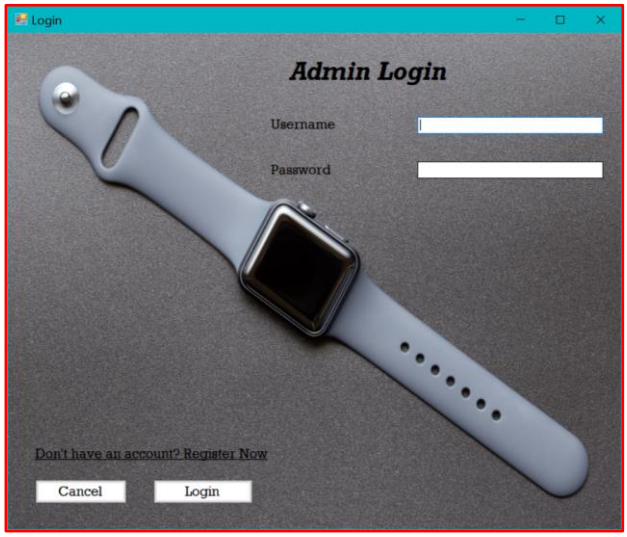


Figure 59

Test Case	Description	Test Procedure	Expected Result	Actual results
3.5	Testing the “Create Survey” button	Click on the “Survey” in the menu bar and select the Create Survey	Should be able to go to the Admin Create Survey page	Admin Create Survey Page appears.

Before testing

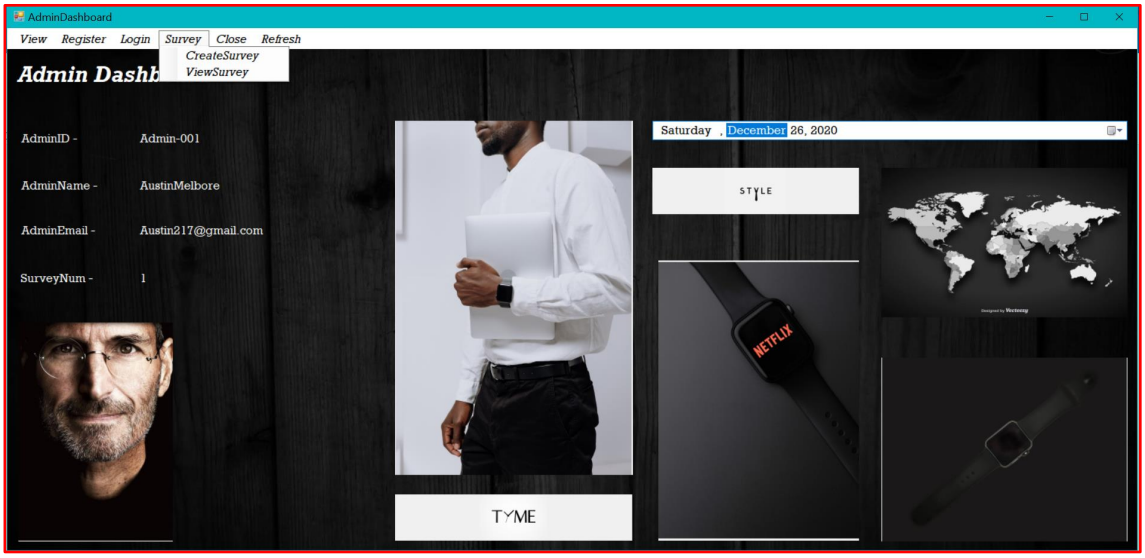


Figure 60

After testing

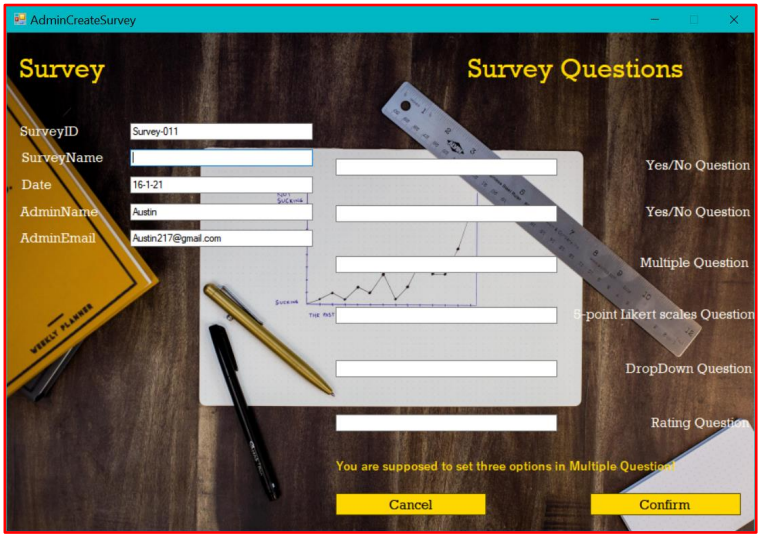


Figure 61

Test Case	Description	Test Procedure	Expected Result	Actual results
3.6	Testing the "View Survey" button.	Click on the "Survey" in the menu bar and select the Survey Create.	Able to go to the Admin View Survey page.	Admin View Survey page appears.

Before testing

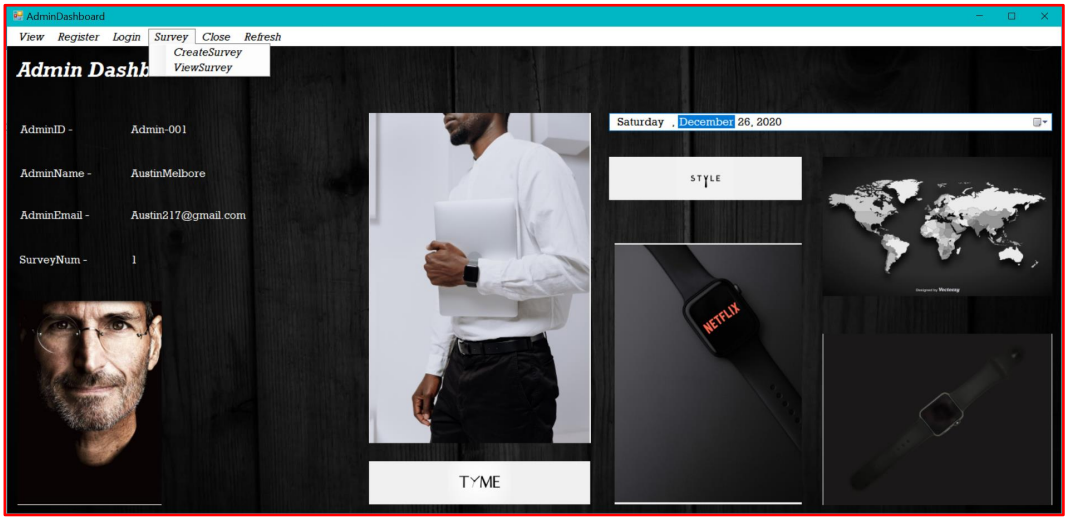


Figure 62

After testing

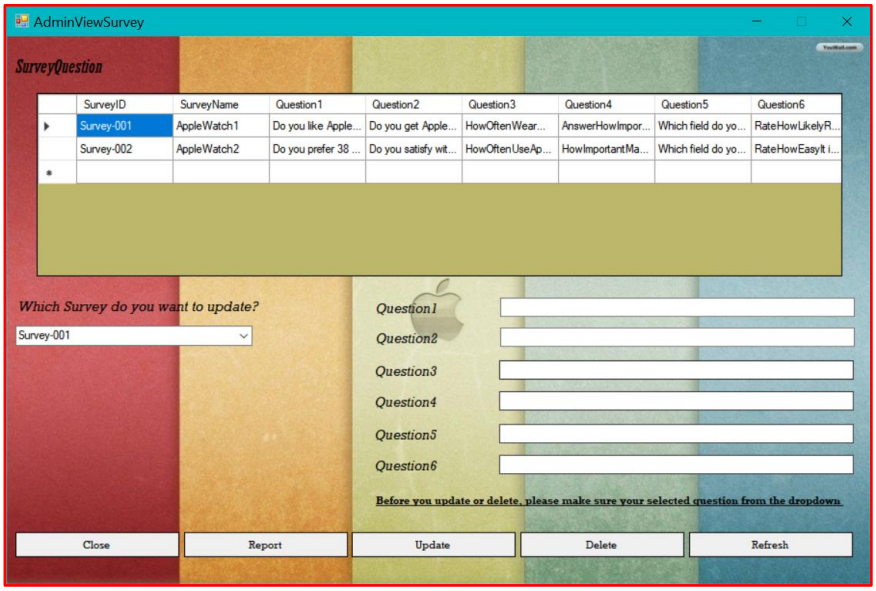


Figure 63

Test Case	Description	Test Procedure	Expected Result	Actual results
4.1	Testing the “Close” button	Click on the “Close”	Able to close the page	The Admin View List page closes and will return back to the admin dashboard

Before testing

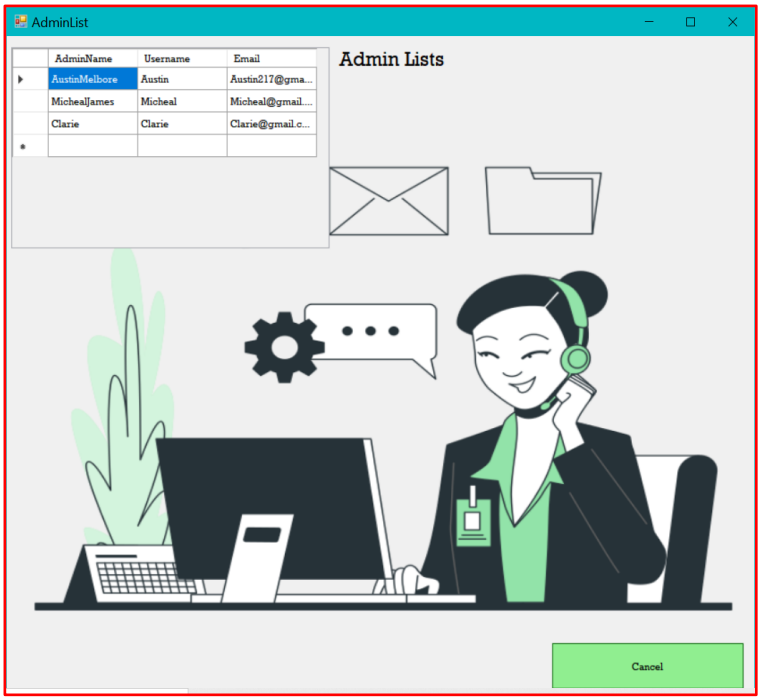


Figure 64

After testing

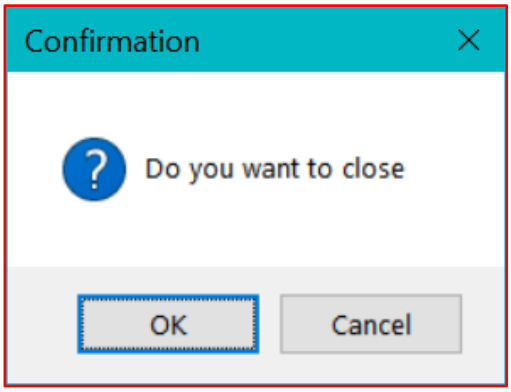


Figure 65

Test Case	Description	Test Procedure	Expected Result	Actual results
5.1	Testing for creating survey process	Click on the “Confirm” button.	Able to save the survey form	“Confirmation” dialog box appears to check out

Before testing

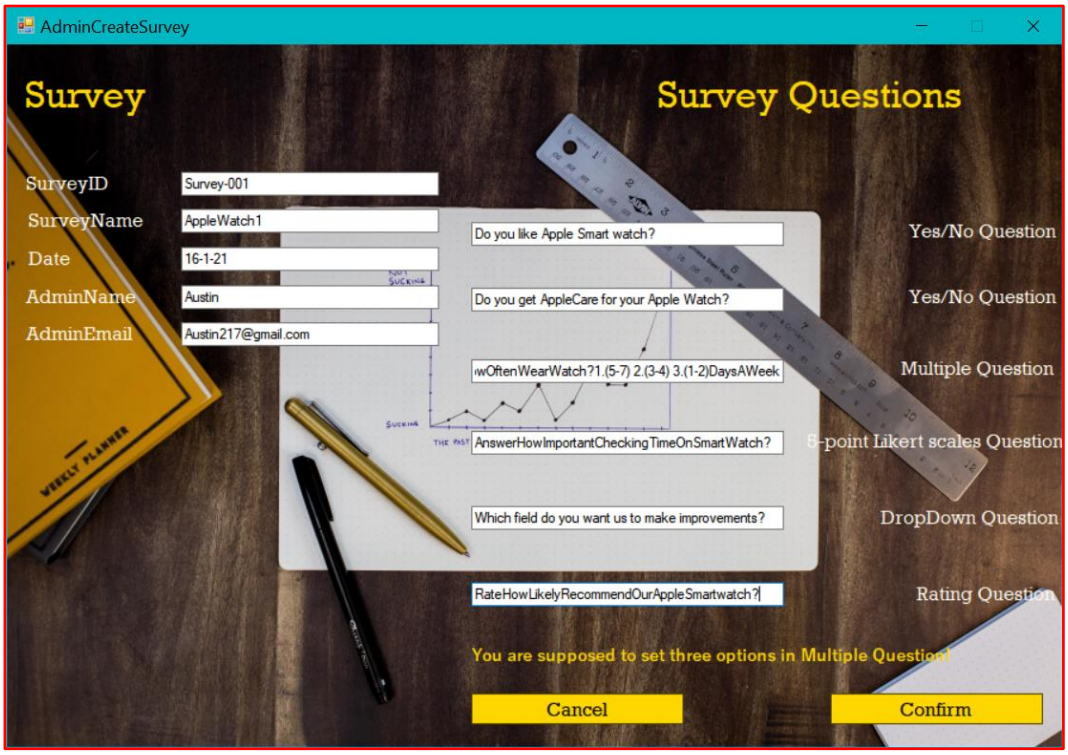


Figure 66

After testing

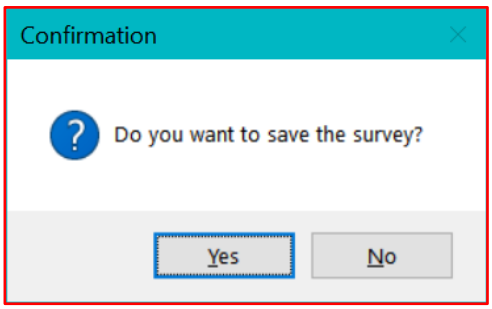


Figure 67 Confirmation box to continue

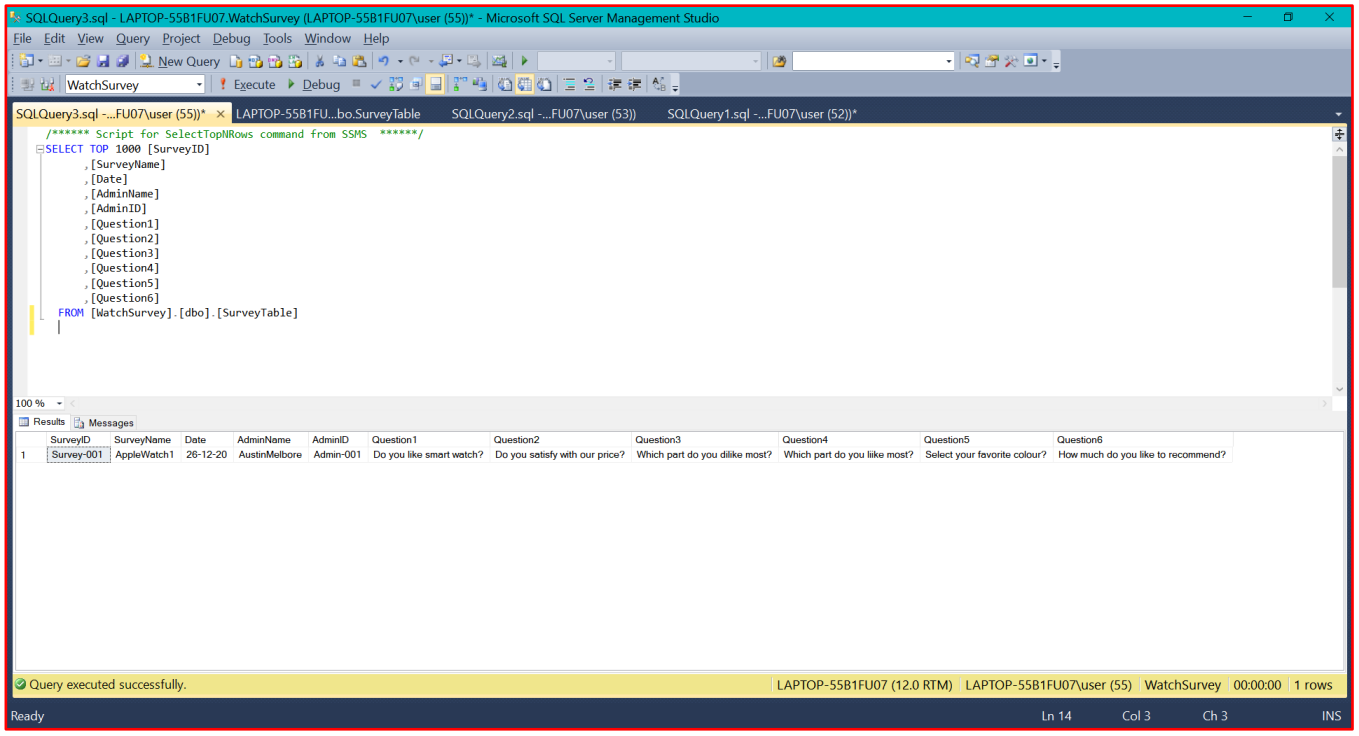


Figure 68 result in Database after saving

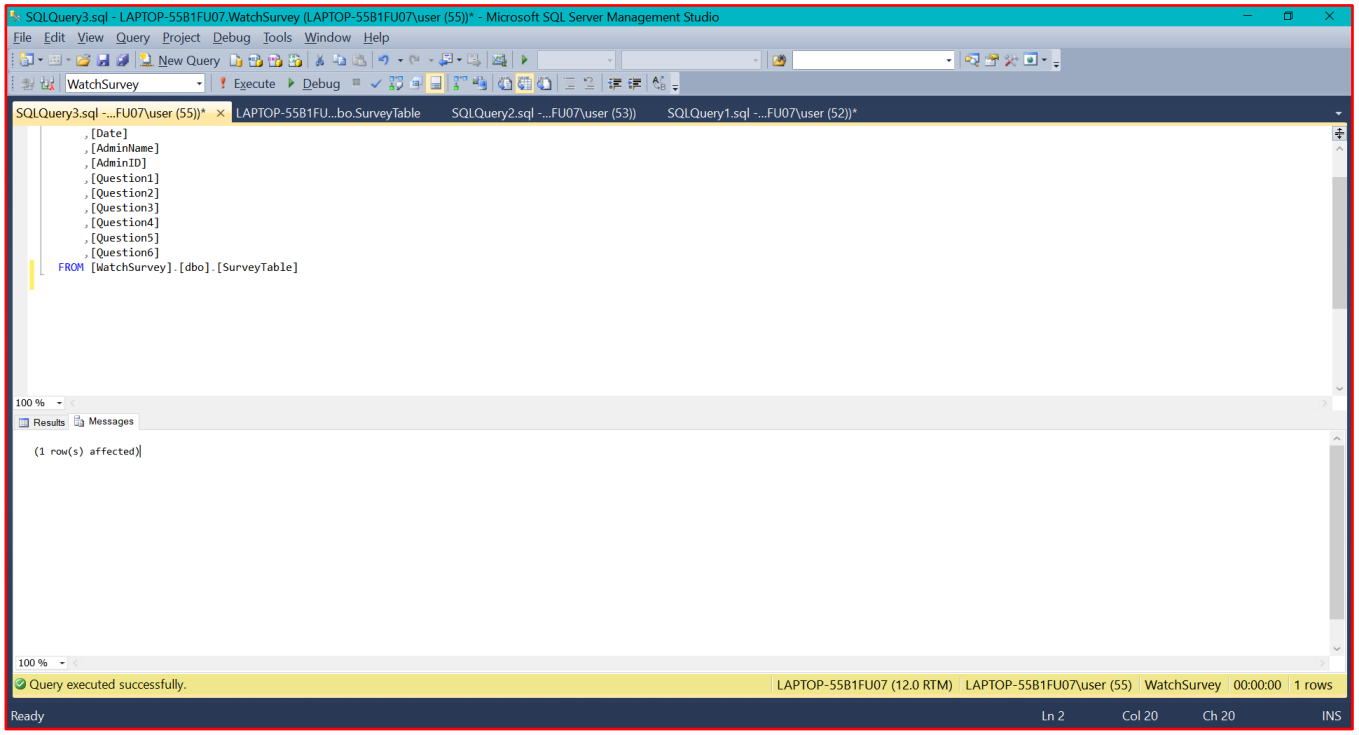


Figure 69 message in Database after saving

Test Case	Description	Test Procedure	Expected Result	Actual results
5.2	Testing for creating survey process with incomplete fill-in	Filling in survey form inadequately	Should not allow incomplete survey	“Survey fail” dialog box pops up, the current page closes and returns back to the home page

Before testing

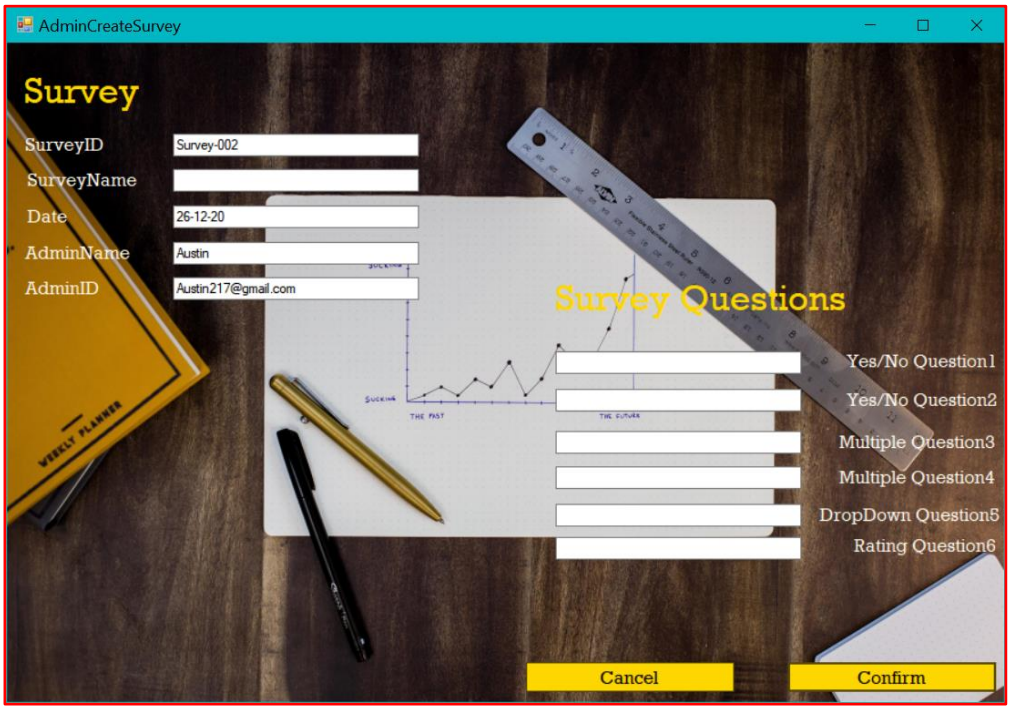


Figure 70

After testing

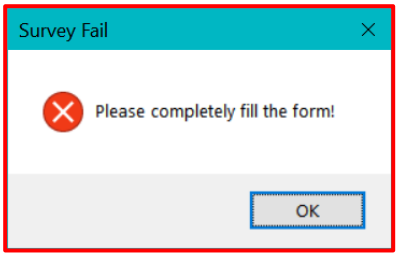


Figure 71

Test Case	Description	Test Procedure	Expected Result	Actual results
5.3	Testing the “Close” button	Click on the “Cancel” button	Able to close the page	The page closes and returns back to the home page

Before testing

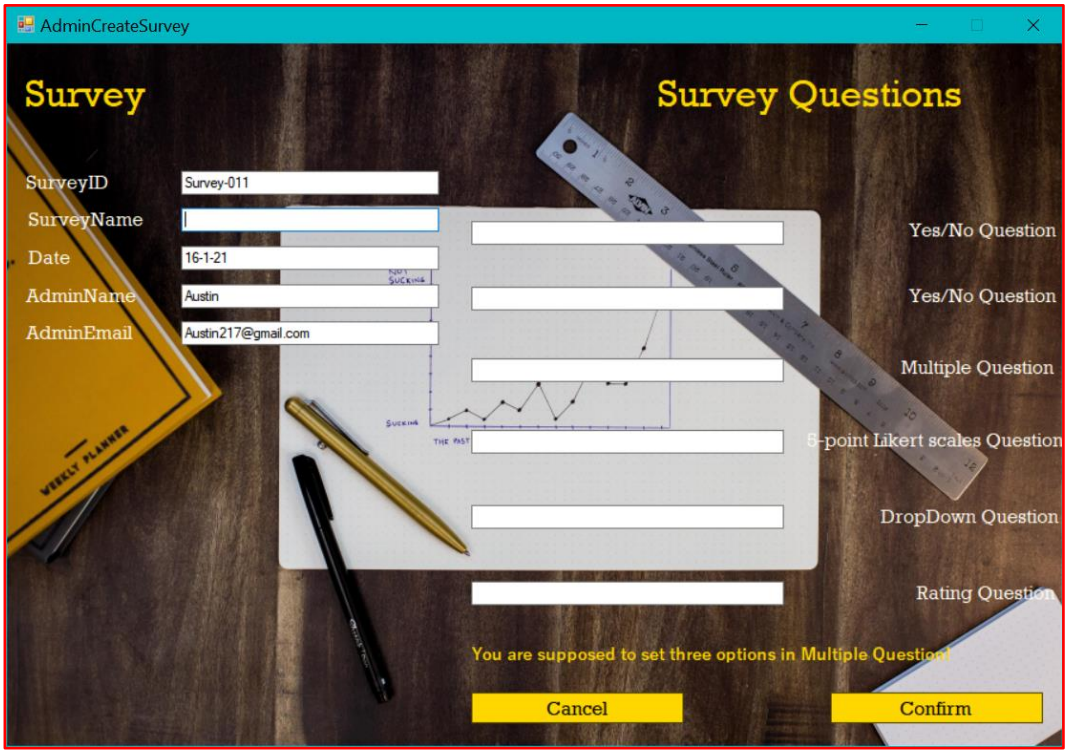


Figure 72

After testing

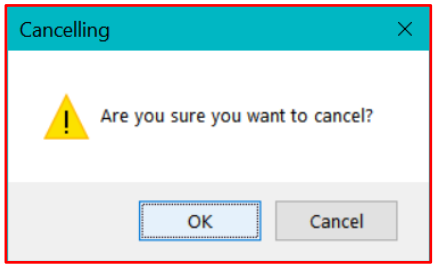


Figure 73

Test Case	Description	Test Procedure	Expected Result	Actual results
6.1	Testing the dropdown box	Click on the arrow of dropdown box.	Items in the dropdown list should be appeared	The created survey items are dropped down

Before testing

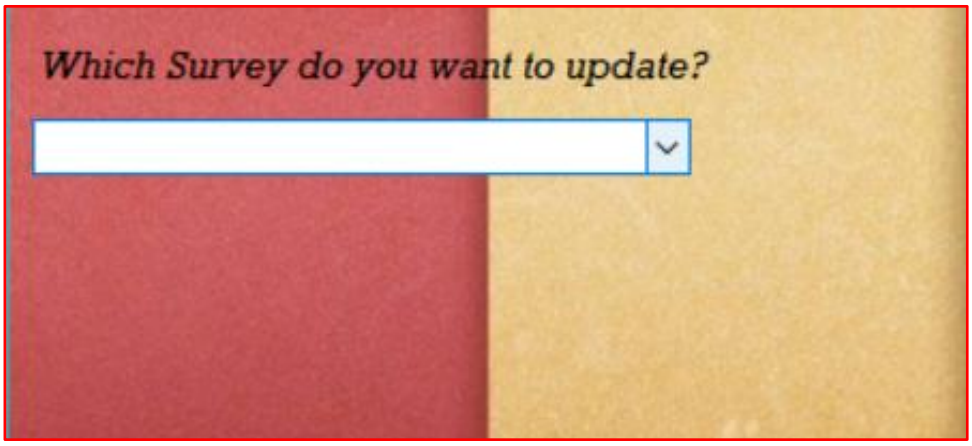


Figure 74

After testing

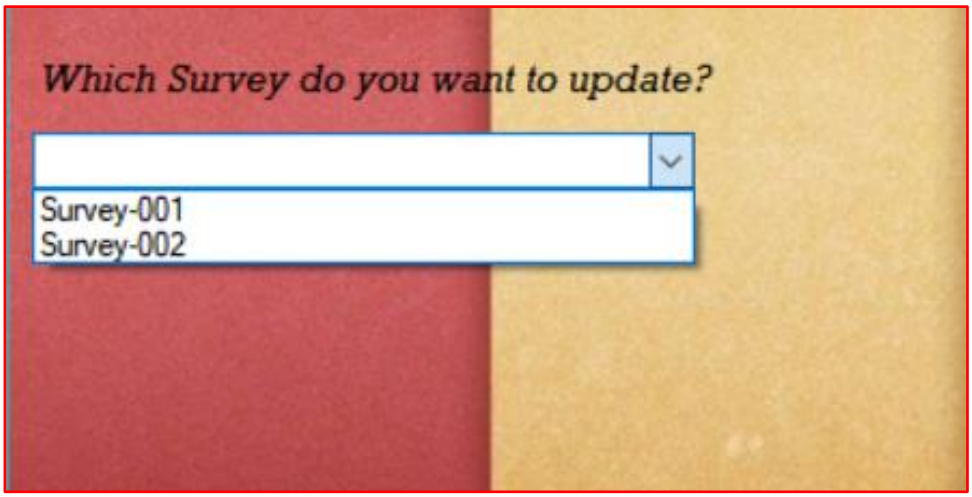


Figure 75

Test Case	Description	Test Procedure	Expected Result	Actual results
6.2	Testing the auto-selecting process	Click on the favorable data grid cell	Questions are able to seen in the respective textboxes	The survey questions are shown up in the textbox.

Before testing

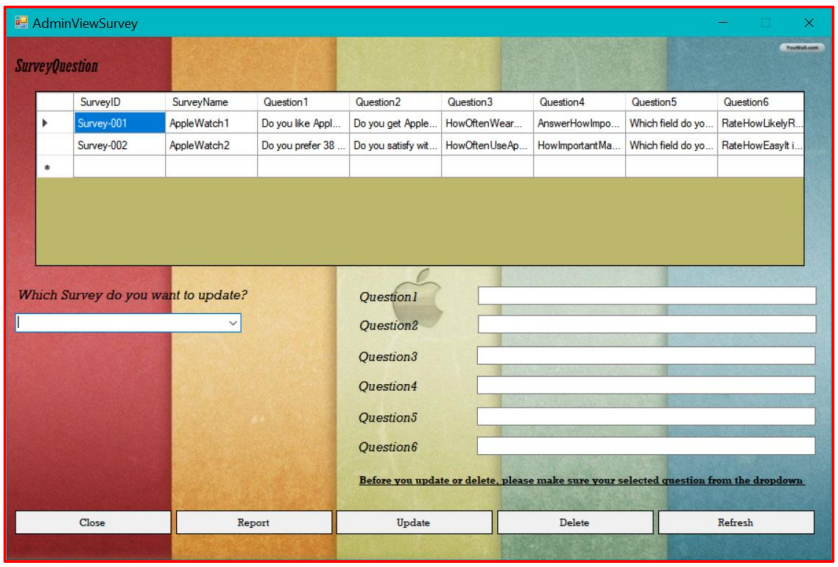


Figure 76

After testing

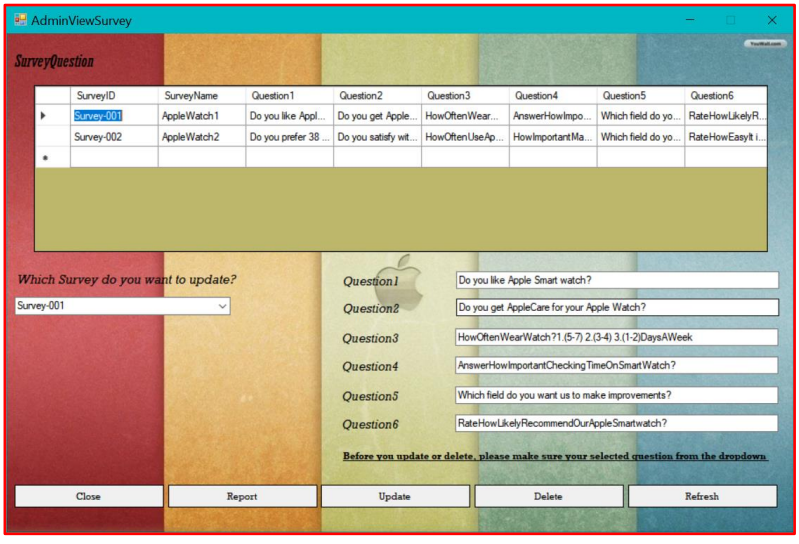


Figure 77

Test Case	Description	Test Procedure	Expected Result	Actual results
6.3	Testing the survey update process	Click on “Update” button	The updated question should be replaced in the original one in the database.	The update complete box appears and updated question is saved in the database.

Before testing

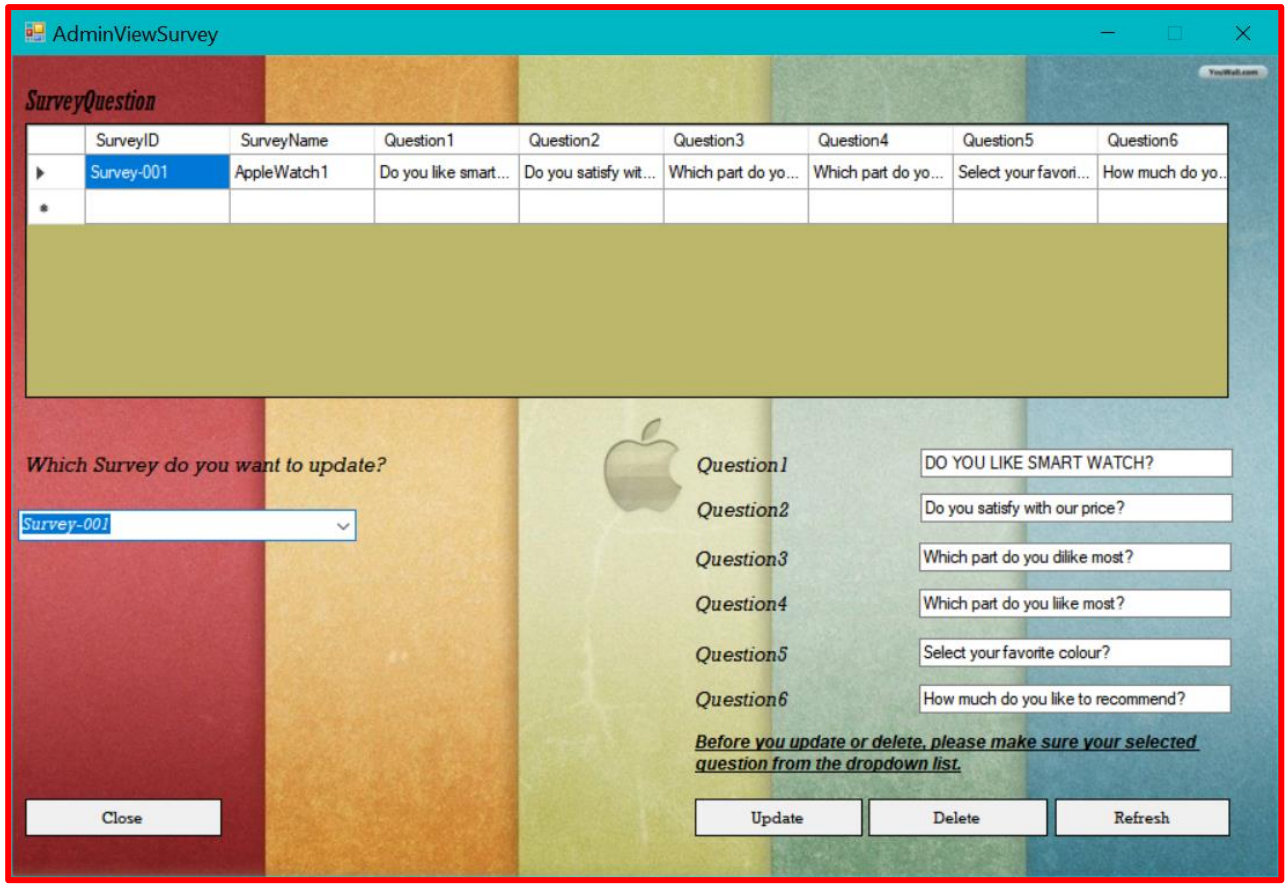


Figure 78

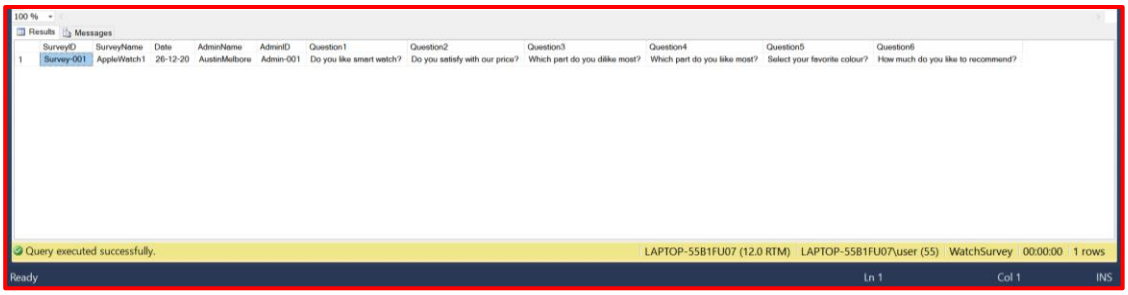


Figure 79

After testing

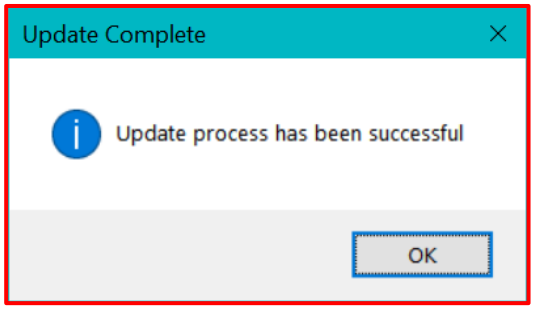


Figure 80

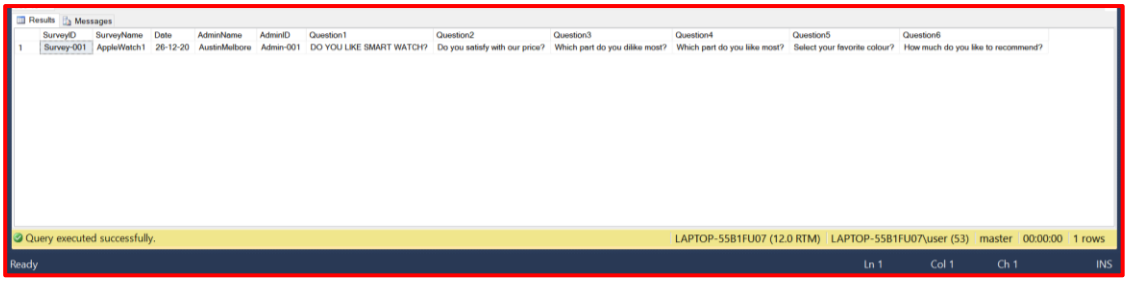


Figure 81

Test Case	Description	Test Procedure	Expected Result	Actual results
6.4	Testing the survey delete process	Click on “Delete” button	The entire selected survey should be deleted.	The dialog box appears which means the survey has been deleted in the database.

Before testing

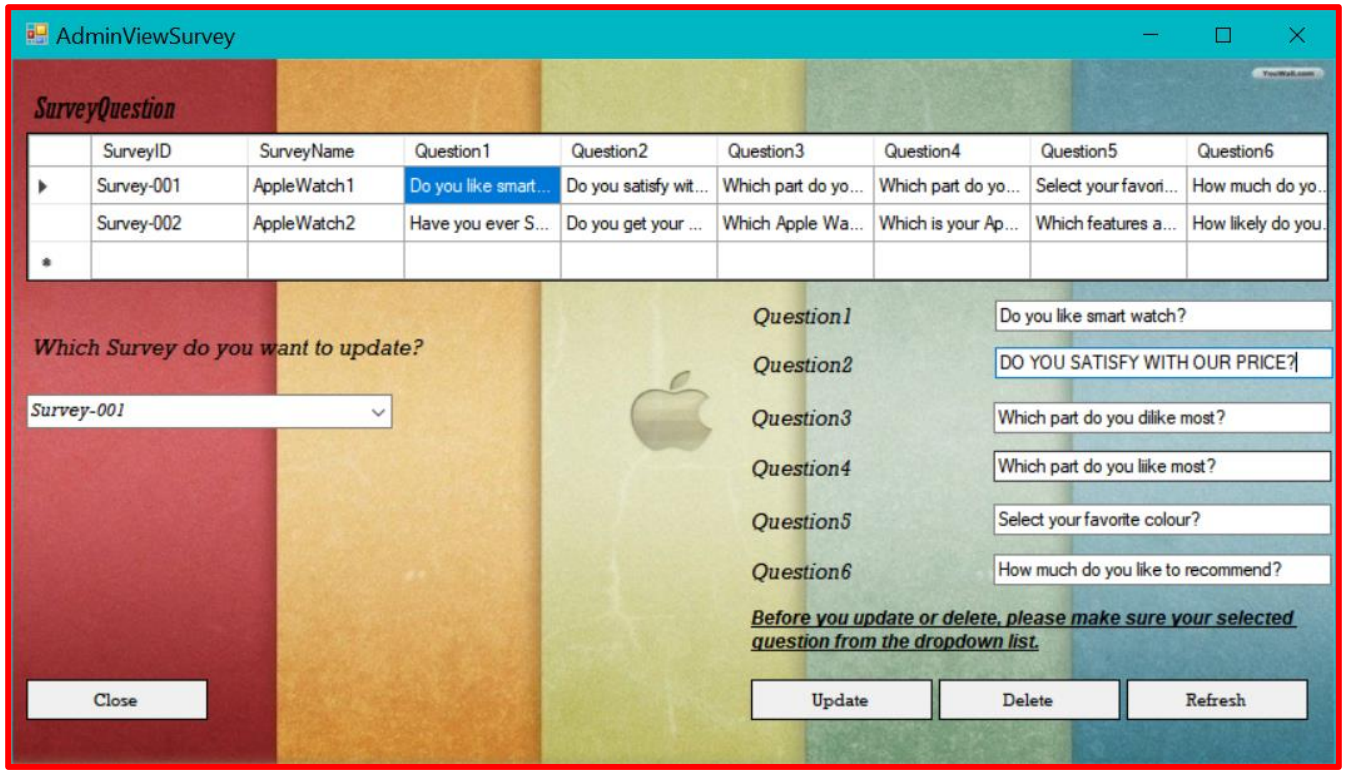
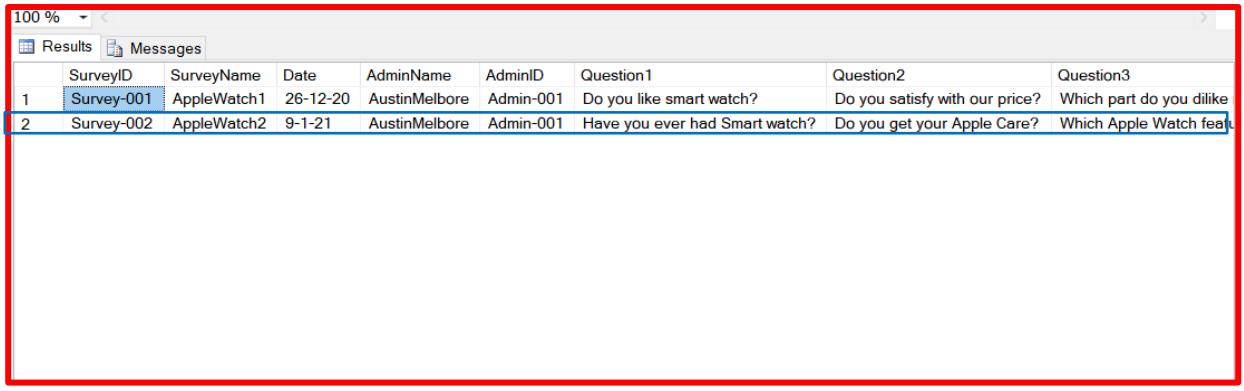


Figure 82 User interface before delete process



	SurveyID	SurveyName	Date	AdminName	AdminID	Question1	Question2	Question3
1	Survey-001	AppleWatch1	26-12-20	AustinMelbore	Admin-001	Do you like smart watch?	Do you satisfy with our price?	Which part do you dilike
2	Survey-002	AppleWatch2	9-1-21	AustinMelbore	Admin-001	Have you ever had Smart watch?	Do you get your Apple Care?	Which Apple Watch feat

Figure 83 Database before delete process

After testing

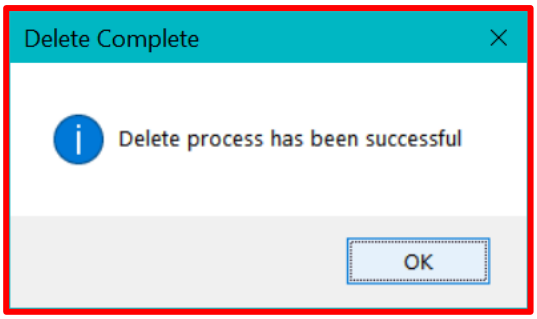
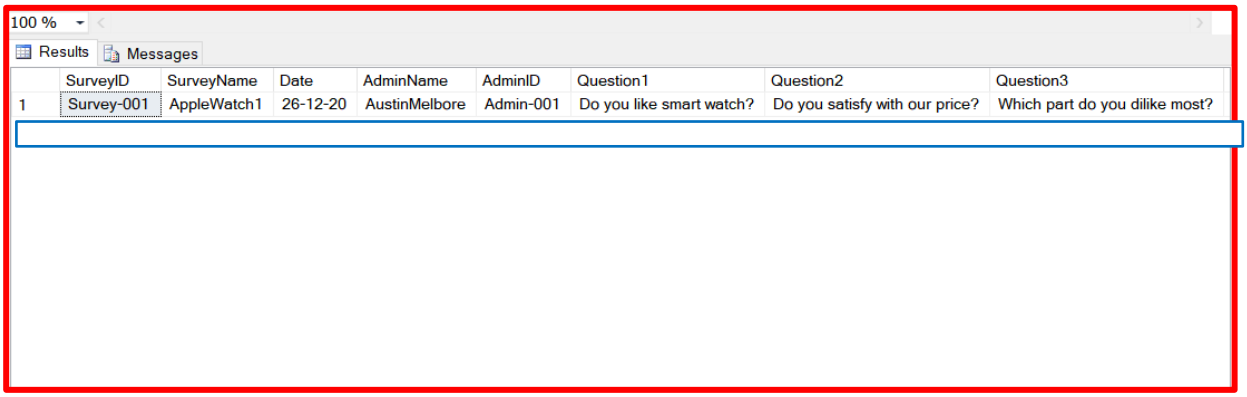


Figure 84 User interface after delete process



	SurveyID	SurveyName	Date	AdminName	AdminID	Question1	Question2	Question3
1	Survey-001	AppleWatch1	26-12-20	AustinMelbore	Admin-001	Do you like smart watch?	Do you satisfy with our price?	Which part do you dilike most?

Figure 85 Database after delete process

Test Case	Description	Test Procedure	Expected Result	Actual results
6.5	Testing the “Close” button	Click on “Close” button	Able to close the page	The Admin Create survey page closes and returns back to the Admin dashboard

Before testing

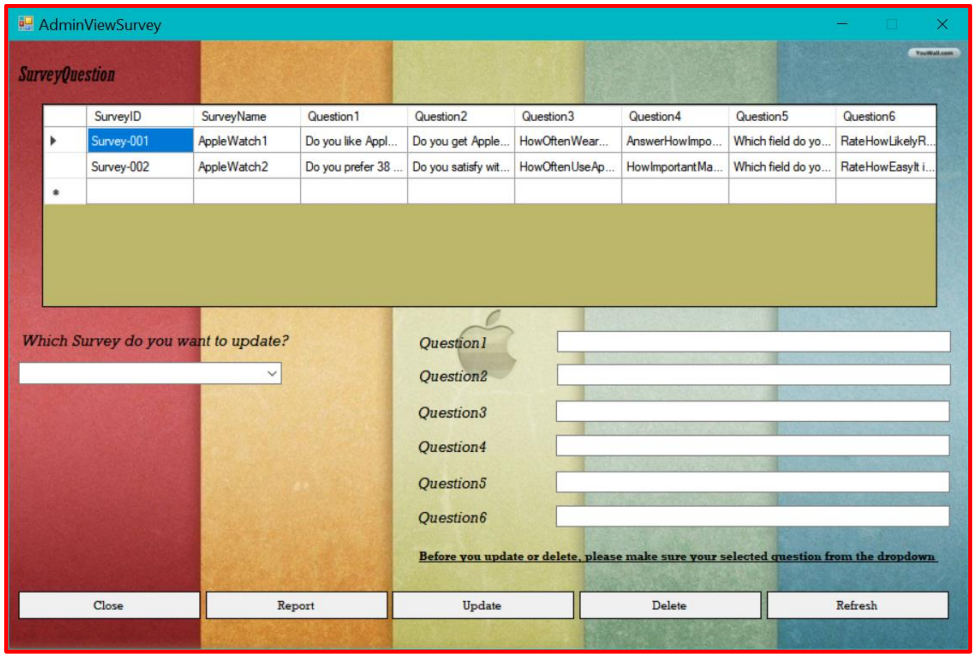


Figure86

After testing

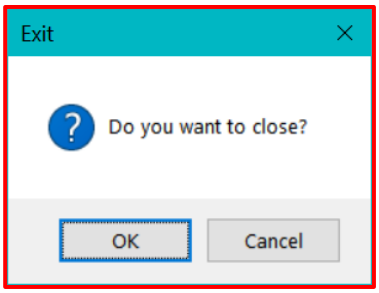


Figure87

Test Case	Description	Test Procedure	Expected Result	Actual results
6.6	Testing the “Report” button	Click on “Report” button	Able to go to the “Report Form” page	The “Report Form” page pops up after selecting survey in the previous page

Before testing

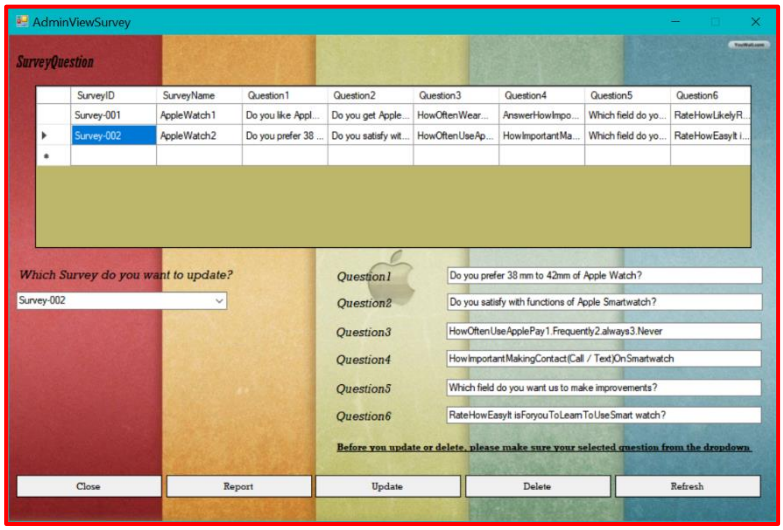


Figure 88

After testing

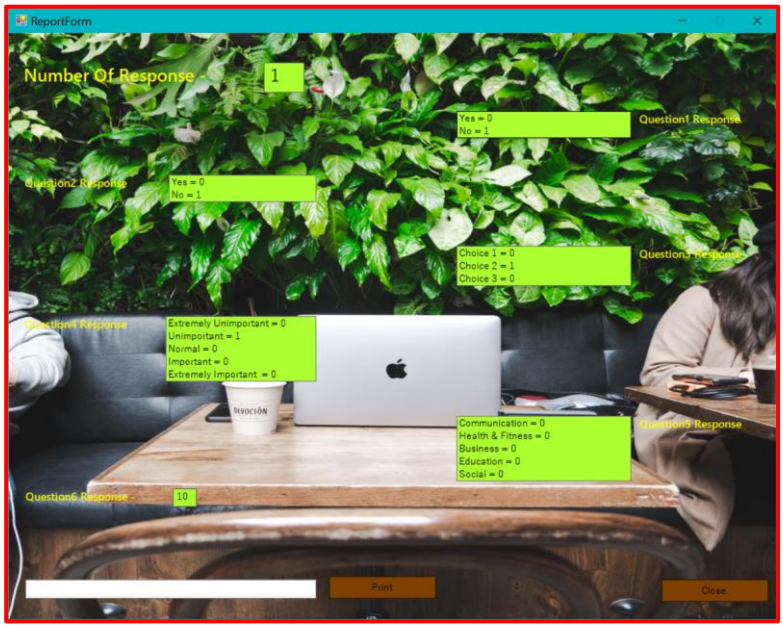


Figure 89

Test Case	Description	Test Procedure	Expected Result	Actual results
7.1	Testing the welcome page	Click on picture labeled “Admin”.	Able to go to the admin login.	Admin login page appears

Before testing

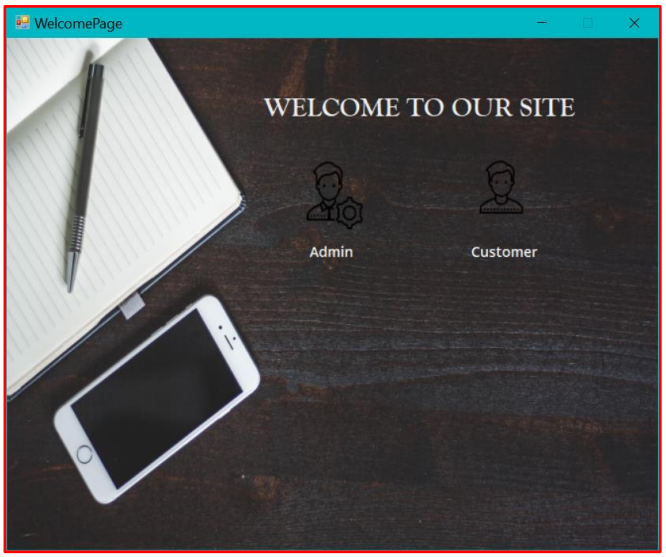


Figure 90

After testing

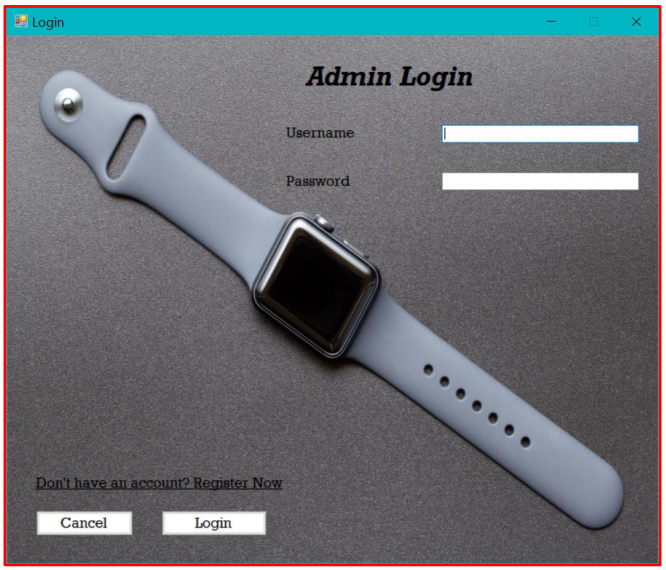


Figure 91

Test Case	Description	Test Procedure	Expected Result	Actual results
7.2	Testing the welcome page	Click on picture labeled “Customer”.	Able to go to the customer login.	Customer login page appears

Before testing

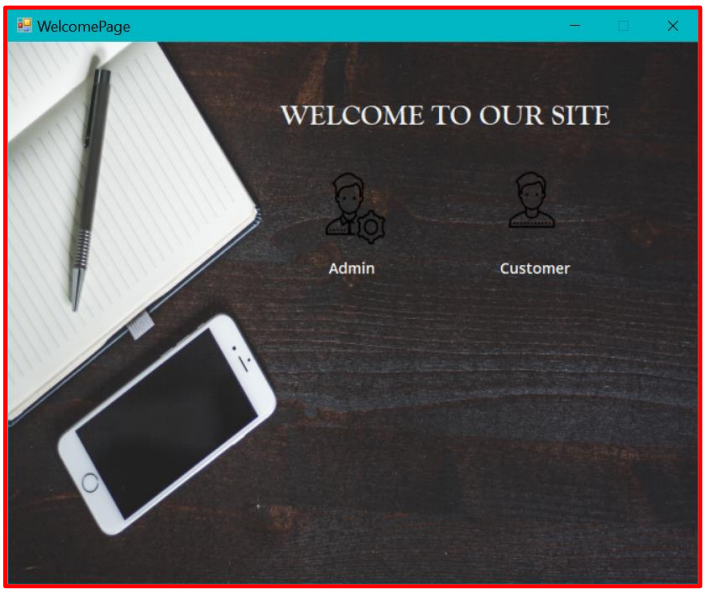


Figure 92

After testing

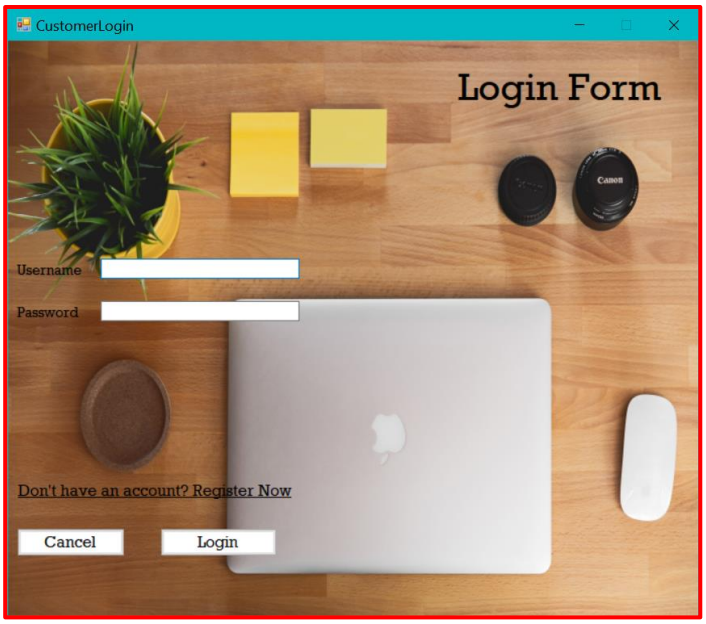


Figure 93

Test Case	Description	Test Procedure	Expected Result	Actual results
8.1	Testing the checkbox	Enable the checkbox label	Should allow to enable the register button	The unable register button will be enabled

Before testing

Figure 94

After testing

Figure 95

Test Case	Description	Test Procedure	Expected Result	Actual results
8.2	Testing the registration process	Click on the “Register” with incomplete information	Should not allow to continue when the information is not completely filled in	The “Registration successful” dialog box appears and give back the registration form again.

Before testing

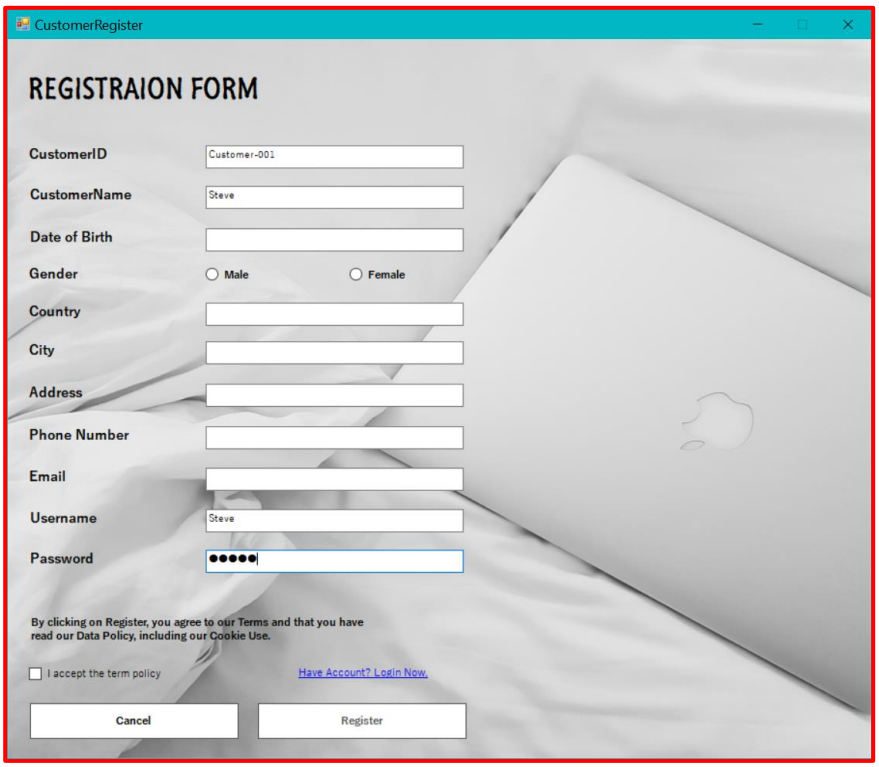


Figure 96

After testing

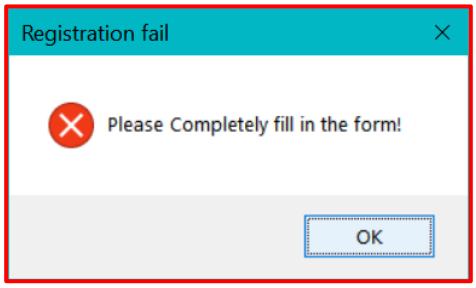


Figure 97

Test Case	Description	Test Procedure	Expected Result	Actual results
8.3	Testing the registration process	Enter valid information, then click on the “Register” button.	Registered data should be saved in the Database and dialog box should also be shown for confirmation	For client side, the successful dialog box appears and returns back to the Login Page when clicking on “OK”. For sever side, new account must has been created.

Before testing

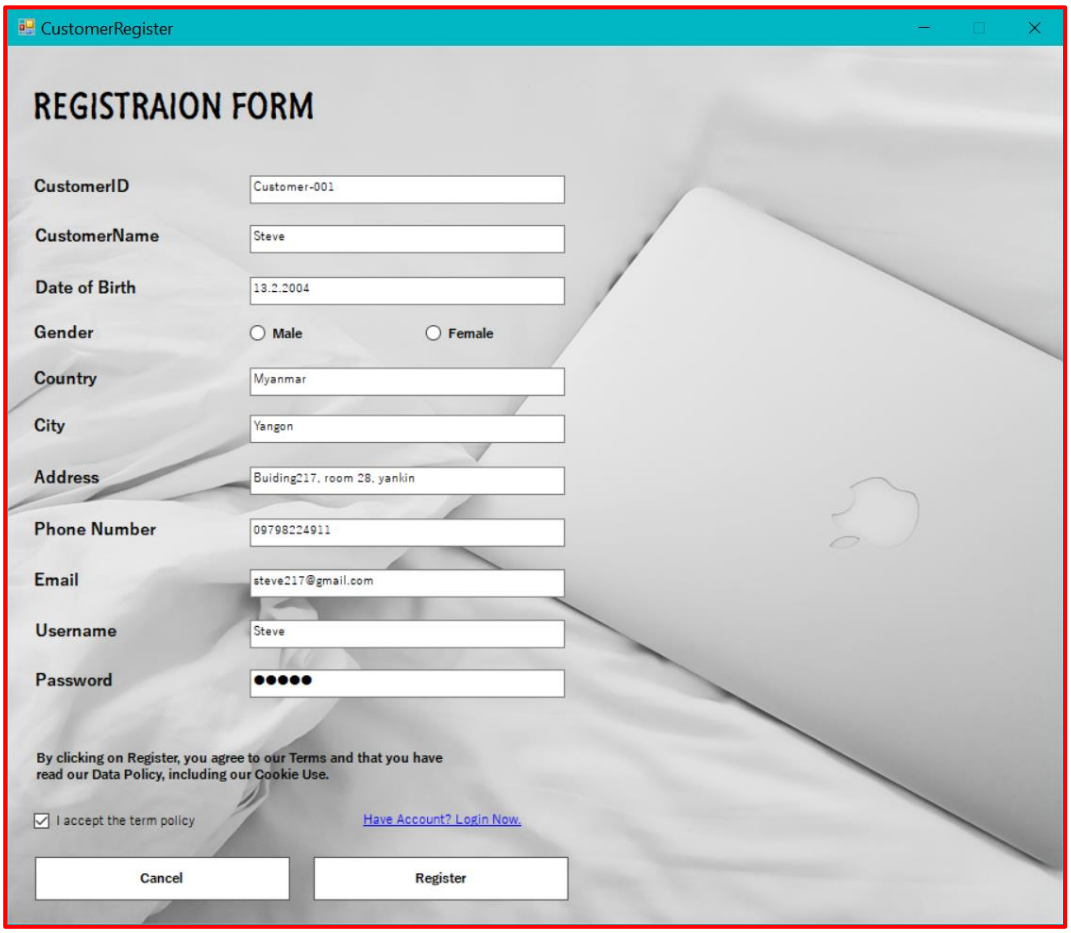


Figure 98 User Interface before registration

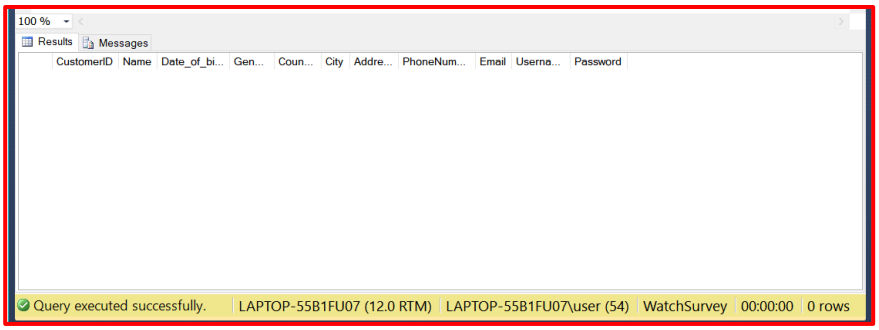


Figure 99 Database before registration

After testing

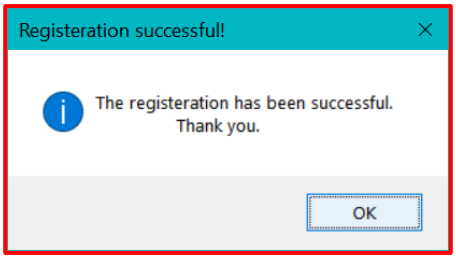


Figure 100 User Interface after registration

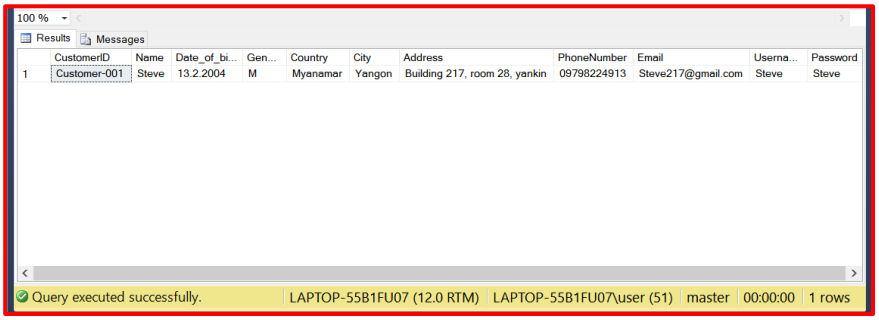


Figure 101 result in Database after registration

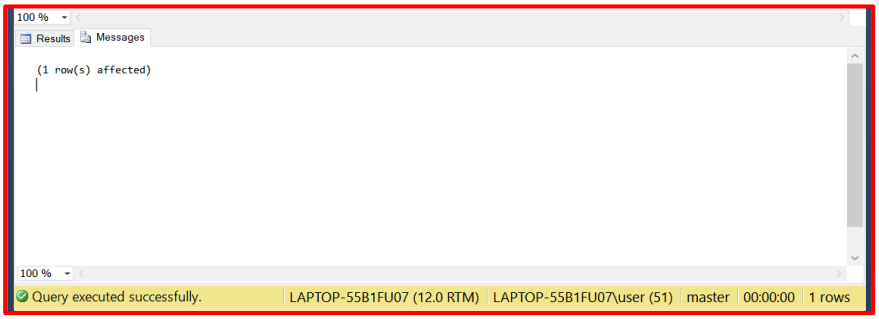


Figure 102 message in Database after registration

Test Case	Description	Test Procedure	Expected Result	Actual results
8.4	Testing the link label	Click on the blue-colored link label	Able to go to the Login Page	The login page appears.

Before testing

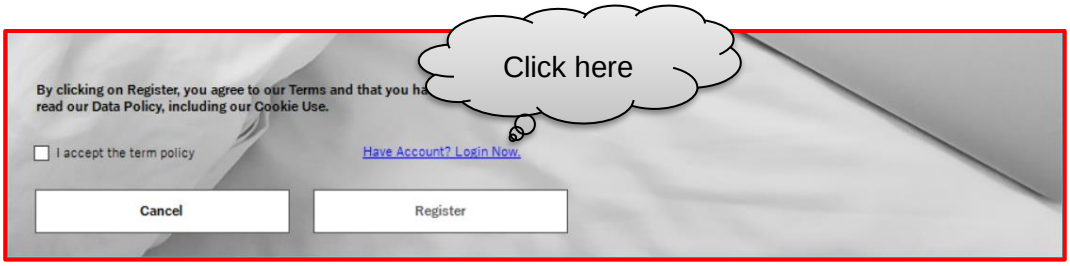


Figure 103

After testing

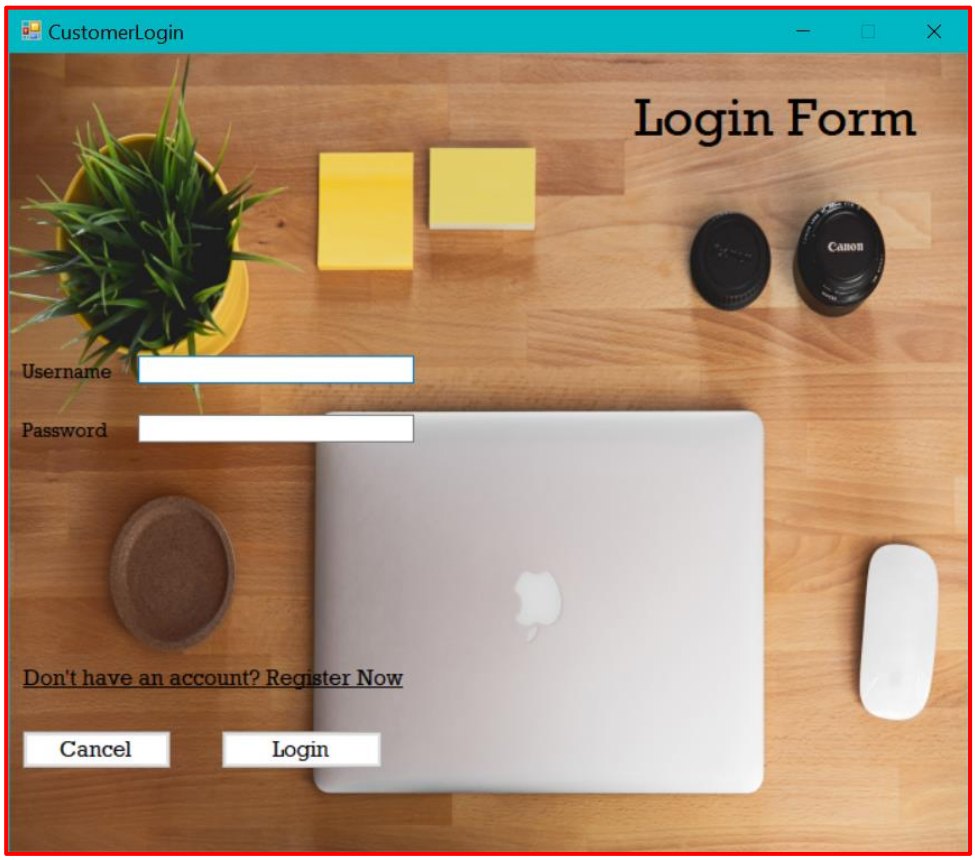


Figure 104

Test Case	Description	Test Procedure	Expected Result	Actual results
8.5	Testing the close button	Click on the “Cancel” button.	Able to close the program	Confirmation box shows to close the program

Before testing

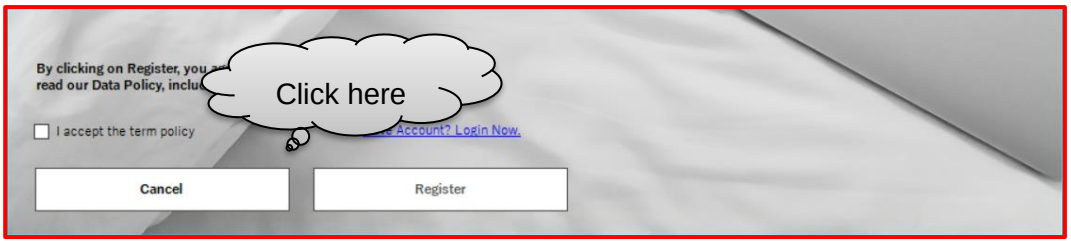


Figure 105

After testing

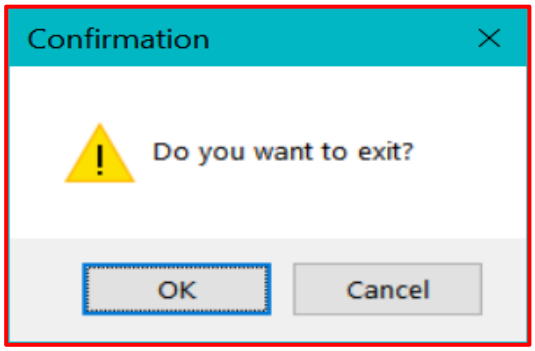


Figure 106

Test Case	Description	Test Procedure	Expected Result	Actual results
9.1	Testing the login	Fill the username and password and then click on the “Login” button.	Should allow login for correct information	“Login successful” dialog box appears and admin Dashboard page has to come up.

Before testing

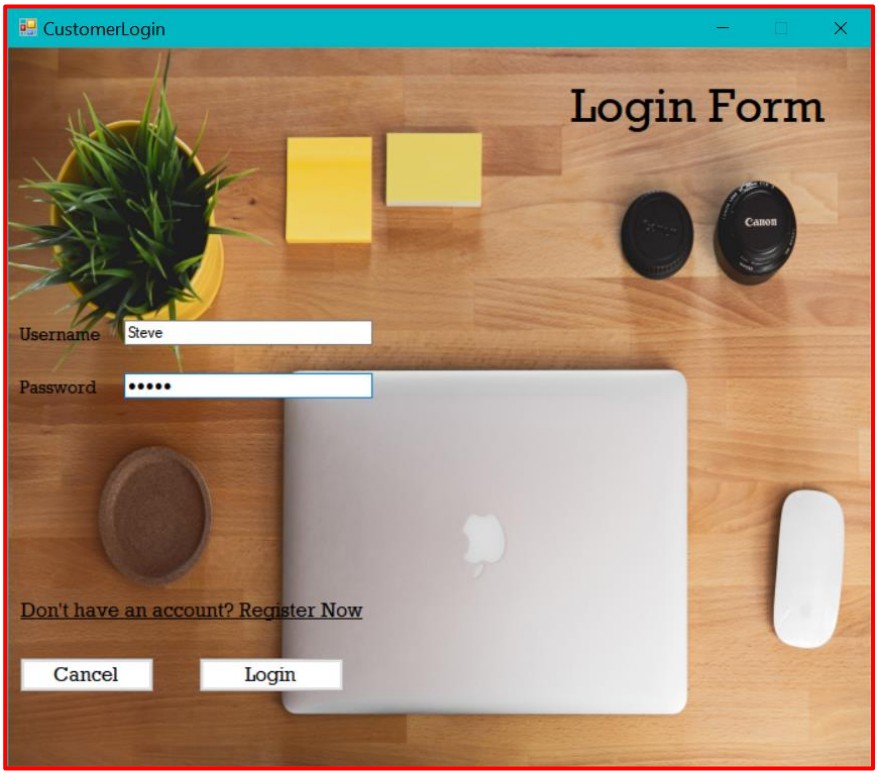


Figure 107

After testing

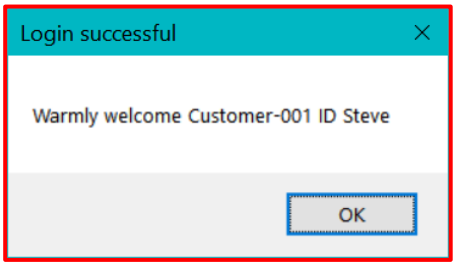


Figure 108

Test Case	Description	Test Procedure	Expected Result	Actual results
9.2	Testing the invalid login	Fill in the blank and enter Login.	Should not allow Login with invalid username or password	“Login fail” dialog box appears.

Before testing

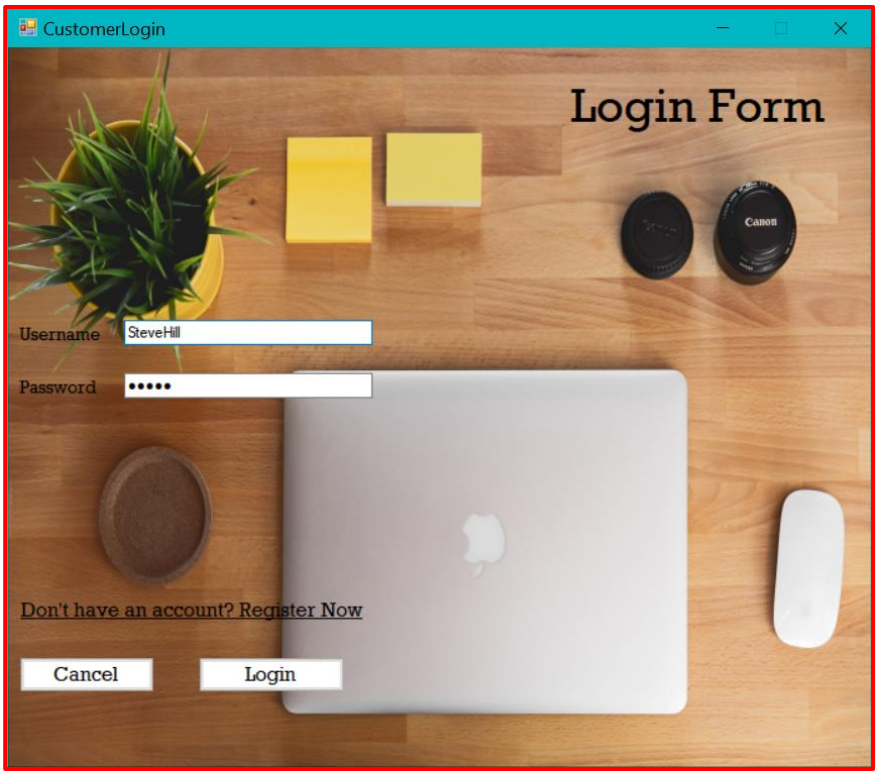


Figure 109

After testing

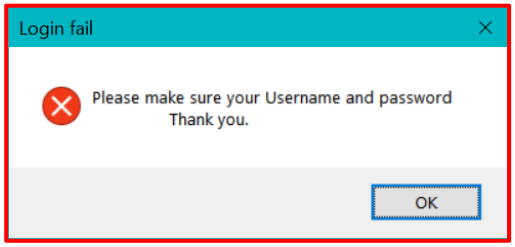


Figure 110

Test Case	Description	Test Procedure	Expected Result	Actual results
9.3	Testing the invalid login	Incomplete form fill-in	Should not allow Login with incomplete information	“Login Incomplete” dialog box appears.

Before testing

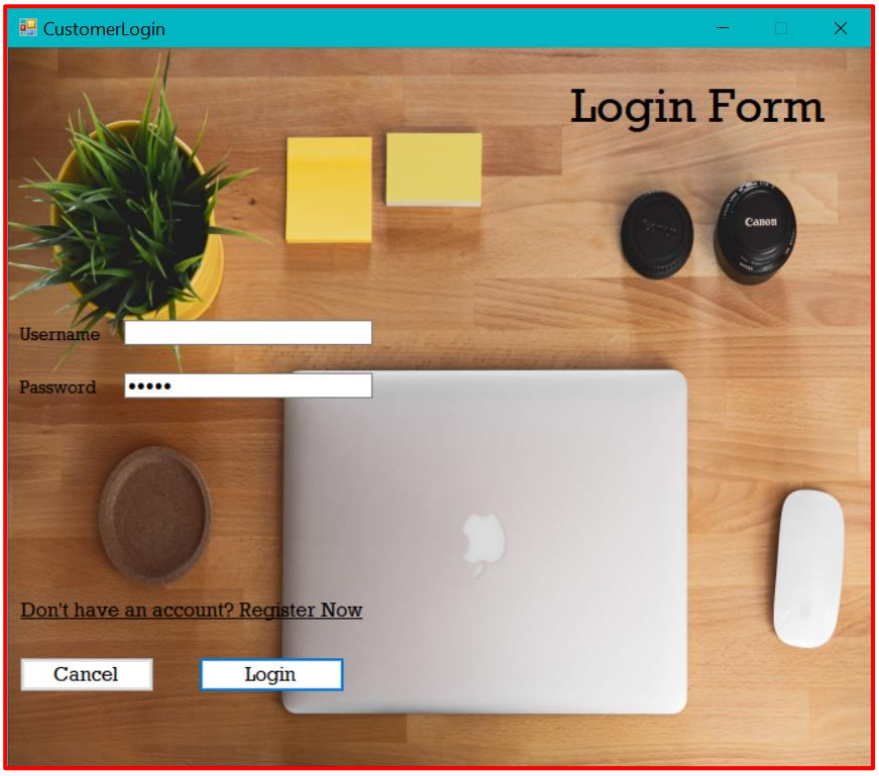


Figure 111

After testing

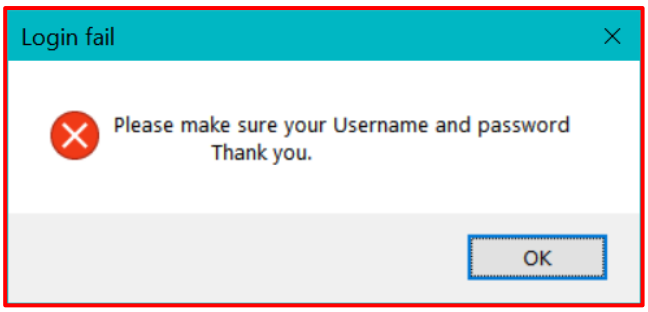


Figure 112

Test Case	Description	Test Procedure	Expected Result	Actual results
9.4	Testing the link label of returning back to the registration page	Click on the “Don’t have an account? Register Now” link Label.	Able to enter the Registration page.	Registration page appears.

Before testing

Figure 113

After testing

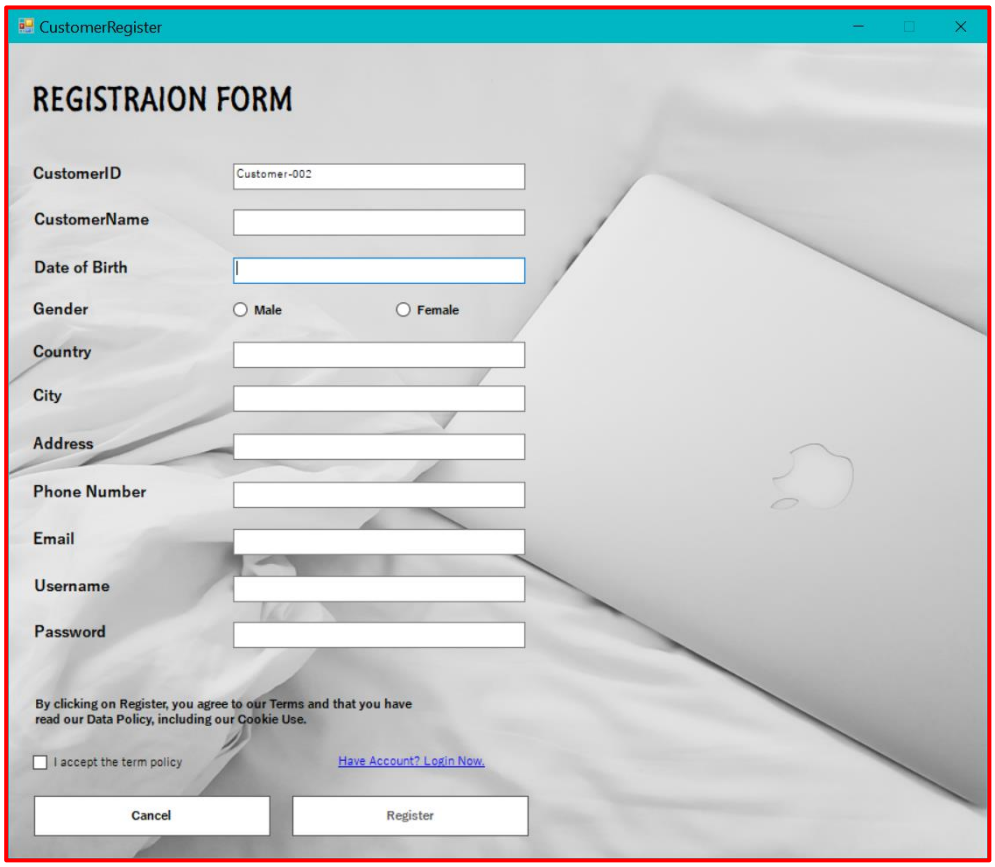


Figure 114

Test Case	Description	Test Procedure	Expected Result	Actual results
9.5	Testing the cancel button	Click on the Cancel button.	Able to close the entire program	Cancelling box appear and the program closes.

Before testing

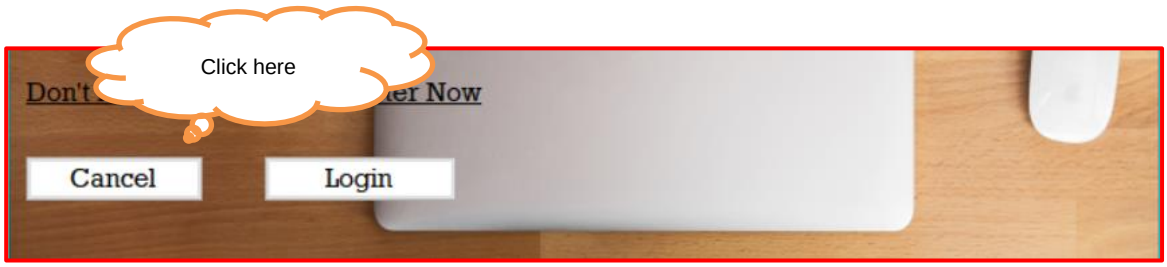


Figure 115

After testing

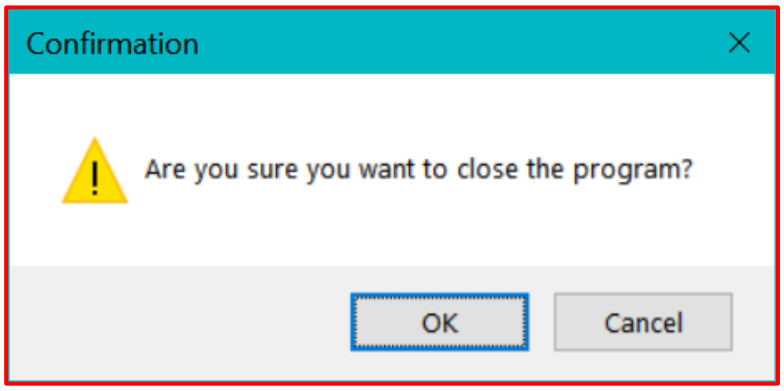


Figure 116

Test Case	Description	Test Procedure	Expected Result	Actual results
10.1	Testing the features	Click on the picture at the middle of the screen.	New features should be able to appear	Some new features appear on the screen.

Before testing

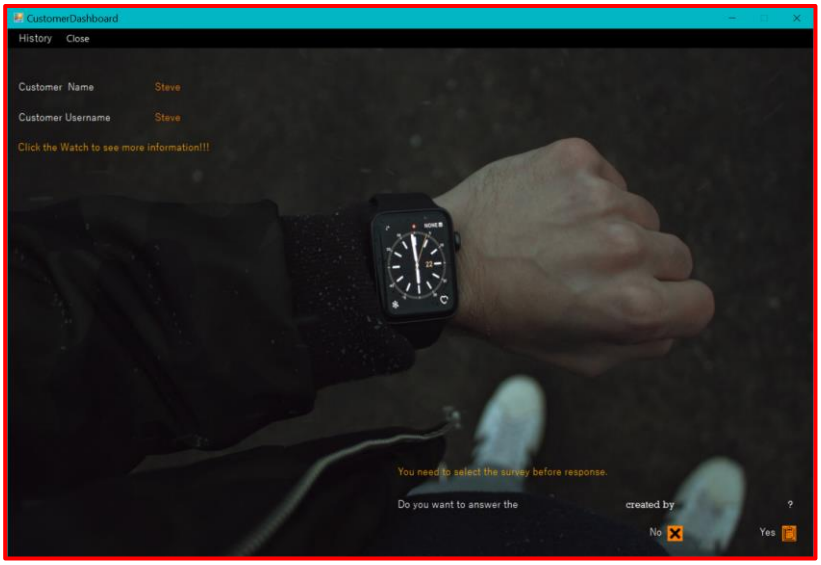


Figure 117

After testing

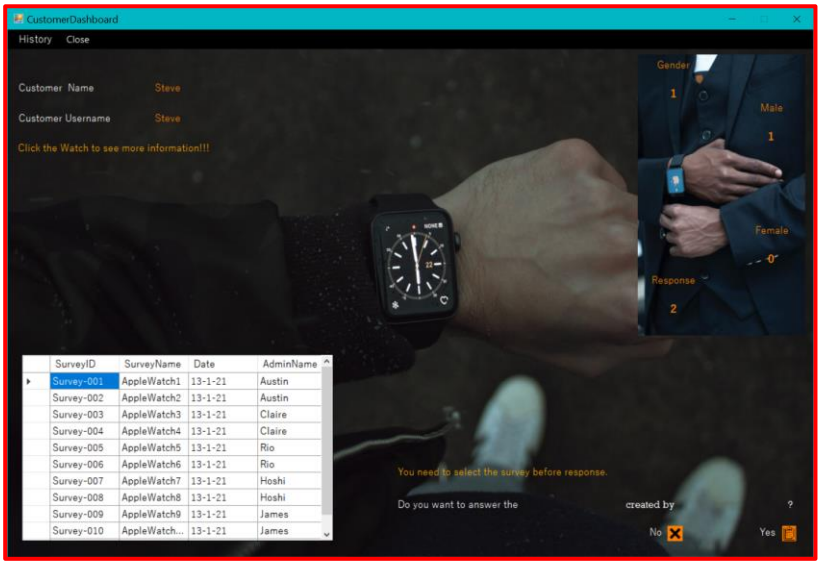


Figure 118

Test Case	Description	Test Procedure	Expected Result	Actual results
10.2	Testing the features	Click on the table cell	The blank spaces should be filled with information	The survey ID and admin Name of selected survey appear in the respective blank spaces.

Before testing

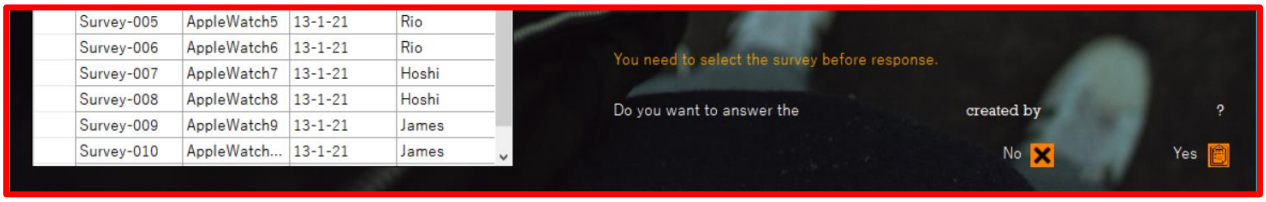


Figure 119

After testing

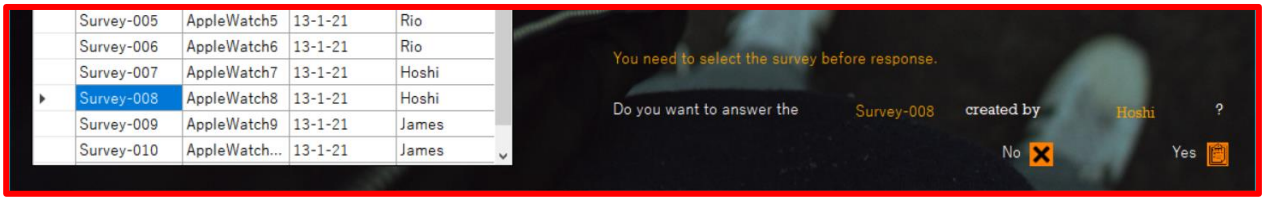


Figure 120

Test Case	Description	Test Procedure	Expected Result	Actual results
10.3	Testing the features	Click on the cross sign	The variable information should disappear	The survey ID and admin Name of selected survey disappear in the respective blank spaces.

Before testing

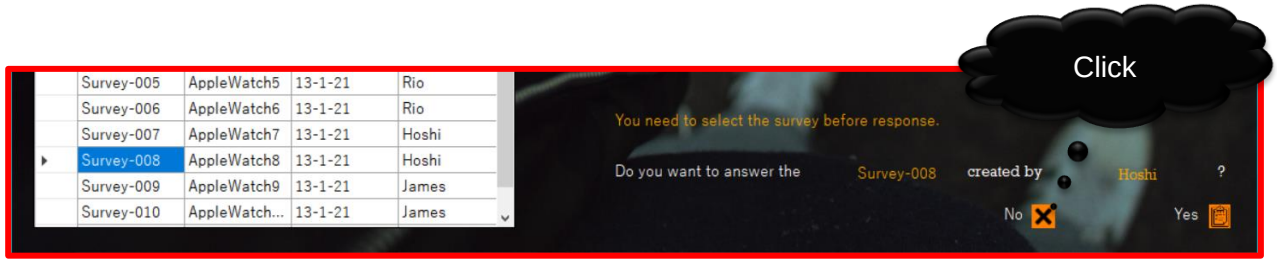


Figure 121

After testing

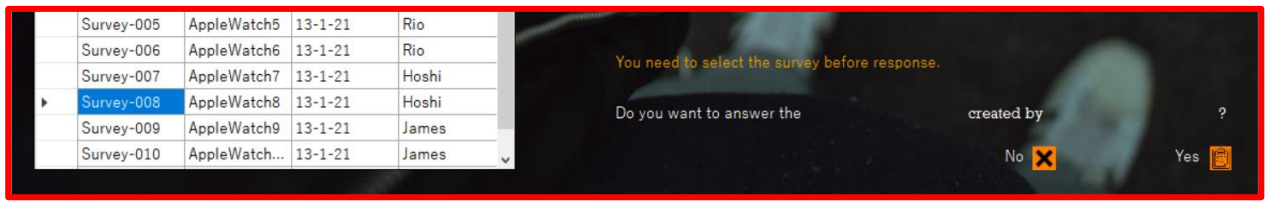


Figure 122

Test Case	Description	Test Procedure	Expected Result	Actual results
10.4	Testing the picture labelled by “Yes” without selecting	Click on the survey picture	Should not allow to continue	“Fail” dialog box appears

Before testing

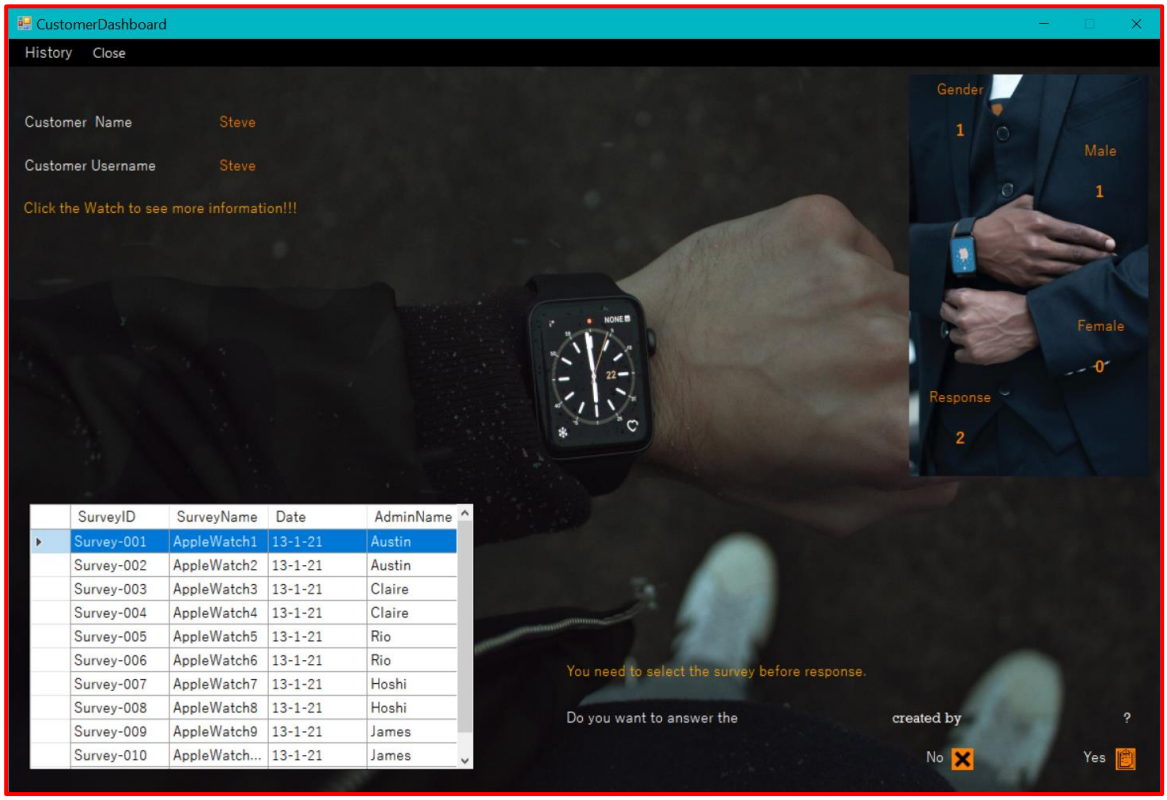


Figure 123

After testing

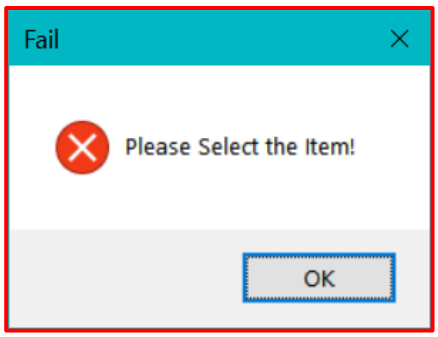


Figure 124

Test Case	Description	Test Procedure	Expected Result	Actual results
10.5	Testing the “History” button	Click on “History” button	Should offer History page	History page appears

Before testing

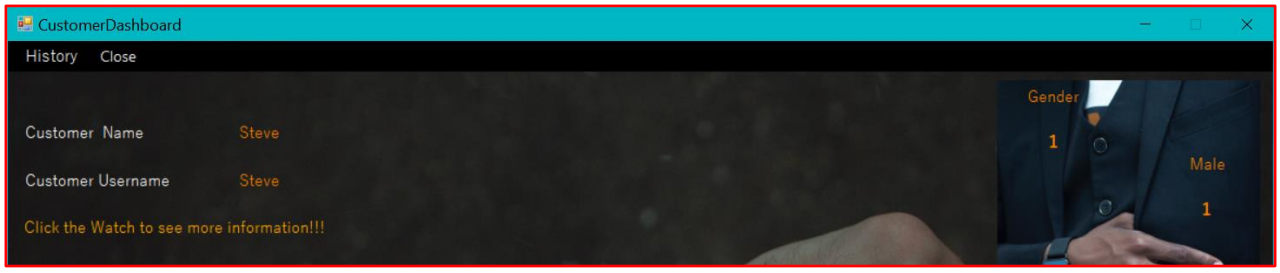


Figure 125

After testing

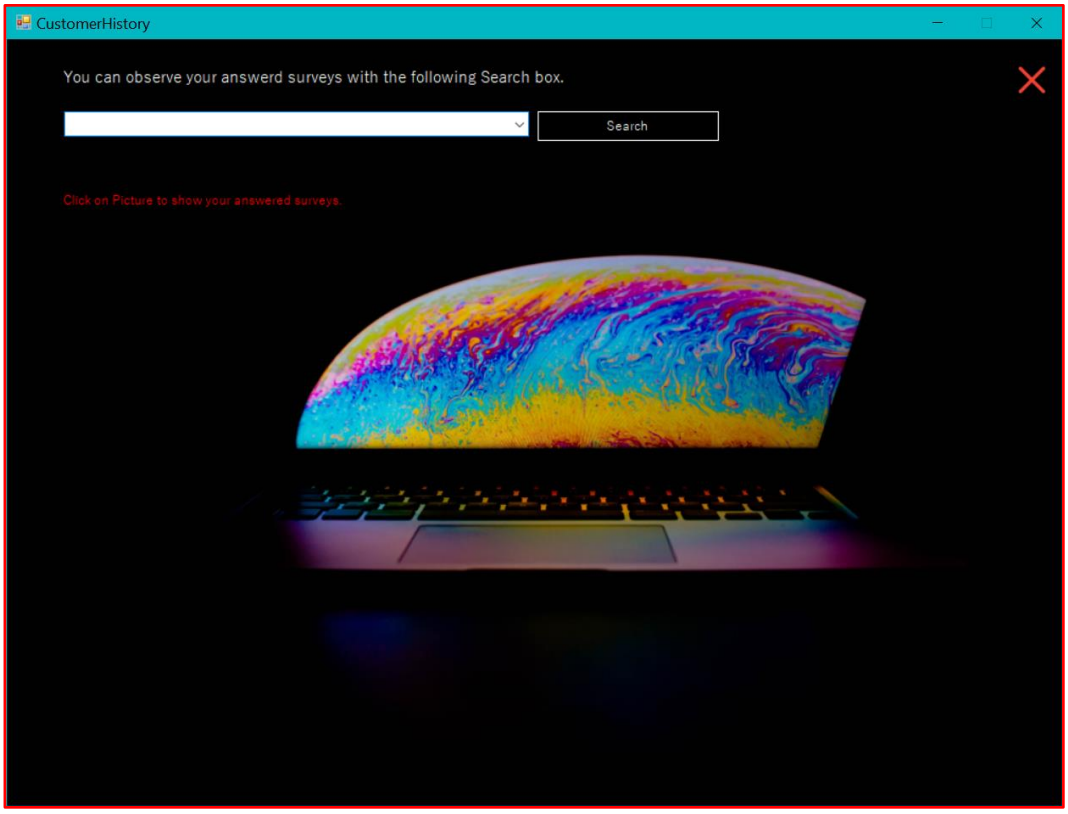


Figure 126

Test Case	Description	Test Procedure	Expected Result	Actual results
10.6	Testing the “Close” button	Click on “Close” button	Should allow to close the whole page	The entire page closes.

Before testing

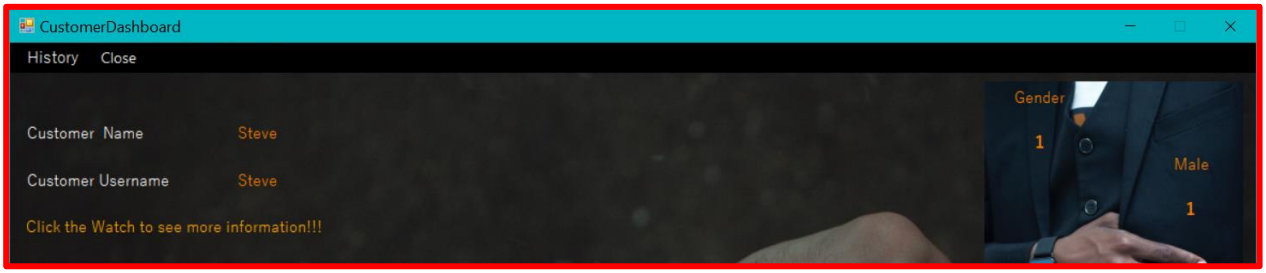


Figure 127

After testing

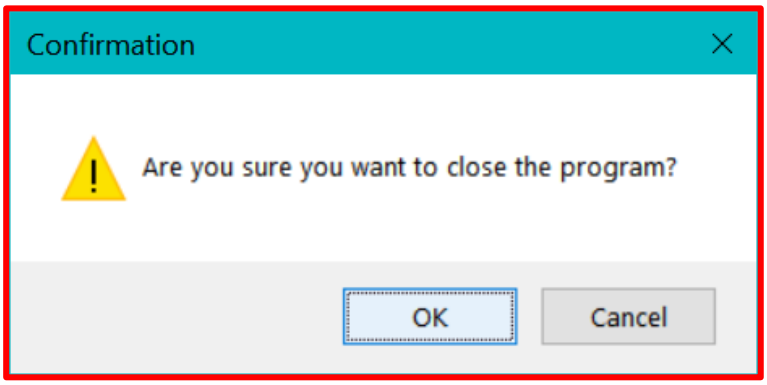


Figure 128

Test Case	Description	Test Procedure	Expected Result	Actual results
10.7	Testing the picture labelled by “Yes” after selecting	Click on picture beside “yes” label	Should allow to go to the “Customer Answer Survey” page	“Customer Answer Survey” page appears

Before testing

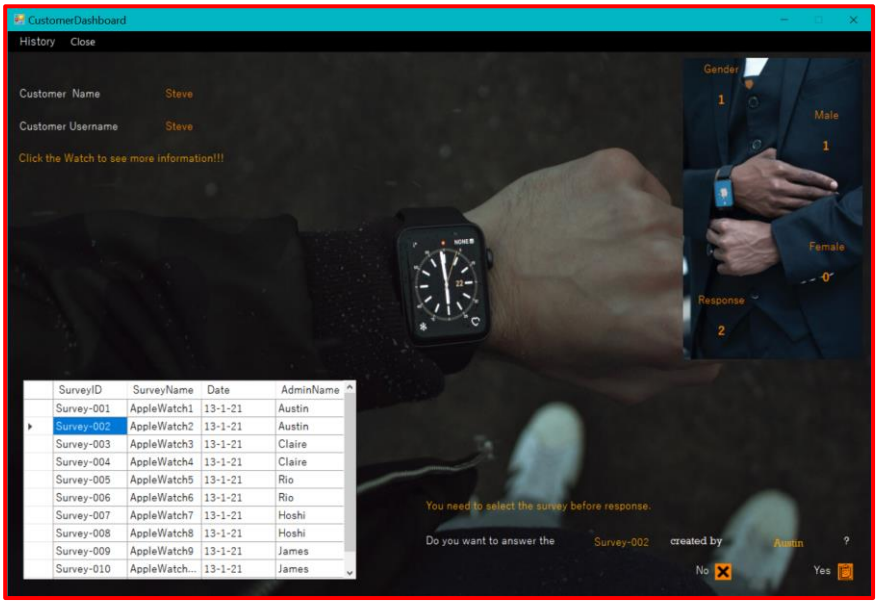


Figure 129

After testing

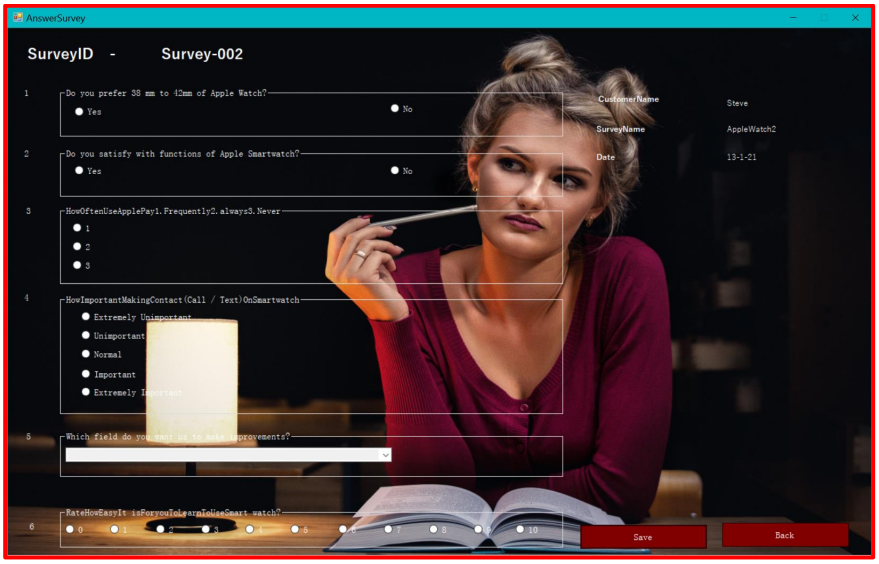


Figure 130

Test Case	Description	Test Procedure	Expected Result	Actual results
11.1	Testing the saving process	Click on “Save” button	Responses should be saved in database	For client side, the successful dialog box appears. For server side, the responses have been saved

Before testing

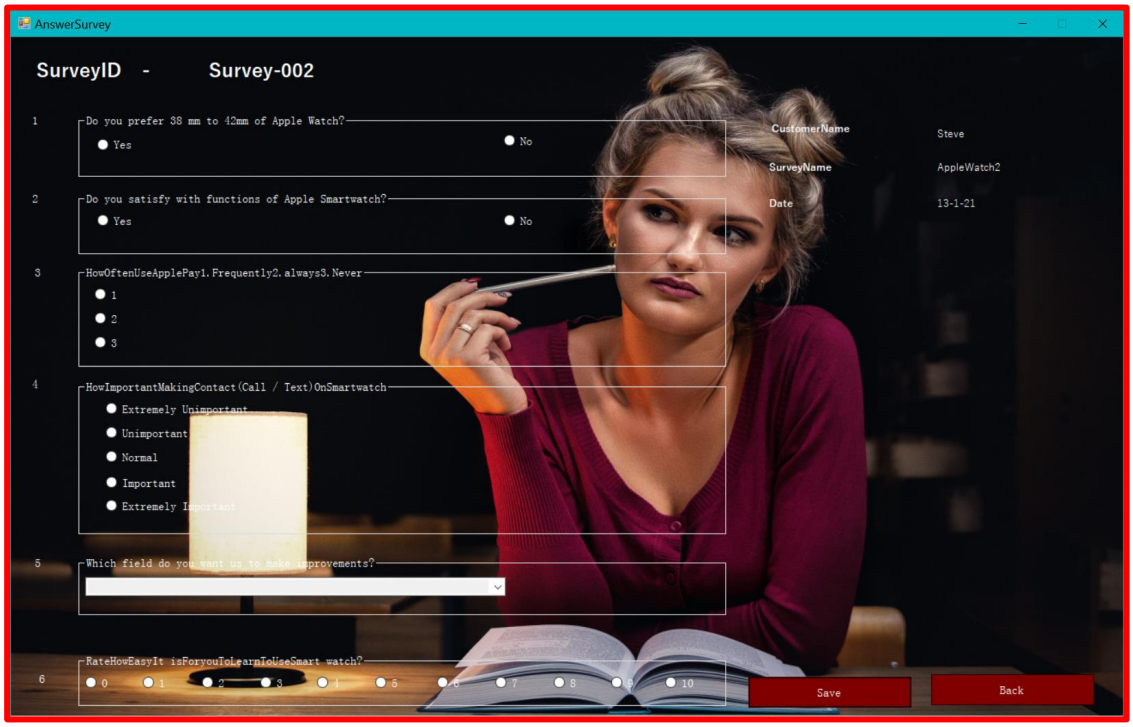


Figure 131User interface before saving

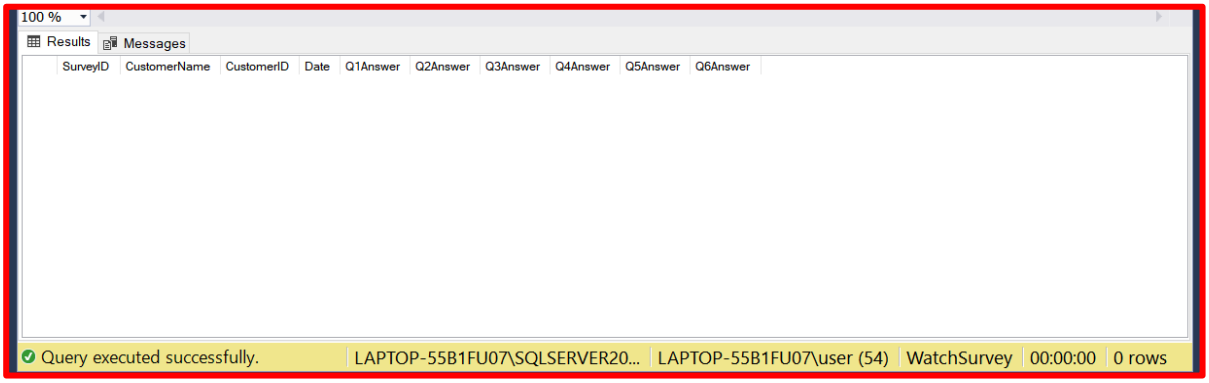


Figure132 database before saving

After testing

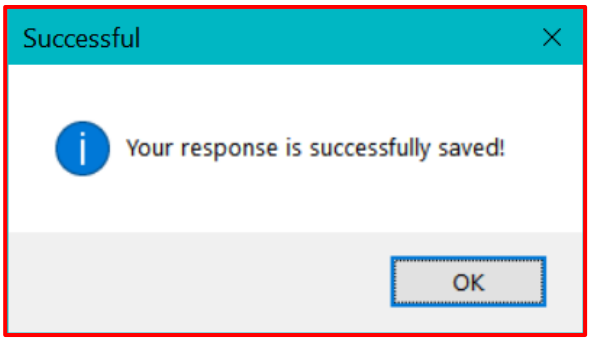


Figure 133 User interface after saving

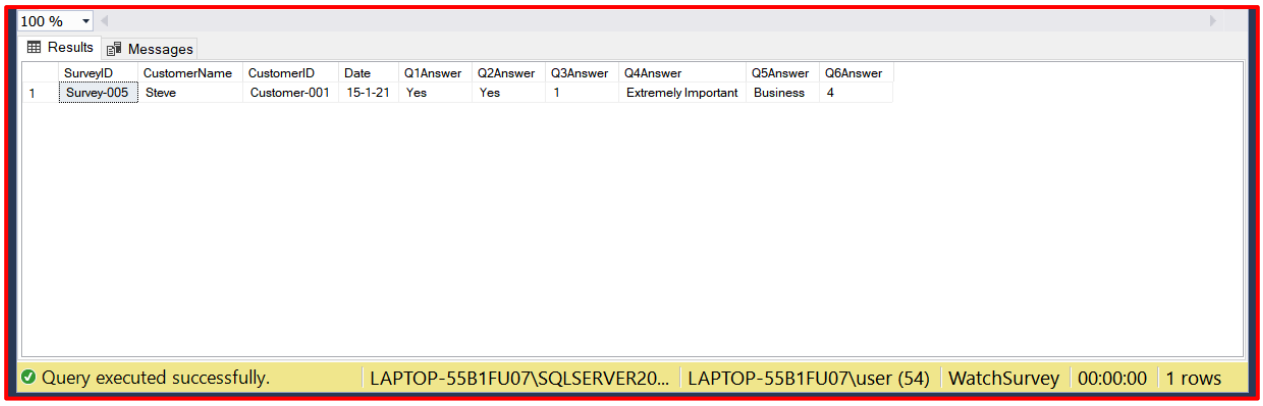


Figure 134 results in Database after saving

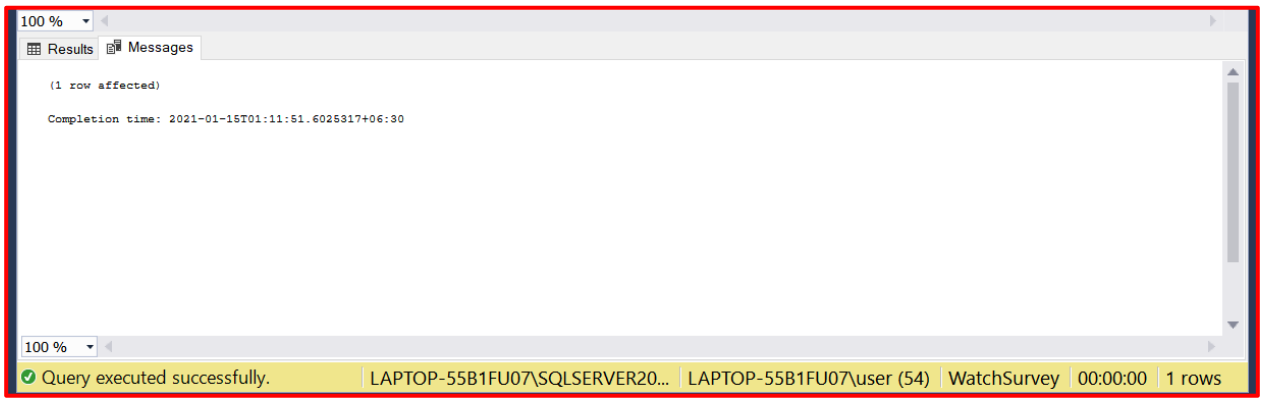


Figure 135 message in Database after saving

Test Case	Description	Test Procedure	Expected Result	Actual results
11.2	Testing the close function	Click on “Back” button	The page should close	The current page closes after hitting the “Back” button

Before testing

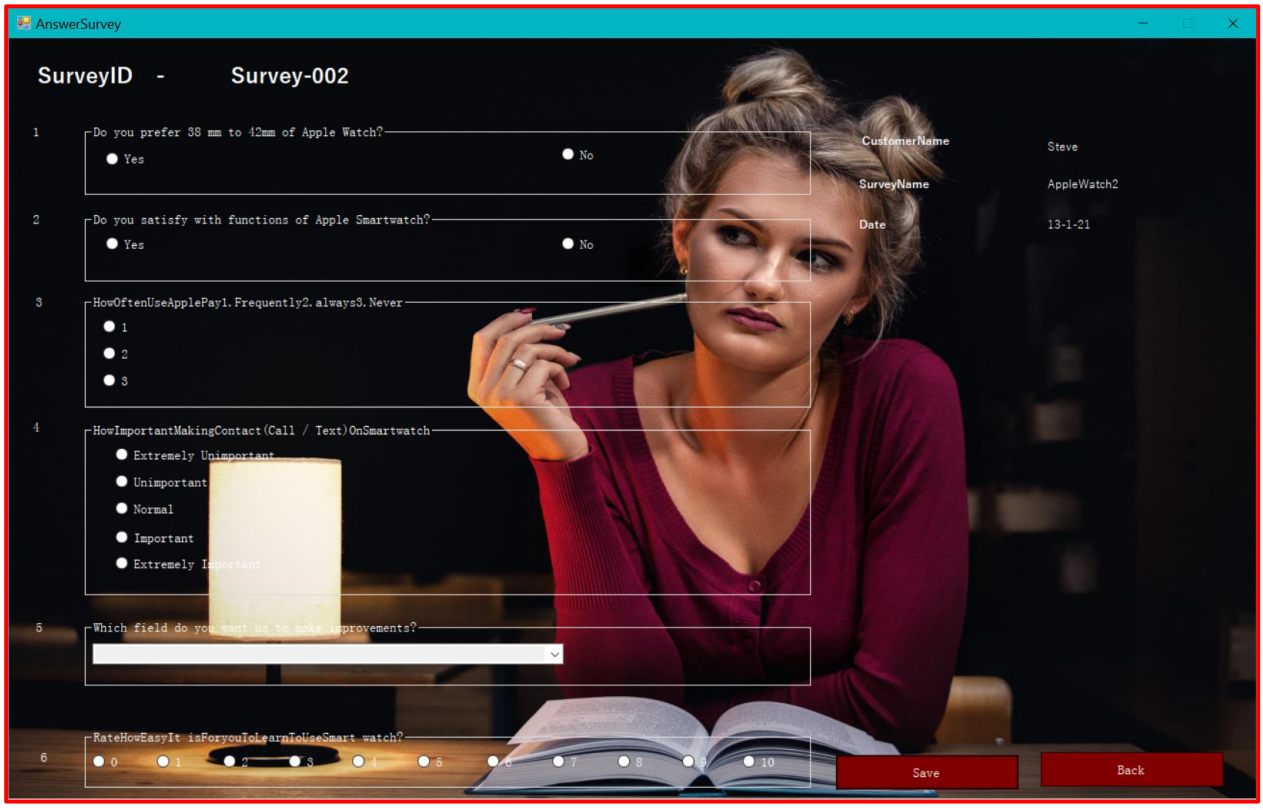


Figure 136

After testing

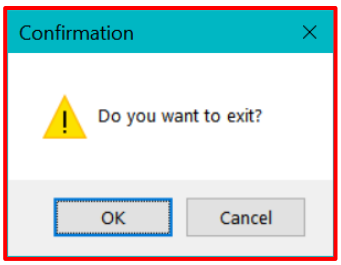


Figure 137

Test Case	Description	Test Procedure	Expected Result	Actual results
12.1	Testing the picture	Click on anywhere on picture	The table should appear	The table including responses and survey questions appears.

Before testing

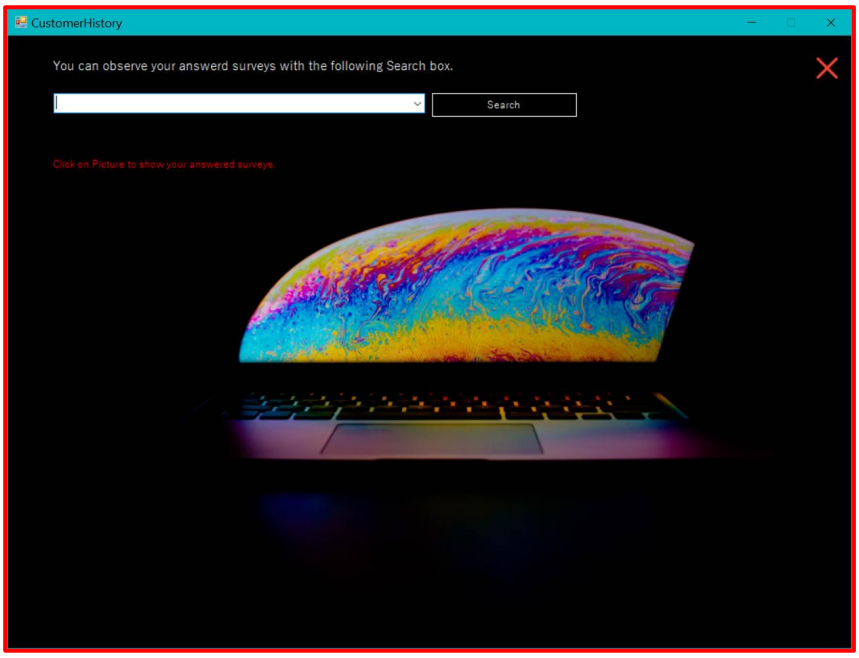


Figure 138

After testing

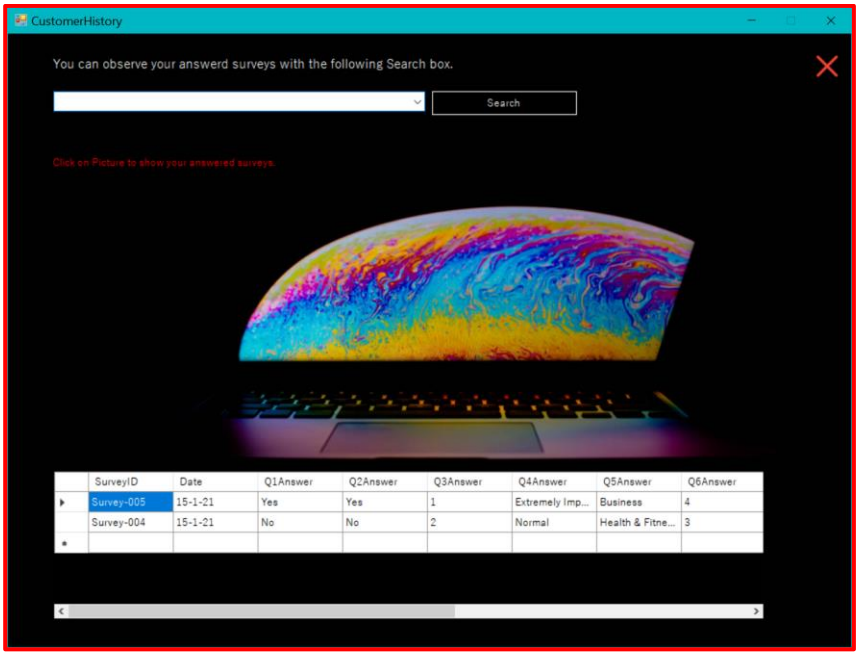


Figure 139

Test Case	Description	Test Procedure	Expected Result	Actual results
12.2	Testing the “Search” button	Select the survey in the search box and hit the “Search” button	The table should show only response and others of selected survey	The selected survey and other information appear in table

Before testing

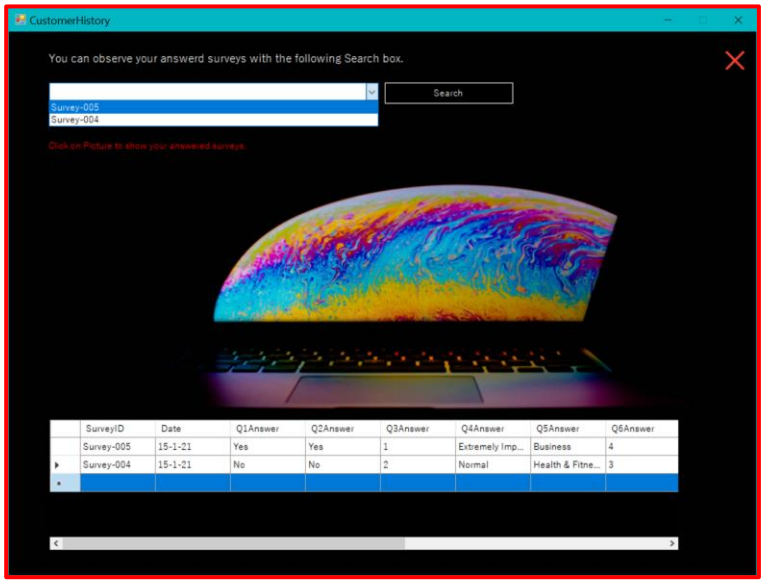


Figure140

After testing

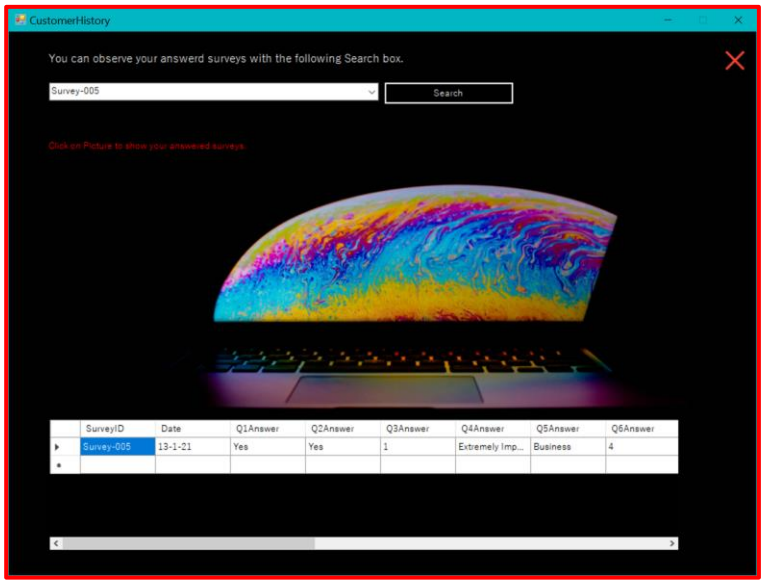


Figure 141

Test Case	Description	Test Procedure	Expected Result	Actual results
12.3	Testing the close button	Hit on “Close” button	The page should close	The “Customer History” disappears when clicking on “OK” in dialog box

Before testing

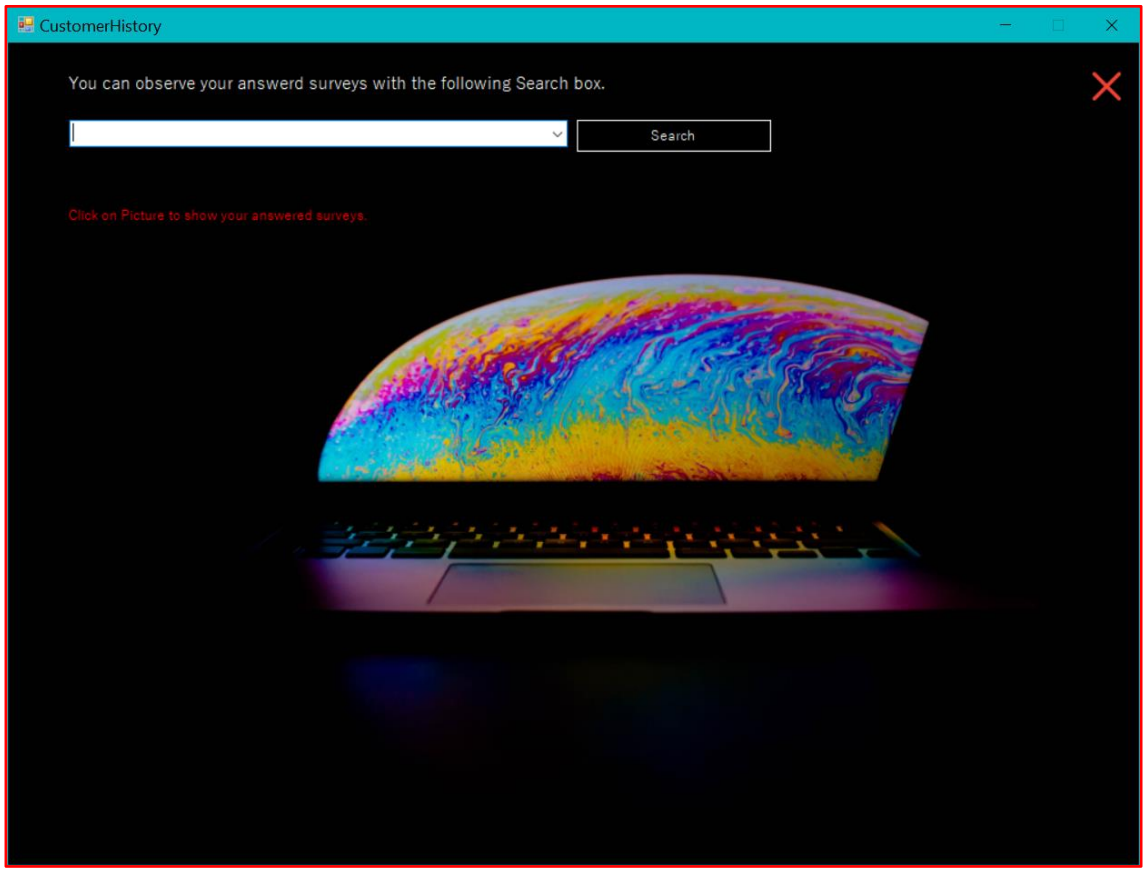


Figure 142

After testing

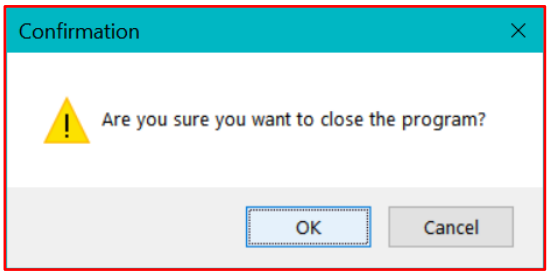


Figure 143

Test Case	Description	Test Procedure	Expected Result	Actual results
13.1	Testing the report process	Hit on “Print” button	The file should be saved.	The “Report complete” dialog box appears and file has been saved in Debug folder.

Before testing

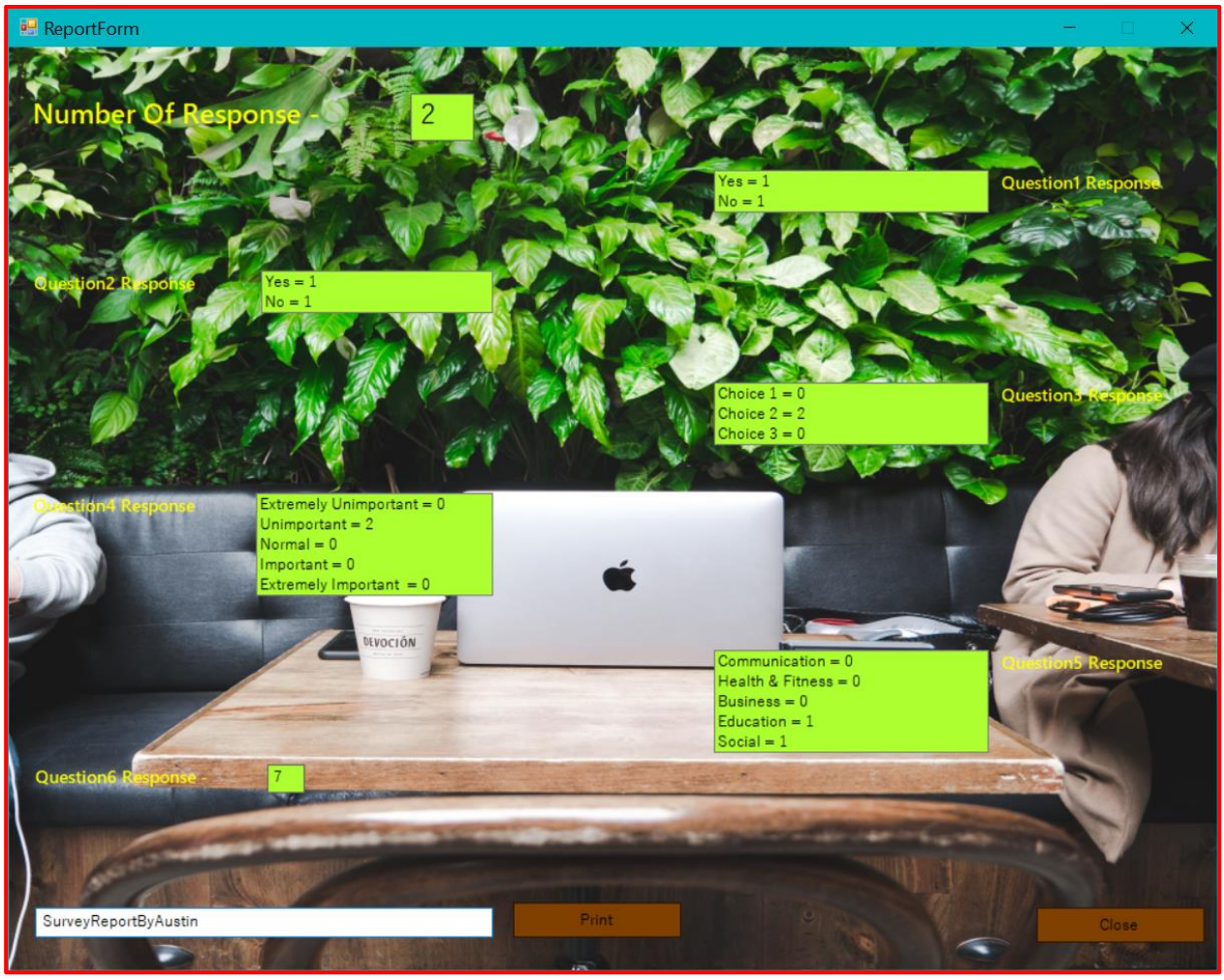


Figure144

After testing

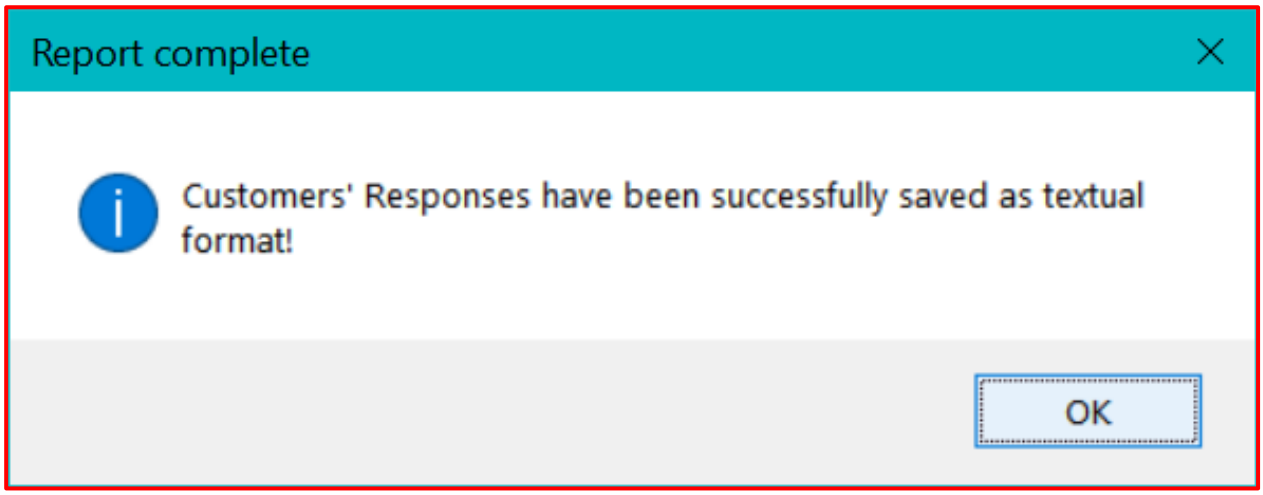


Figure 145

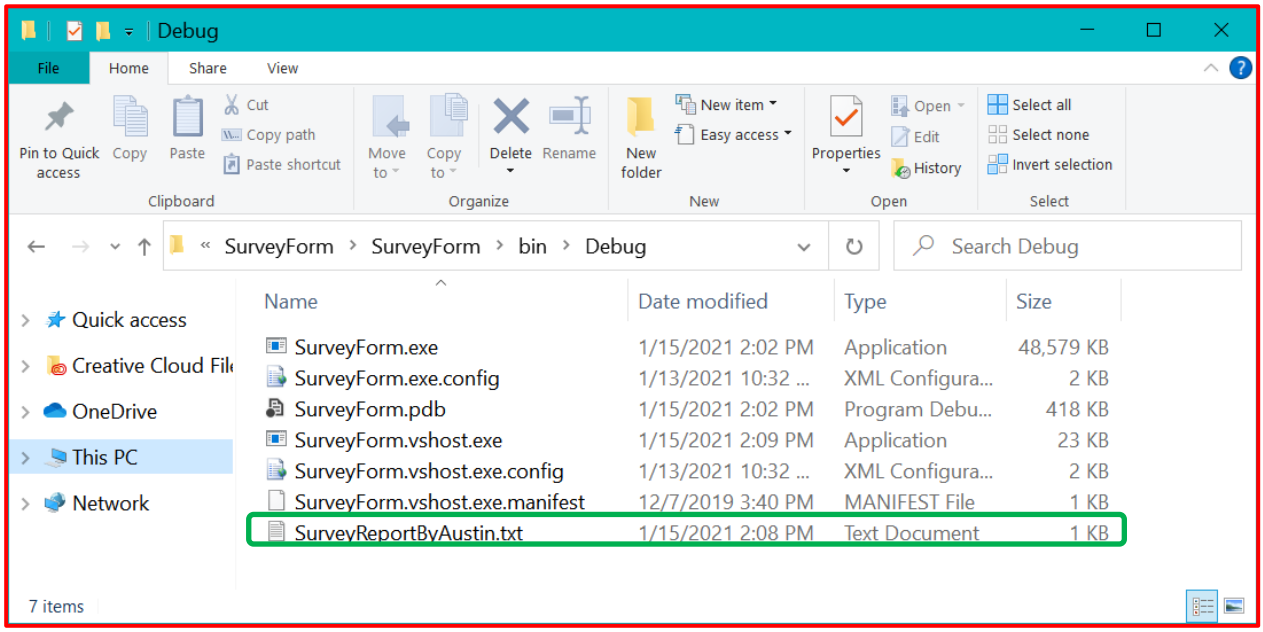


Figure 146

- The file path may vary for each user.

Test Case	Description	Test Procedure	Expected Result	Actual results
13.2	Testing the “Close” button	Hit on “Close” button	The page should close.	The “Report form” page disappears when clicking on “OK” in dialog box

Before testing

Figure 147

After testing

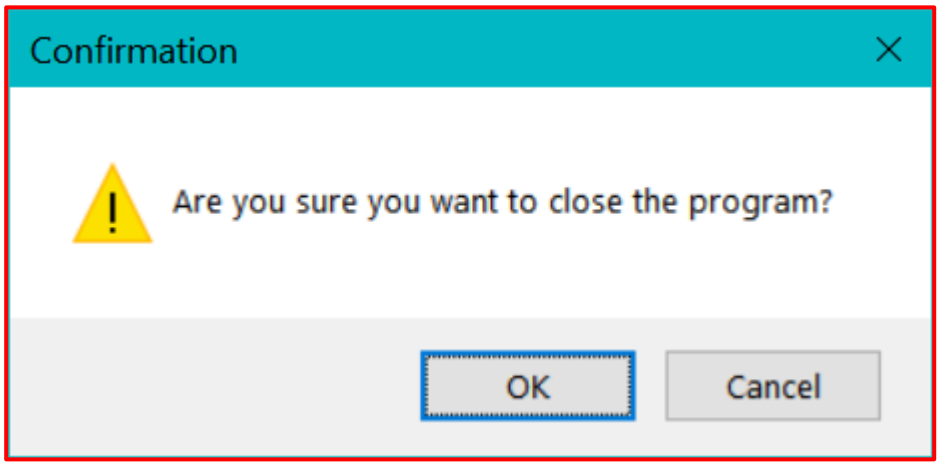


Figure 148

White box testing

Test Case	Description	Test Procedure	Actual results
14.1	Whitebox testing	Highlighting each line of code step-by-step	Table

```
        private void btnregister_Click_1(object sender, EventArgs e)
        {
            ai.adminID = txtid.Text;
            ai.adminName = txtadname.Text;
            ai.adminUsername = txtusername.Text;
            ai.adminPassword = txtpassword.Text;
            ai.adminEmail = txtemail.Text;

            if (txtadname.Text != "" && txtusername.Text != ""&& txtpassword.Text != ""
            && txtemail.Text != "")
            {
                adminDataset.SaveAdmin(ai.adminID, ai.adminName, ai.adminUsername,
                ai.adminPassword, ai.adminEmail);
                MessageBox.Show("The registration has been successful. \n \t Thank
                you.", "Registration successful!
                ",MessageBoxButtons.OK,MessageBoxIcon.Asterisk);
                Auto_ID();
                txtDataEmpty();
            }

            else
            {
                MessageBox.Show("Please completely fill the form! \n \t Thank you.",
                "Registration fail!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                txtDataEmpty();
            }
        }
```

Admin ID	Admin Name	Admin Username	Admin Password	Admin Email	Condition	Database	Output
Admin-001							
	Austin						
		Austin Be					
			AustinBe123				
				Austin217@gmail.com			
					txtadname.Text != "" && txtusername.Text != ""&& txtpassword.Text != "" && txtemail.Text != ""(True)		
Admin-001	Austin	Austin Be	AustinBe123	Austin217@gmail.com		SaveAdmin	
							The registration has been successful. \n \t Thank you.
Admin-002	Austin	Austin Be	AustinBe123	Austin217@gmail.com			
Admin-002							

2.4 Justification of Testing

The program is tested for making positive impact on profitability for the long term. Being bugs caught in the initial stage of software testing, the organization will be cost-effective. Testing delivers secure high-quality products to customers without worries for risks and flaws. Undoubtedly, testing is bound to fulfil the clients' requirements.

The whole program is tested by black-box approach to focus on the input which passes to the program and the output process. The focal point of testing is functionality of the program as well as linkage between one page and another. is mainly tested. Certain buttons are tested whether they show confirmation dialog boxes or work when they are clicked. This type of testing contributes better user experience as it sees a program from a user's perspective.

Admin Registration page is tested by white-box method to identify all the possible paths and internal hidden errors. Furthermore, this method is applied to identify internal security holes and to check the functionality of conditional loops.

The welcome page is tested whether it allows two user options so that the user would not be confused about its main objective.

The Admin dashboard is mainly tested whether it links to the "Admin View List", the "Registration", the "login", the "Admin Create Survey" and the "Admin View Survey" pages. As a consequence, the admin only has to take less time and he would easily go to page without any fail or delay. The peripheral buttons such as "Close" and "Refresh" are also tested by black box method.

The "Admin Create Survey" plays a key role in black box testing as it is one of the central portions of the program. It has to be well-tested to reach the main target. Whether it handles the errors is tested for generating the complete and effective survey. "Null reference exception" dialog box must be thrown when inadequate fill-in occurs. As a consequence, the input data will never be saved as null.

The "Admin View Survey" is tested for whether the additional functions like update and delete process work. The "Report" button is tested whether there is a linkage to "Report form" page.

Another target of the program is analyzing the responses so that the “Report form” page is tested. As a consequence, it ensures the admins will successfully report the survey as a textual format.

For the “Customer Dashboard”, embedded functions are chiefly approached by black-box method for the sake of user experience. Table cells are also tested so that the customers would be able to select and respond surveys. “History” button is tested in order that the clients would easily go to “History” page and review their responses without any delay.

“Customer Answer Survey” is tested for whether it performs saving responses efficiently. The “Customer History” page is only tested for its embedded function in order to fulfil the perfect user experience.

The “Admin registration” page is verified by white-box testing for the flow of input-output, security, error handling and reusability. Step-by-step approaches make sure the code infrastructure has no fault.

2.5 Demonstration of Program

The initial page will ask user whether he is an admin or customer. At first, let's suppose user as an administrator. Click on anywhere on the picture labelled by "Admin".

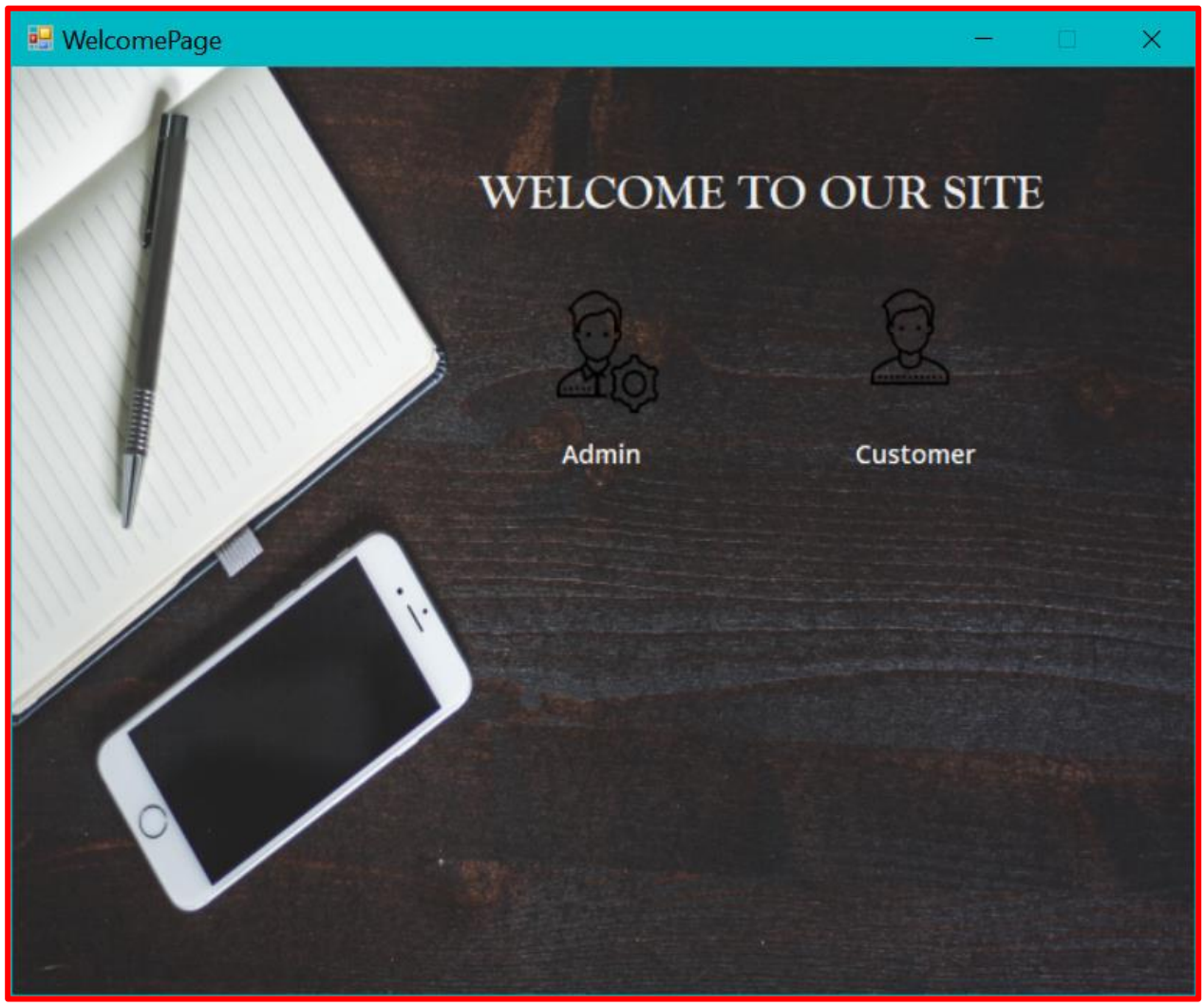


Figure 149

Only if the admin fills the form completely, the registration process will be accomplished. Nevertheless, there is no need for registered admin to do so. Instead, he has to click on the link-label to login.

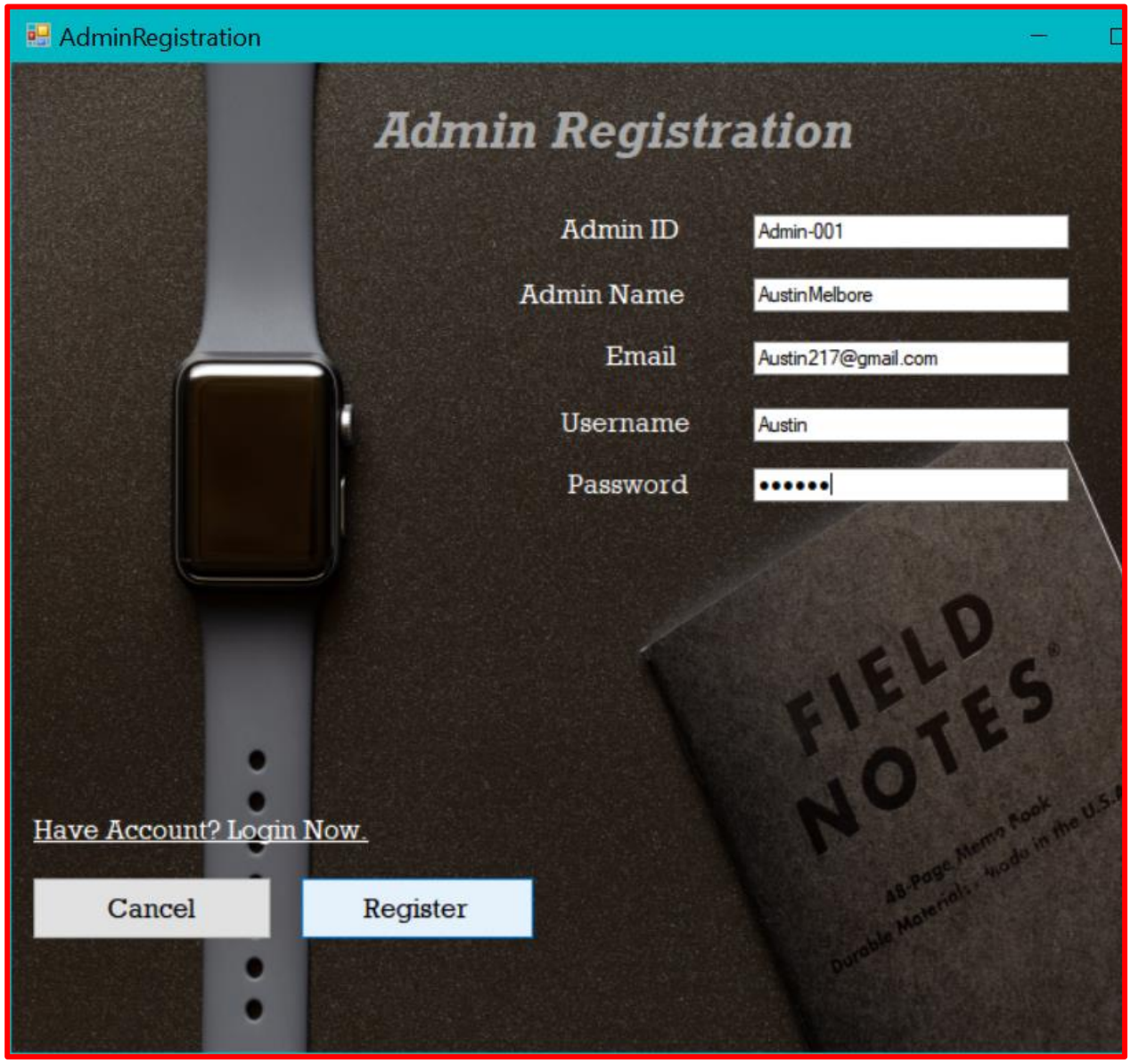


Figure 150

As soon as the registration procedure has been successful, the registered admin needs to login in for the sake of getting to the Admin Dashboard page.

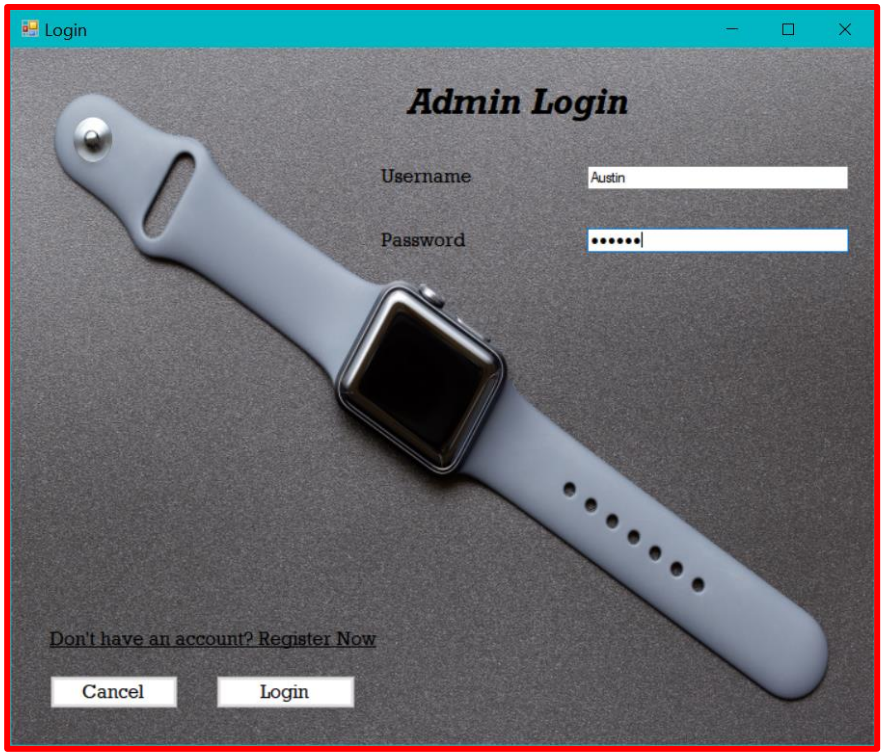


Figure 151

As soon as the admin login process has been accomplished, the Administrative Dashboard will emerge. This main page supports admins in conducting surveys and its corresponding functions.

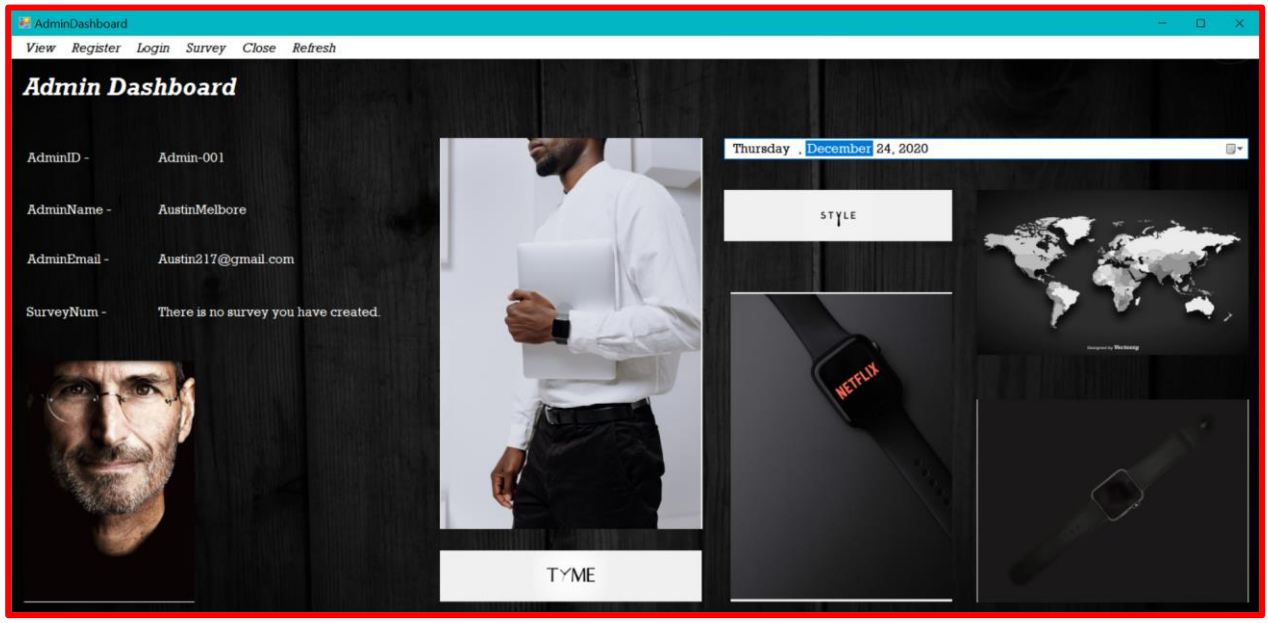


Figure 152

Clicking on “View” on the menu bar of the admin dashboard, the number of surveys of respective admin and the registered admins can be observed in the administrative list page.

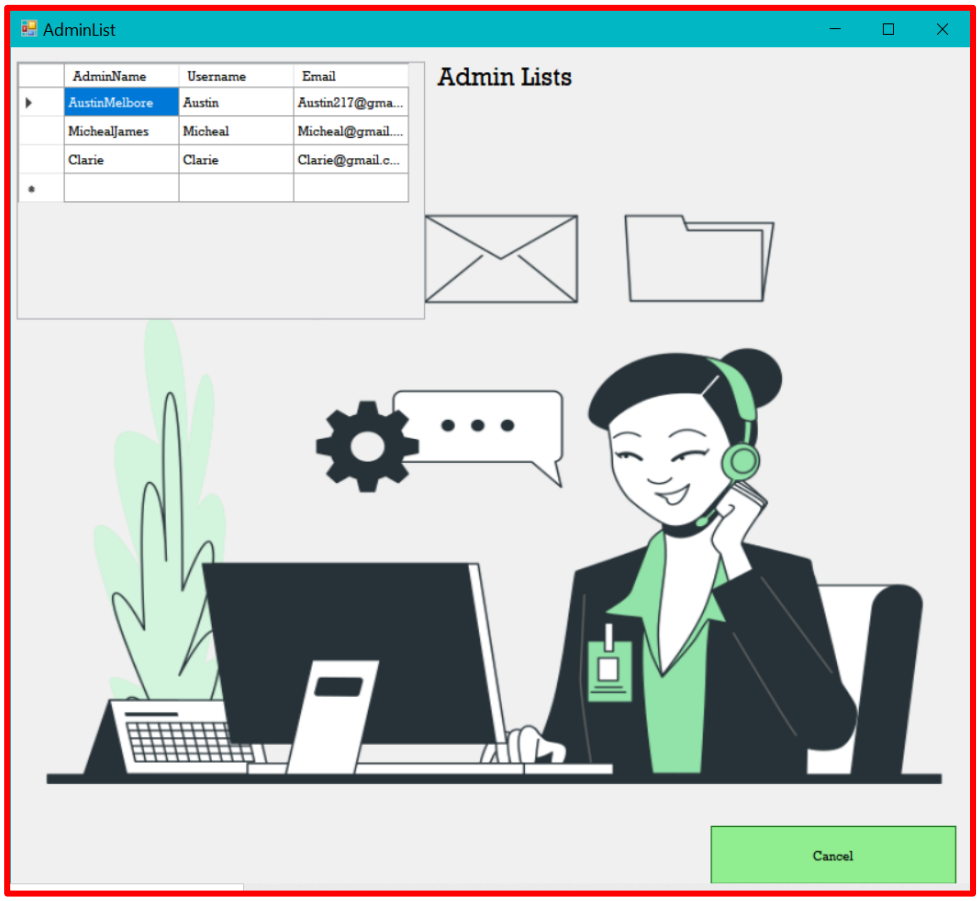


Figure 153

Survey button is created for two options for creating surveys and viewing them. The first button is linked to the “Admin Create Survey” page while the second is to “Admin View Survey” page.

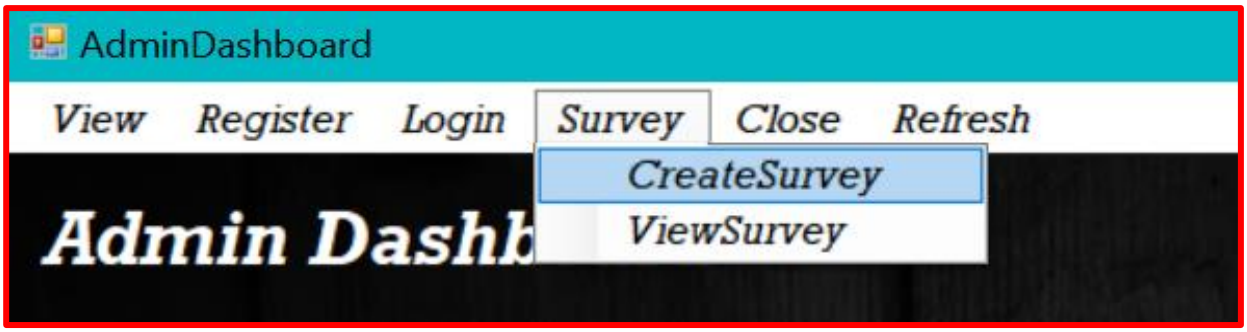


Figure 154

The admin only needs to fill the survey name and six survey questions and hit the “Confirm” button.

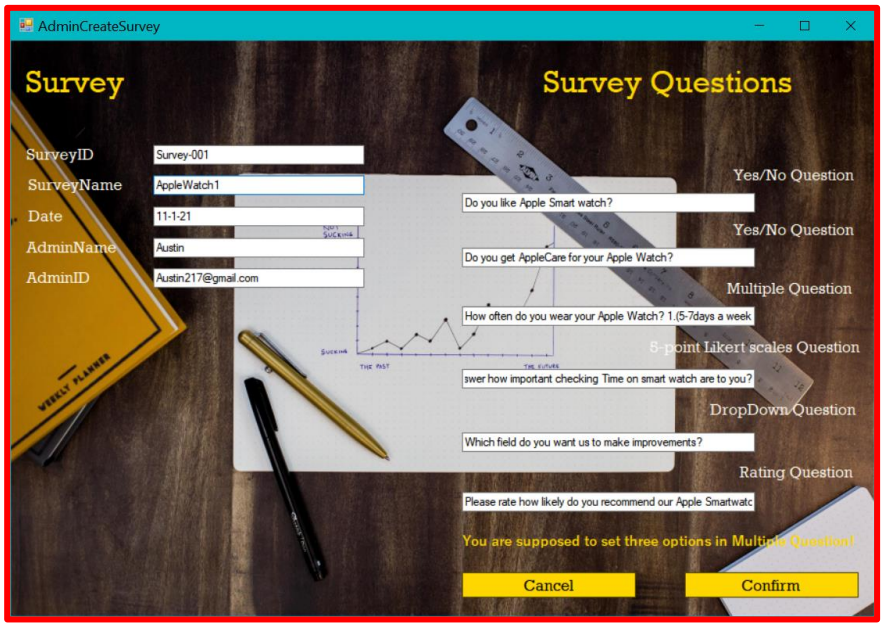


Figure 155

In this Admin View Survey page, three major functions can be performed only by clicking on the table cell of what the admin would like to make changes. “Refresh” button is designed for observing the new changes.

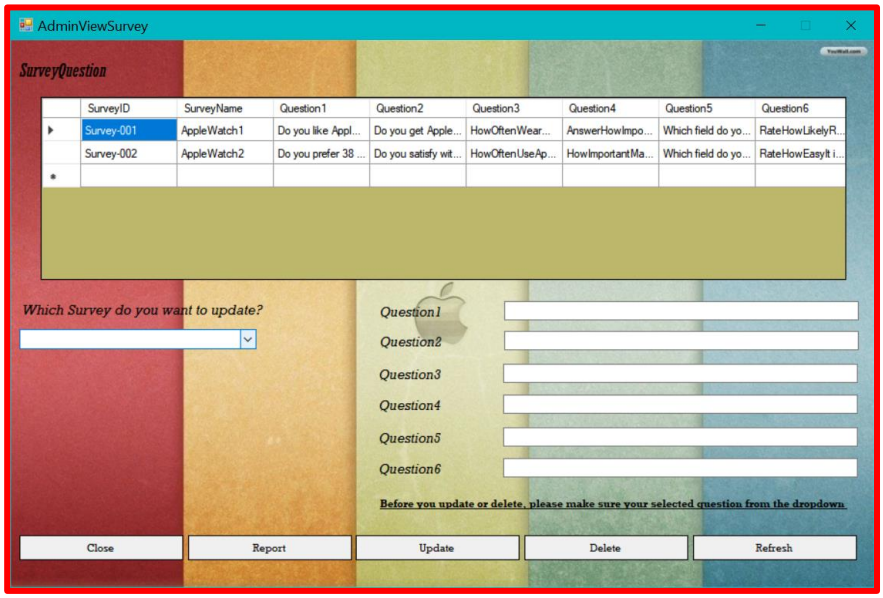


figure 156 before clicking

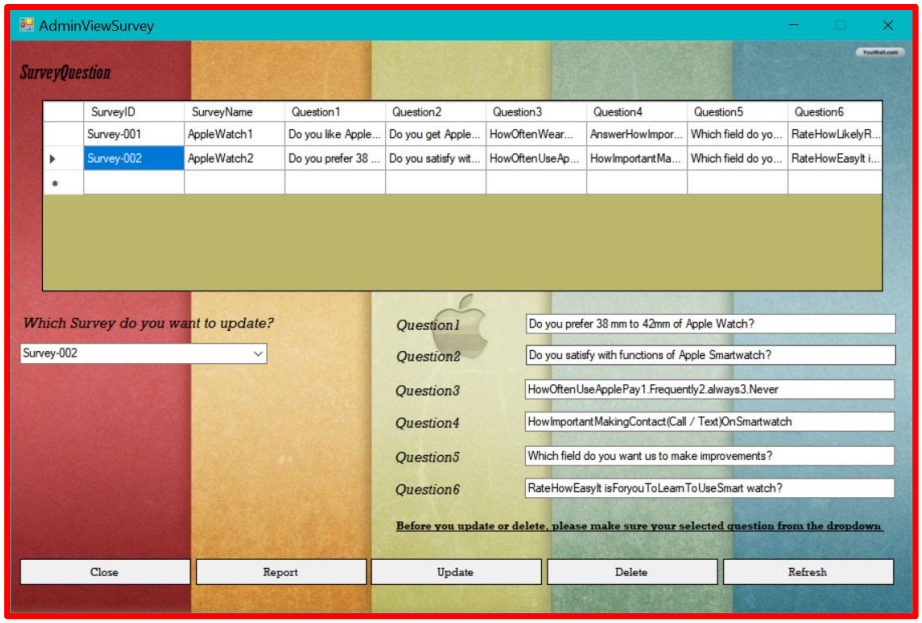


Figure 157 after clicking

After clicking on the “Report” button in the previous page, the Report form page comes out to save the responses as a textual format. Don’t forget to name the report in the textbox.

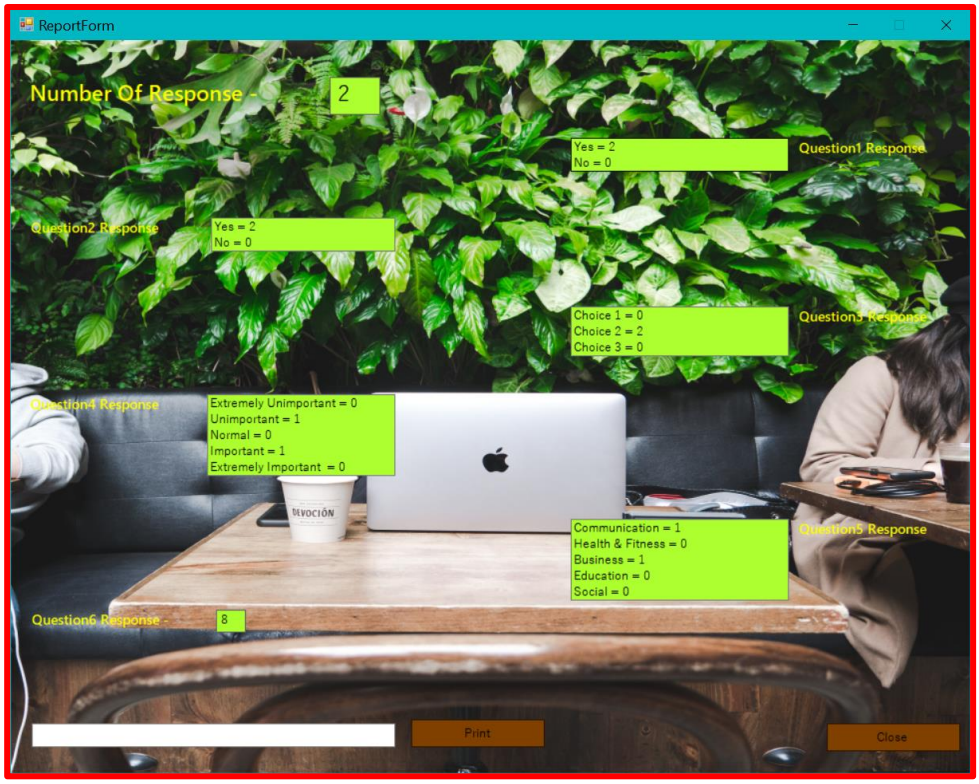


Figure 158

The above-mentioned procedures are perfect for admins but the followings are for customers.

Fill in the textboxes. After that, clicking on “Login” allows to go to the customer dashboard. To accomplish this process, the customer needs to have registered first. Clicking on link-label offers to registration page.

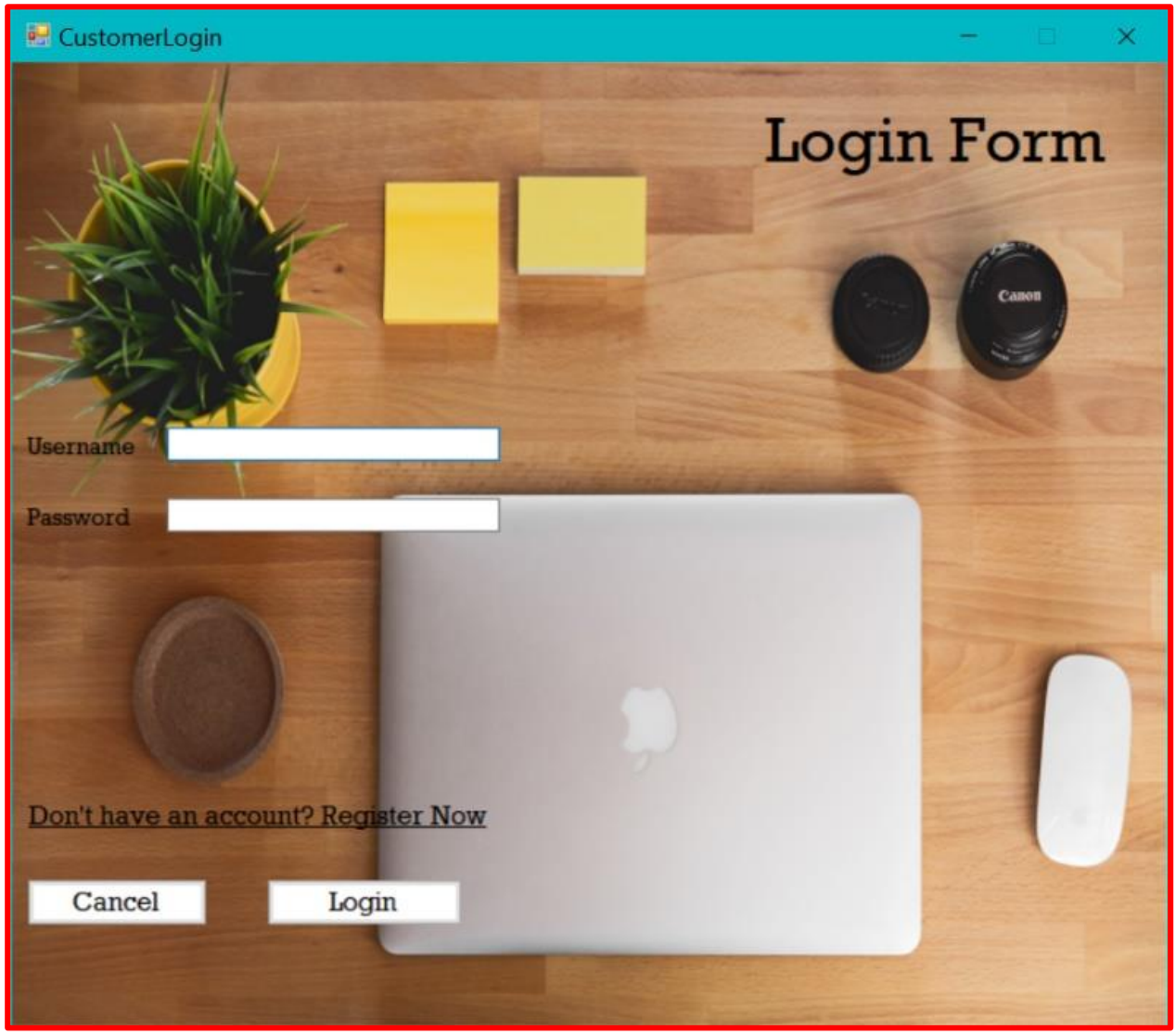


Figure 159

The customer must fill up the form completely. Only after the customers have read the term policy and enabled the checked box, register button enables to work. Registration process will not be successful when completely filled in.

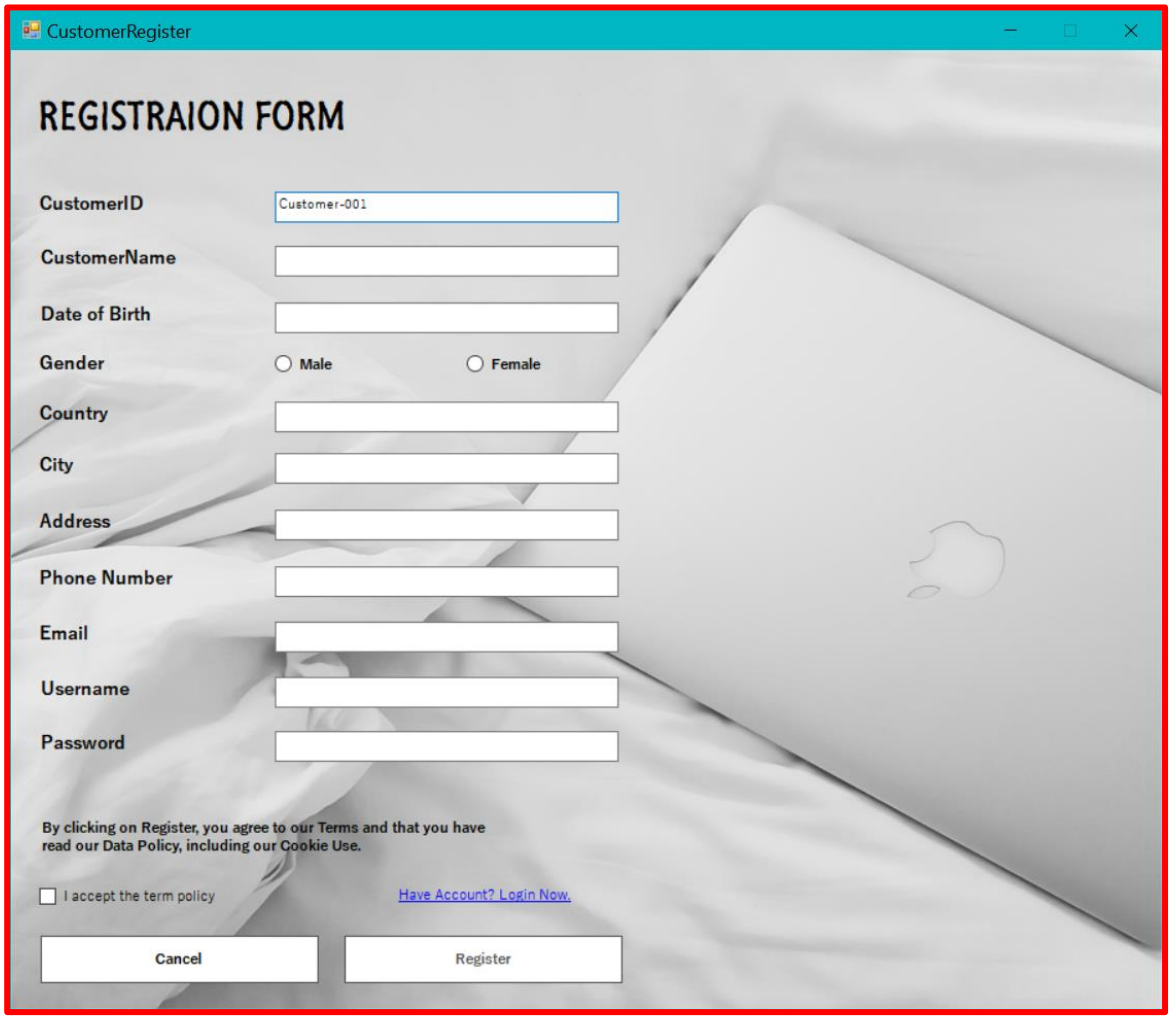


Figure 160

The significant feature includes only in this page unlike others pages as soon as the client sees the page, the survey table won't be shown. According to the yellow instruction, he must click on the picture to continue for responding. The desired survey can be selected by clicking on the cell in table. And then, the survey ID and admin name of the selected survey will appear on the space. Having selecting, click on the picture beside "Yes" label.

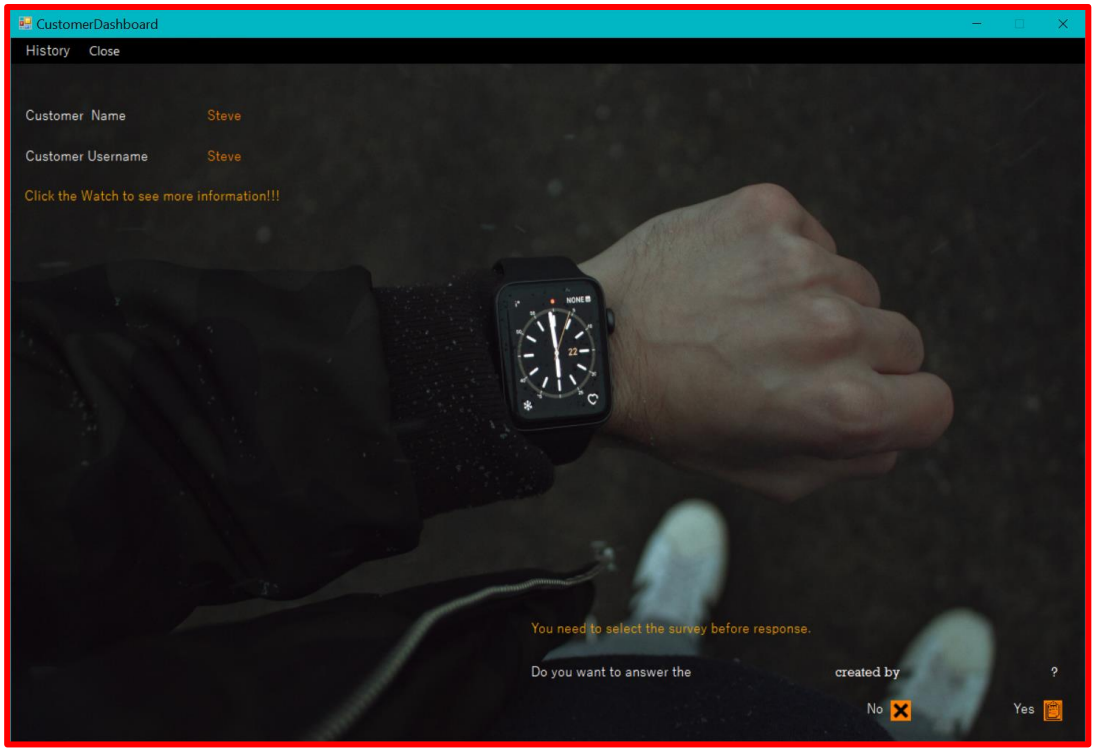


Figure 161 before clicking on the picture

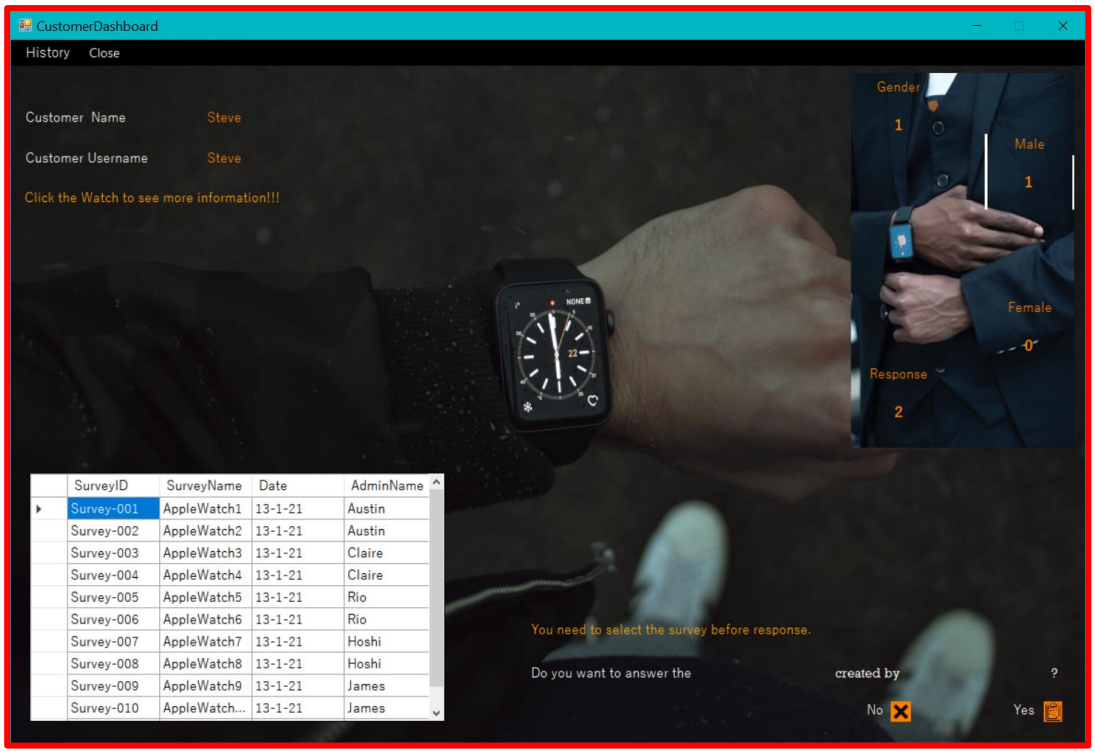


Figure 162 before clicking on the picture

The customer must answer all the six questions and then the response will be saved in the database after clicking on “Save” button. Then, click on “Back” to return back to the dashboard.

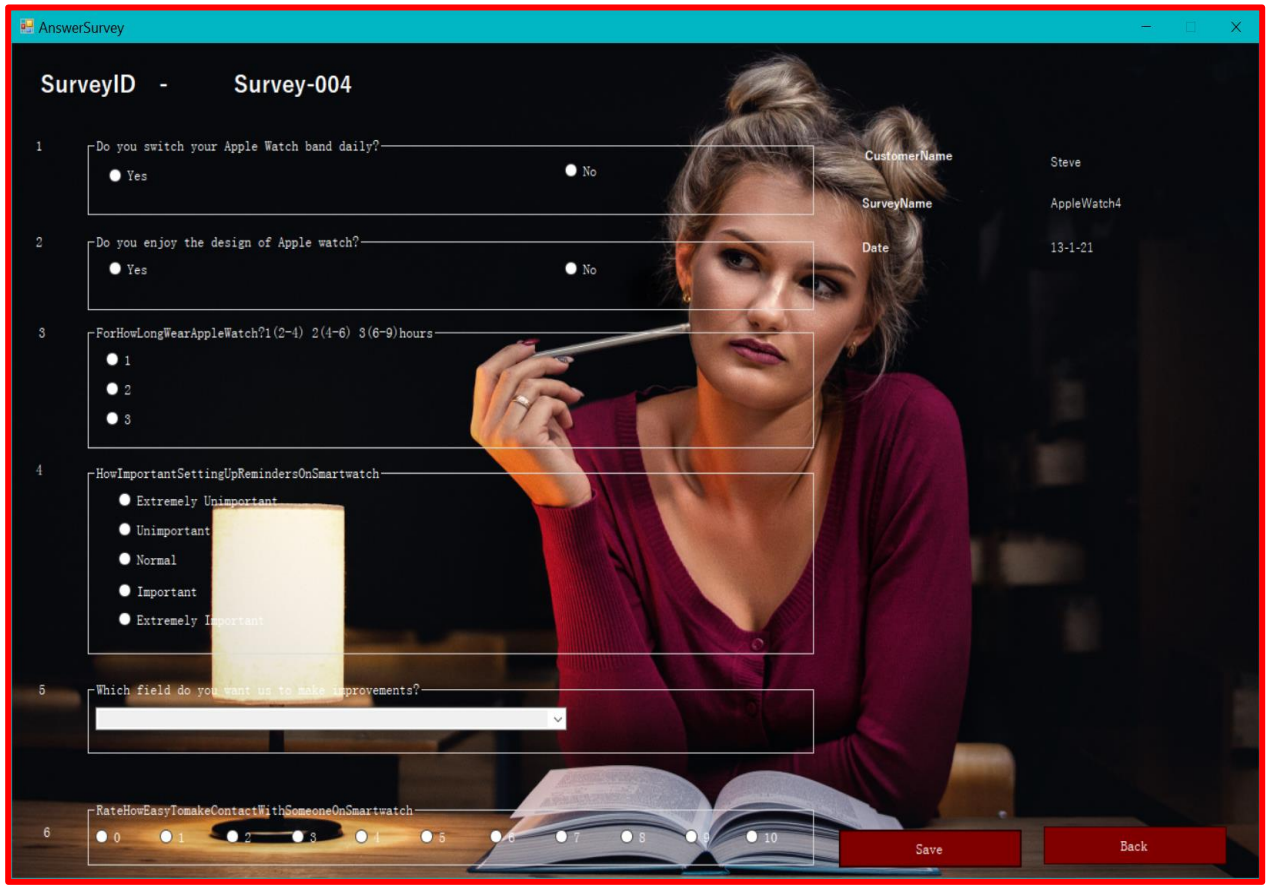


Figure 163

Clicking on the picture will make response table materializes below the picture. Responses can be explored with search box. The red cross sign at the top-right of the page is to close it. Customer should read the red-colored instruction first so that he would be able to perform searching process.

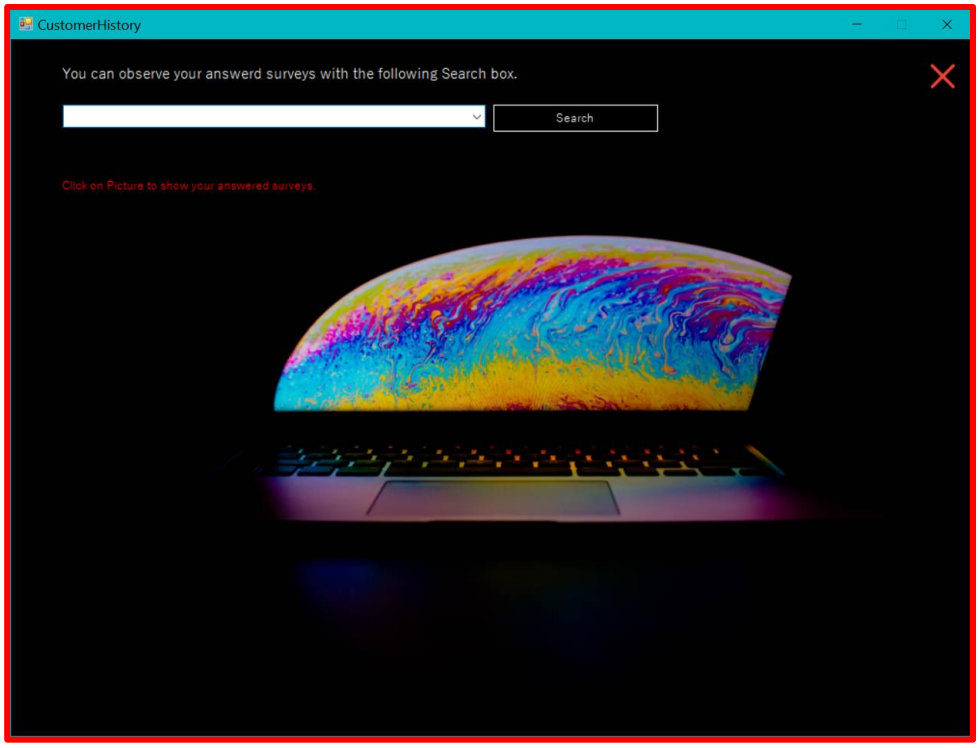


Figure 164 before clicking on the picture

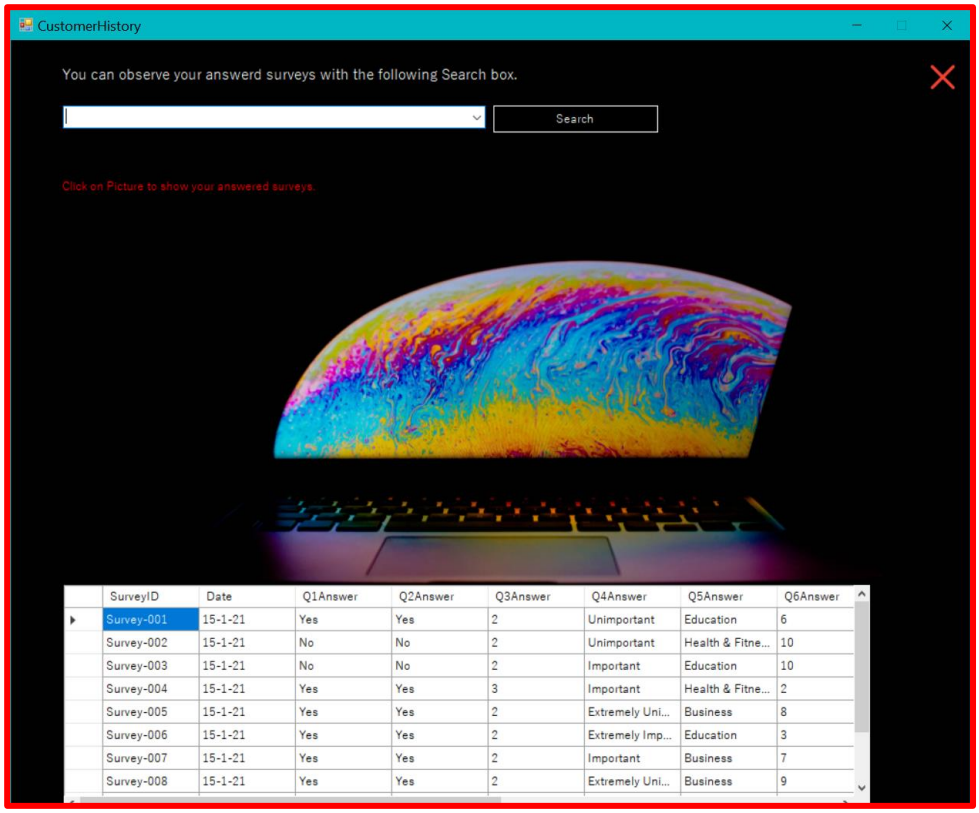
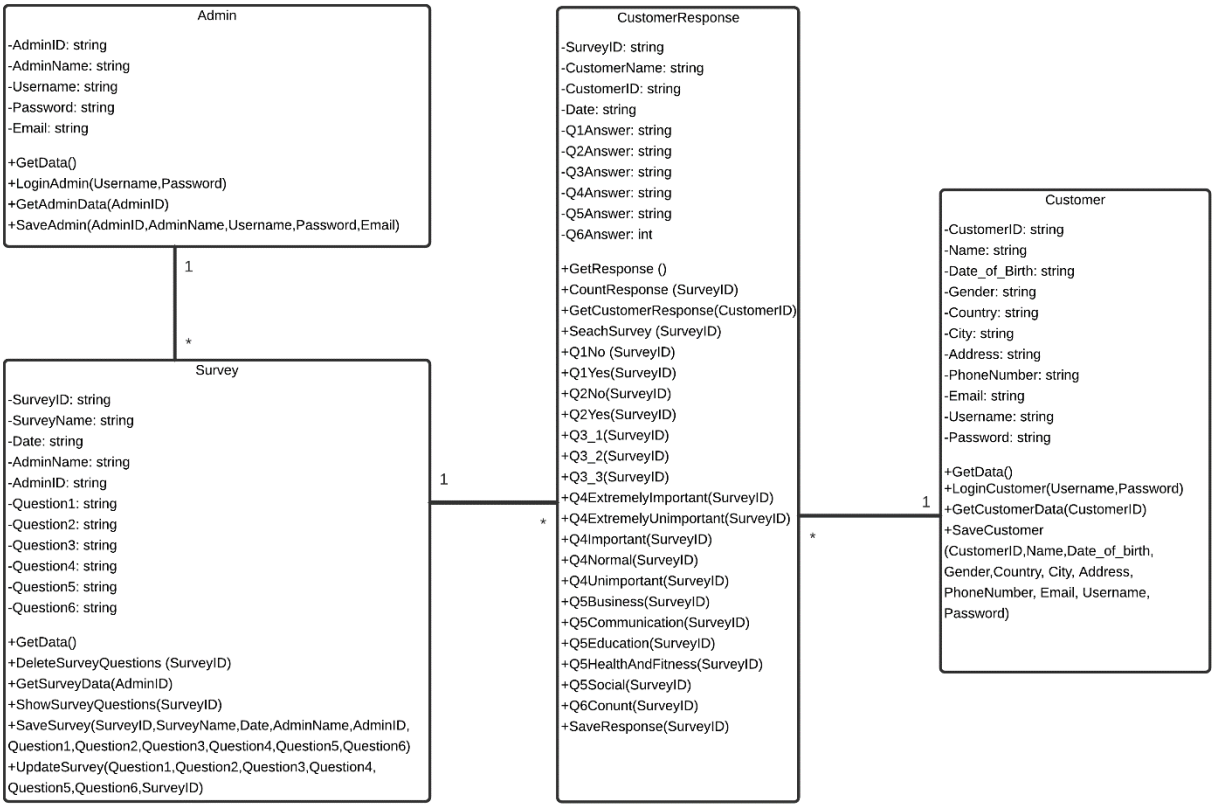


Figure 165 after clicking on the picture

3. Task – 3

3.1 Class Diagram



3.2 Class Description

Class name	Admin
Attributes	- AdminID: string - AdminName: string - Username: string - Password: string - Email: string
Operation	+ GetData() + LoginAdmin(Username, Password) + GetAdminData(AdminID) + SaveAdmin(AdminID, Admin Name, Username, Password, Email)
Description	Admin class was constructed in order to accumulate the particulars as regards administrators

Class name	Customer
Attributes	- CustomerID: string - Name: string - Date_of_Birth: string - Gender: string - Country: string - City: string - Address: string - PhoneNumber: string - Email: string - Username: string - Password: string
Operation	+ GetData() + LoginCustomer(Username, Password) + GetCustomerData(CustomerID) +SaveCustomer(CustomerID,Name,Date_of_birth, Gender,Country, City, Address, PhoneNumber, Email, Username, Password)

Description	Customer class was created for saving the customers' particulars

Class name	**Survey**
Attributes	-SurveyID: string -SurveyName: string -Date: string -AdminName: string -AdminID: string -Question1: string -Question2: string -Question3: string -Question4: string -Question5: string -Question6: string
Operation	+GetData() +DeleteSurveyQuestions (SurveyID) +GetSurveyData(AdminID) +ShowSurveyQuestions(SurveyID) +SaveSurvey (SurveyID, SurveyName, Date, AdminName, AdminID, Question1, Question2, Question3, Question4, Question5, Question6) +UpdateSurvey(Question1, Question2, Question3, Question4, Question5, Question6, SurveyID)
Description	Survey class was designed to save surveys

Class name	**CustomerResponse**
Attributes	-SurveyID: string -CustomerName: string -CustomerID: string

	-Date: string -Q1Answer: string -Q2Answer: string -Q3Answer: string -Q4Answer: string -Q5Answer: string -Q6Answer: int
Operation	+GetResponse () +CountResponse (SurveyID) +GetCustomerResponse (CustomerID) +SeachSurvey (SurveyID) +Q1No (SurveyID) +Q1Yes (SurveyID) +Q2No (SurveyID) +Q2Yes (SurveyID) +Q3_1(SurveyID) +Q3_2(SurveyID) +Q3_3(SurveyID) +Q4ExtremelyImportant (SurveyID) +Q4ExtremelyUnimportant (SurveyID) +Q4Important (SurveyID) +Q4Normal (SurveyID) +Q4Unimportant (SurveyID) +Q5Business (SurveyID) +Q5Communication (SurveyID) +Q5Education (SurveyID) +Q5HealthAndFitness (SurveyID) +Q5Social (SurveyID) +Q6Conunt (SurveyID) +SaveResponse (SurveyID)
Description	CustomerResponse class was built in order to save survey answers responded by customers

3.3 Justification of Class Diagram

Class diagrams act as building blocks in object-oriented modeling describing a variety of objects in a system. In this program, four class diagrams are constructed to model the static perspective of a program. The top portion of each diagram contains the class name. The middle part includes encapsulated attributes by making the variables as private and exposing the property for the entrance of private data which would be public. The bottom partition is composed of operations for class to execute the program.

The Admin class is designed with private operations and attributes providing data hiding and security. The operations are to insert administrators' information into database when they are invoked. On top of that, "Login Admin" operation is created to check the input and allow admins only if the input data match with ones in database. As one admin can create many surveys and one survey should have only one admin, the Admin class links to the Survey class by means of one-to-many relationship.

The Survey class is built for storing the input concerning surveys with encapsulated attributes. In this class, the operations are developed for updating and deleting surveys as well as refreshing the page. "Save Survey" operation, main features of program, is to transfer the surveys to the database. Moreover, the specific operation "Show Survey Questions" is to exhibit only survey questions. As one survey can have many responses but one response must have only one survey, the "Survey" class links to the "Customer Response" class by means of one-to-many relationship.

The Customer Response class is constructed for collecting the answers and adding them to database. There is one operation for each choice of survey question. For the rating question, the average value will be calculated for each survey by the "Count Response". The "Search Survey" operation is to explore surveys easily. The attributes are string data type with the exception of "Q6Answer". This is because the "Count Response" operation gets only integer data type to determine the average. As one customer can answer many responses but one response must be answered by only one customer, the "Survey" class links to the "Customer Response" class by means of one-many relationship.

The customer class is much similar to the admin class. In this class, the "Save Customer" operation is designed for adding clients' information into database instead of admin's when they are invoked. "Login Customer" operation examines whether the customer input data match with ones in database. And then, allow login process only when the correct data are inputted.

3.4 User Manual.

1. Welcome page

Welcome page allows two options for admins and customers. Choose your role to go to the respective pages.

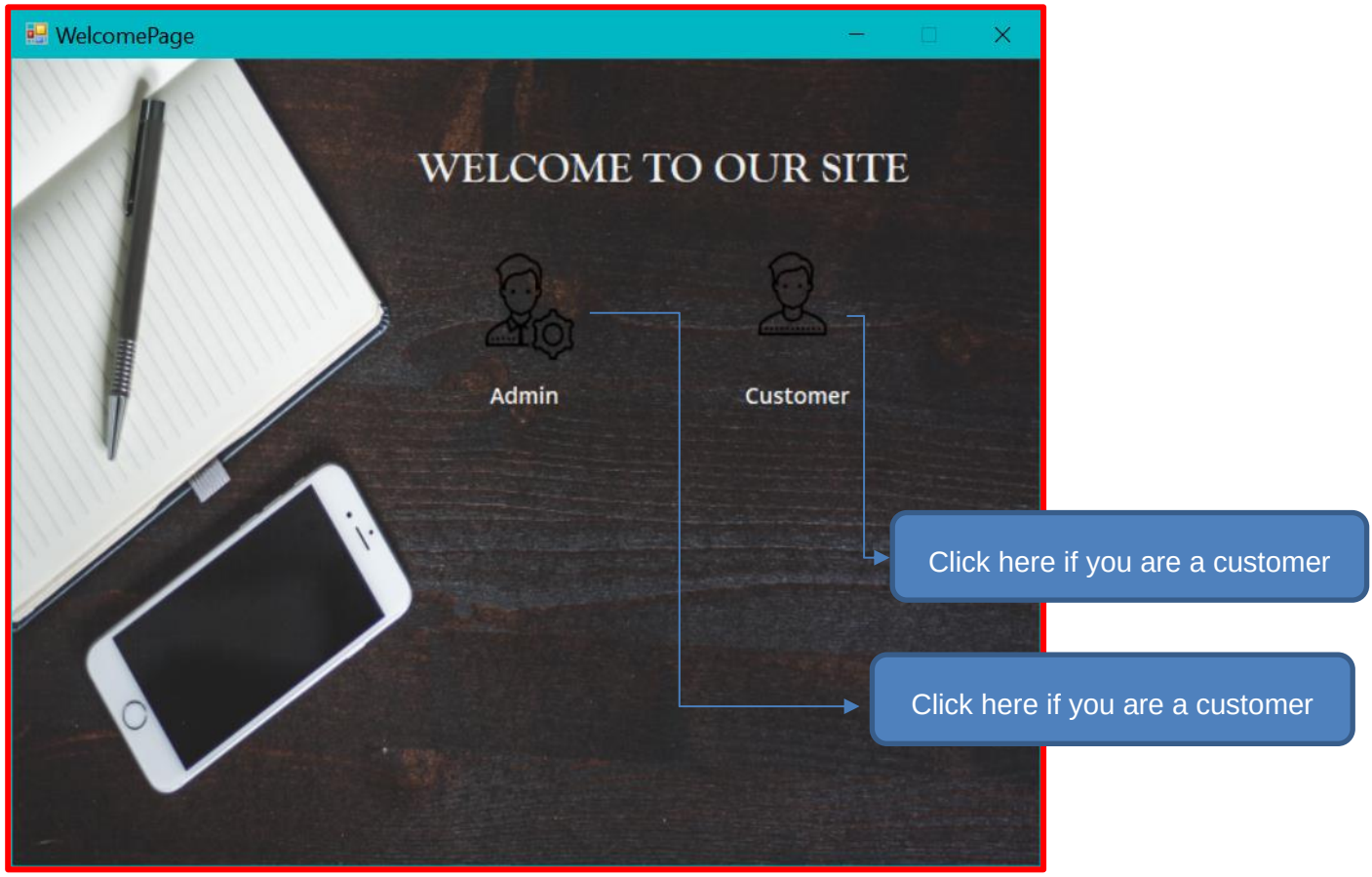


Figure 166

2. Admin login page

The admin can login with valid username and password after registration. However, the admin has to register first if he is a new user.

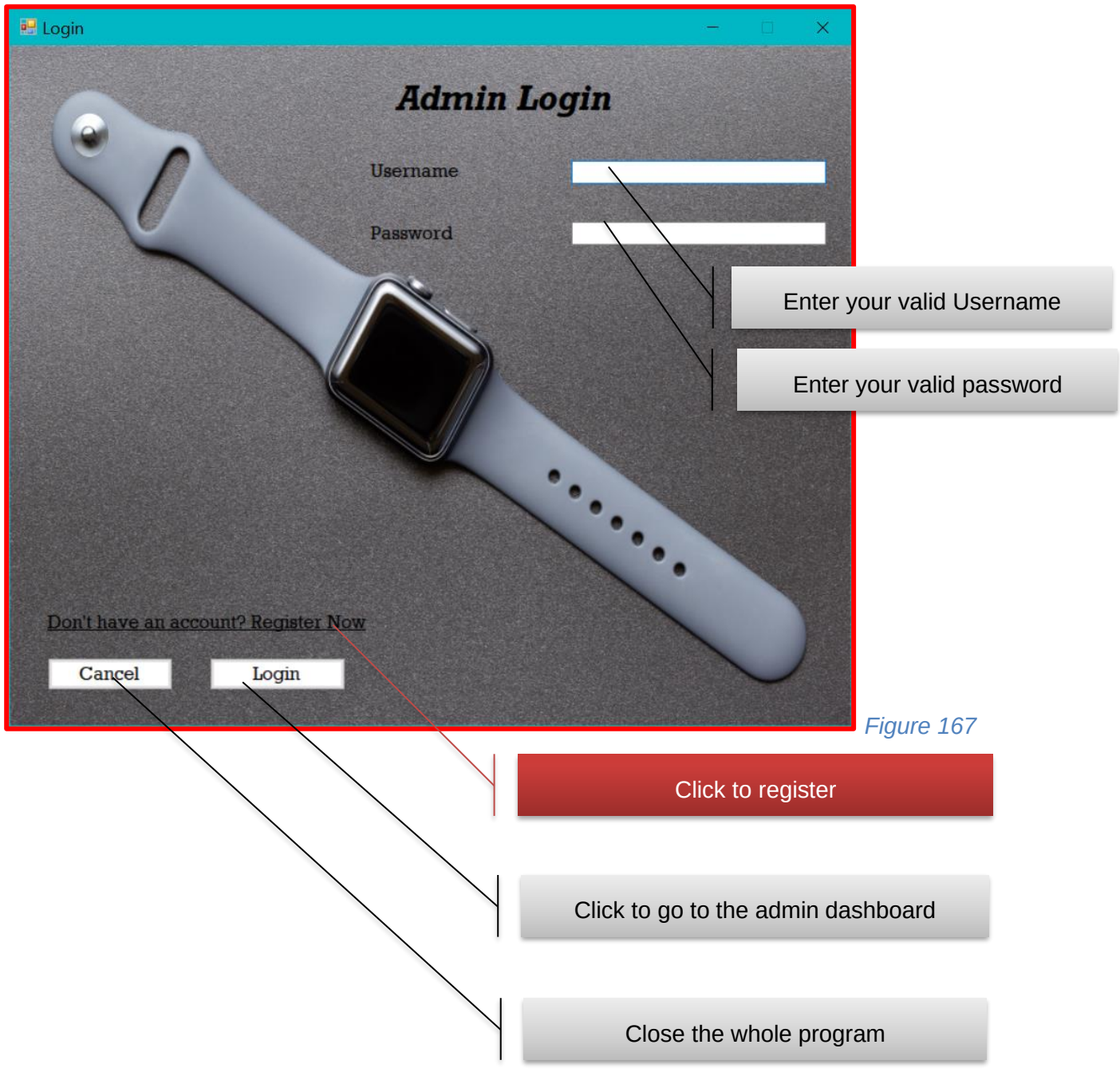


Figure 167

3. Admin Registration page

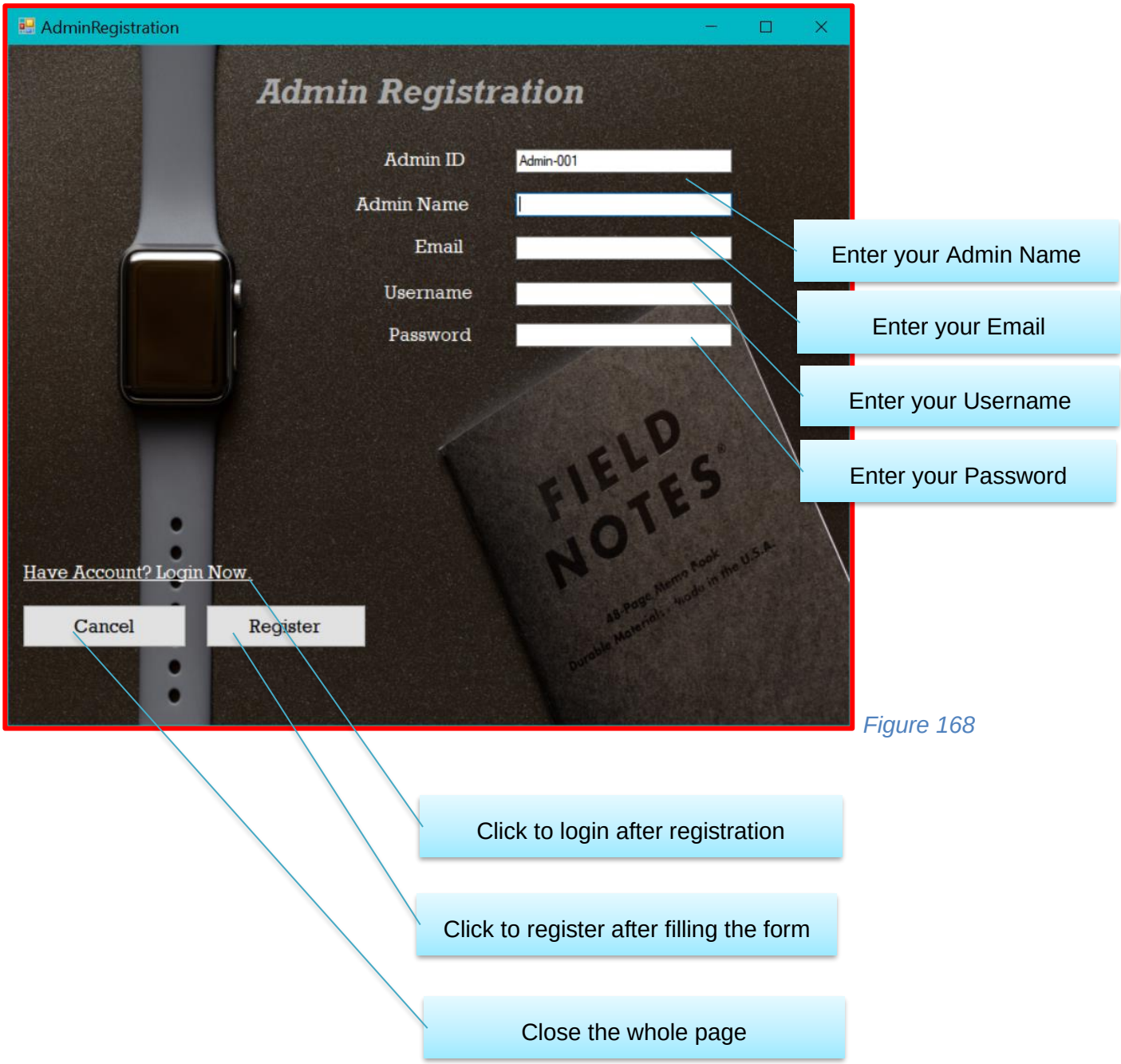


Figure 168

4. Admin Dashboard page

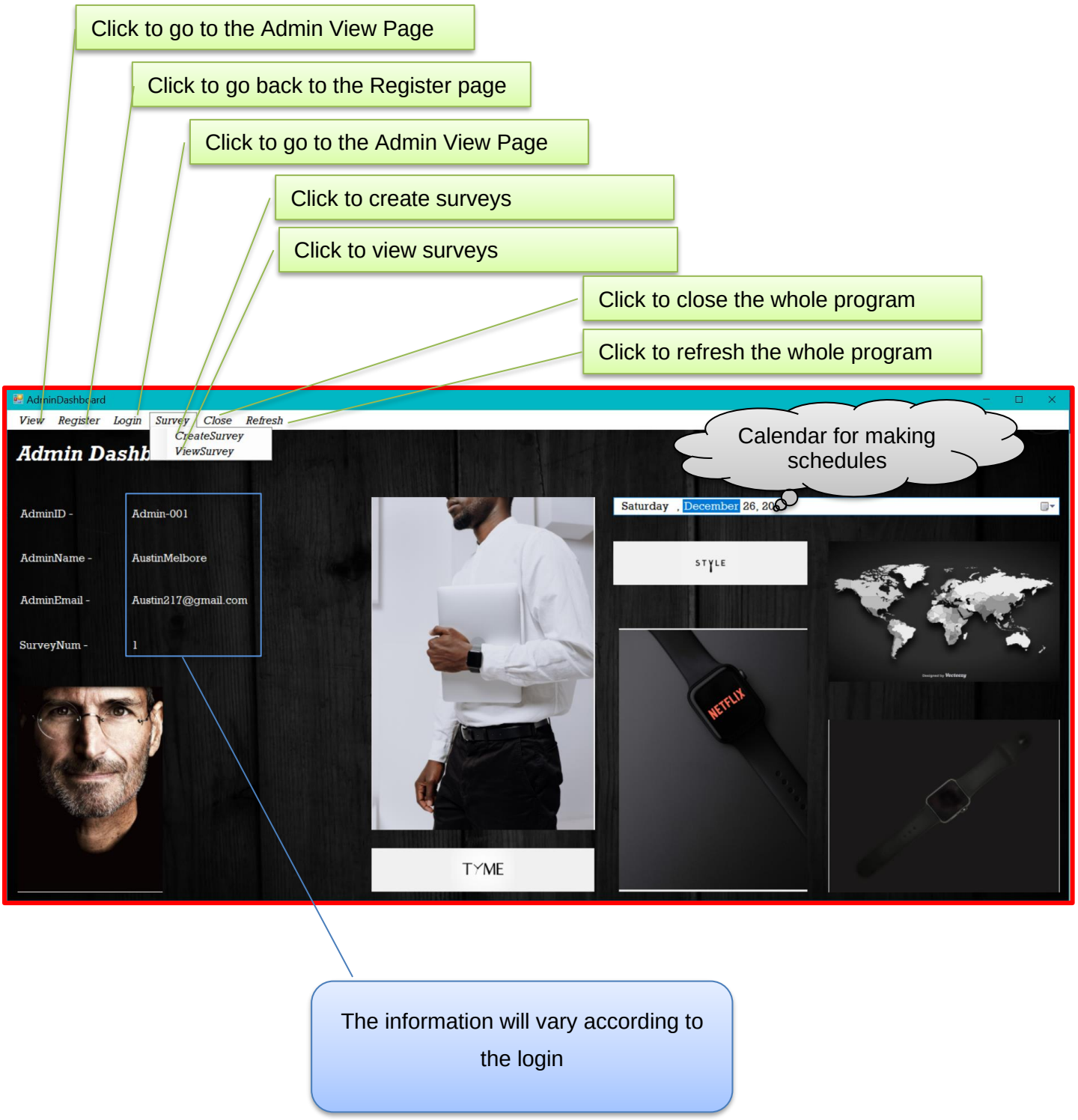


Figure 169

5. Admin view list page

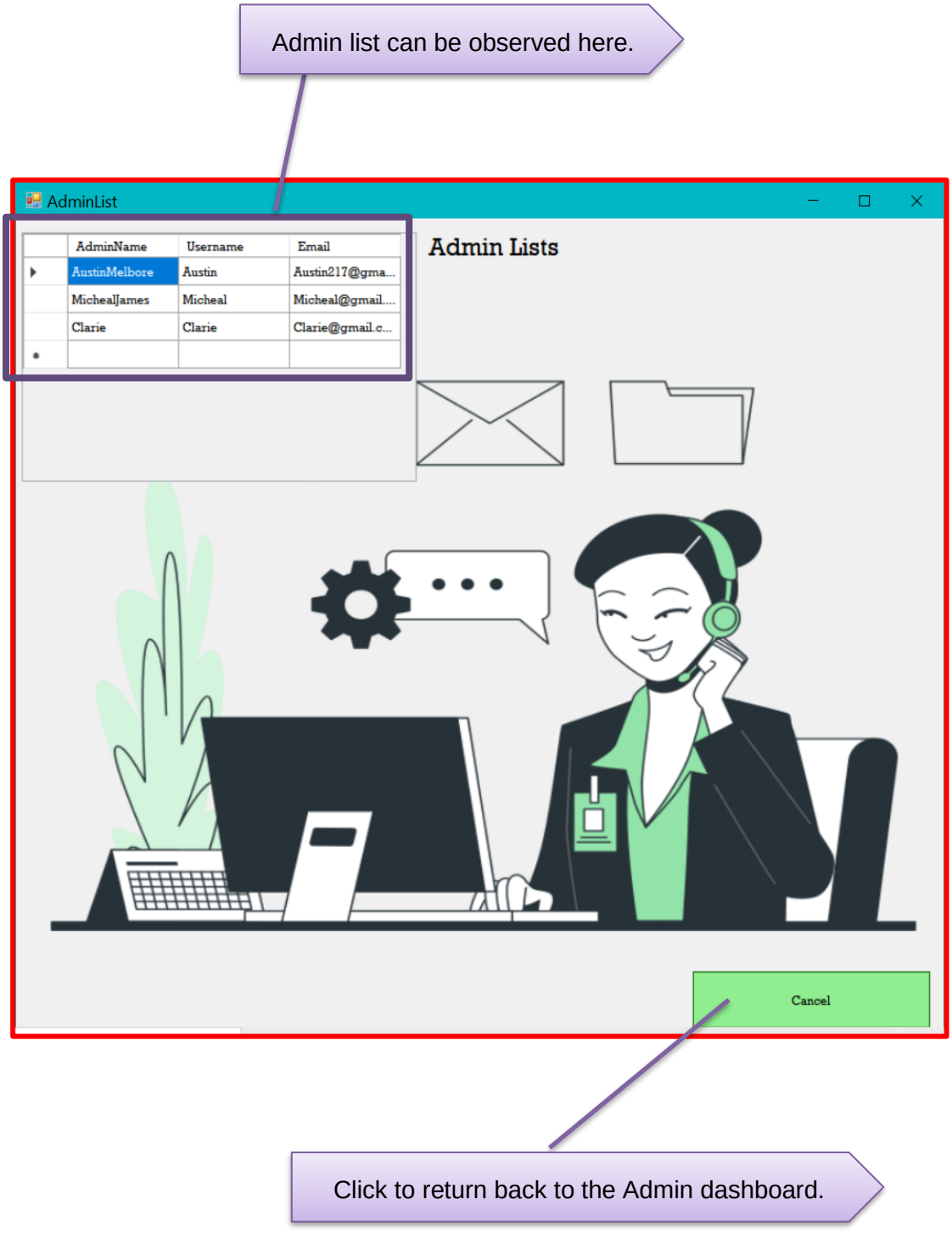
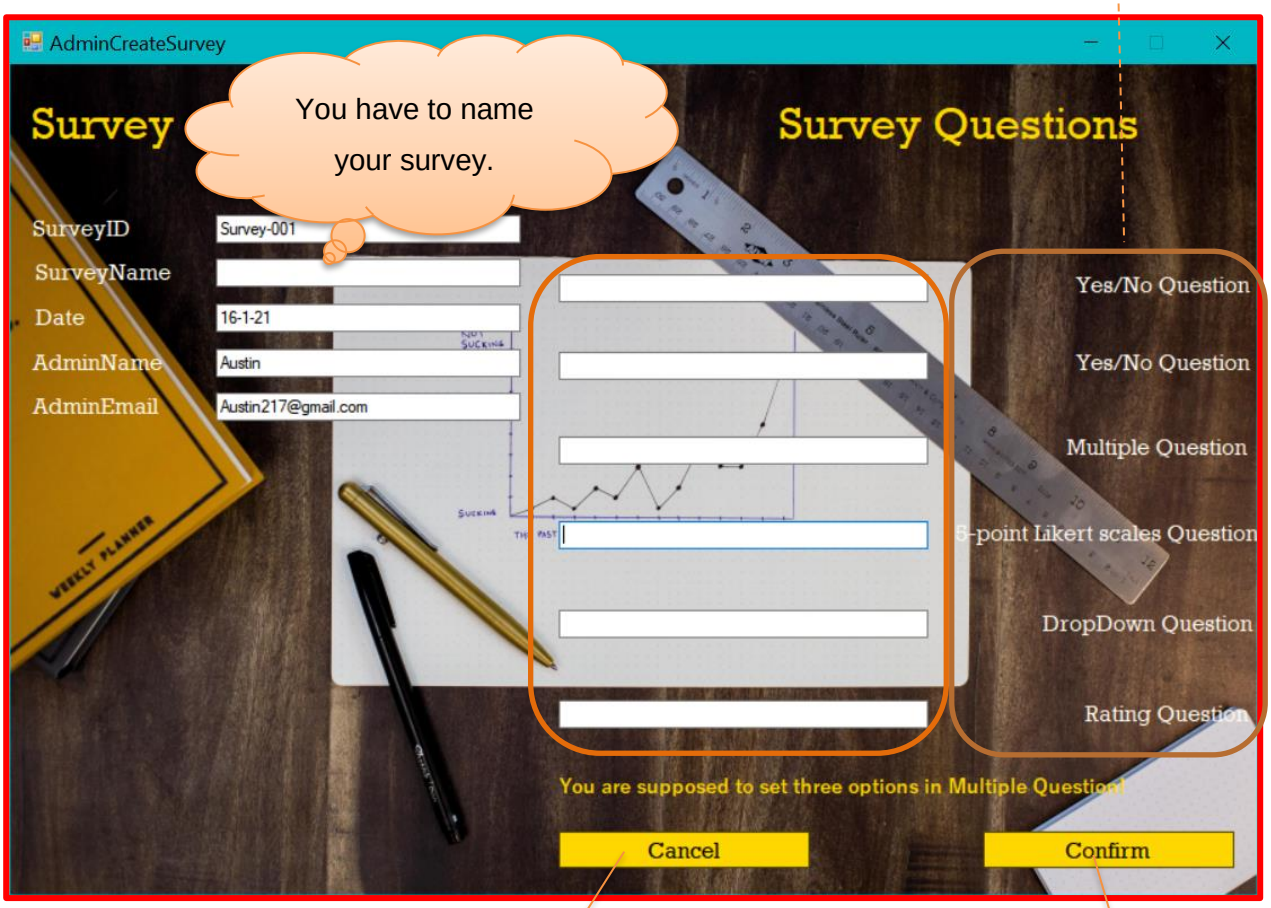


Figure 170

6. Admin Create Survey page

Figure 171



7. Admin view survey page

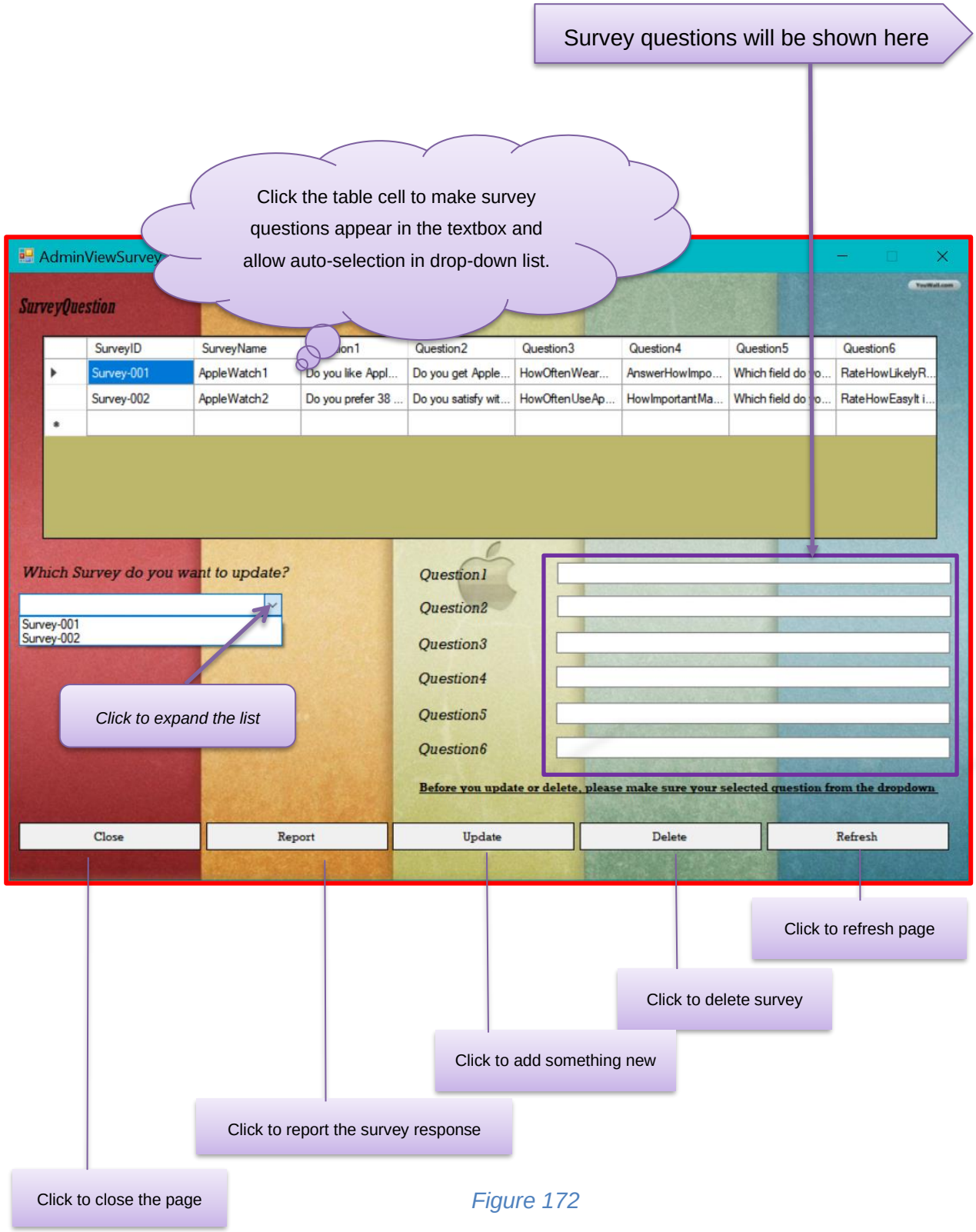
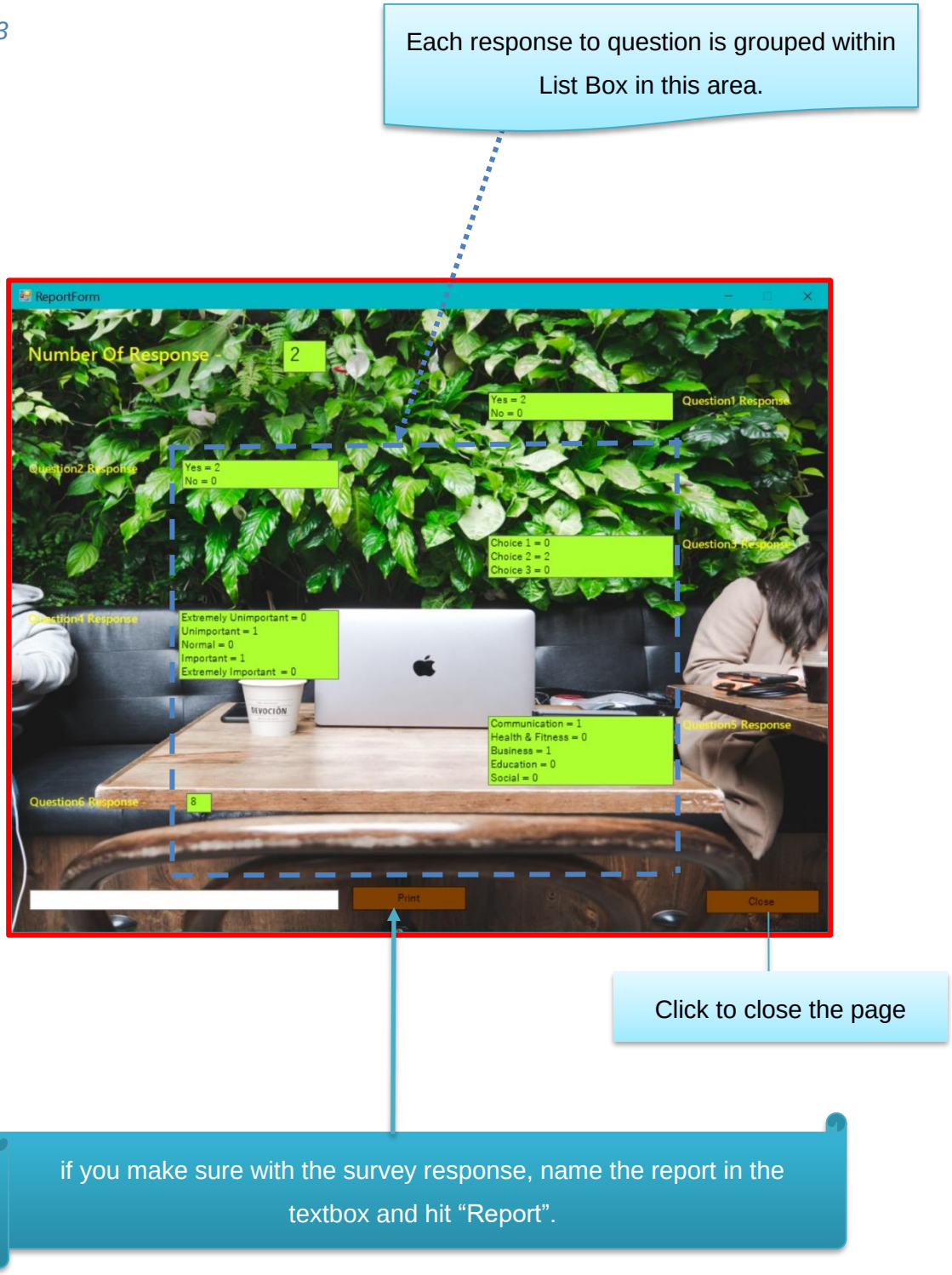


Figure 172

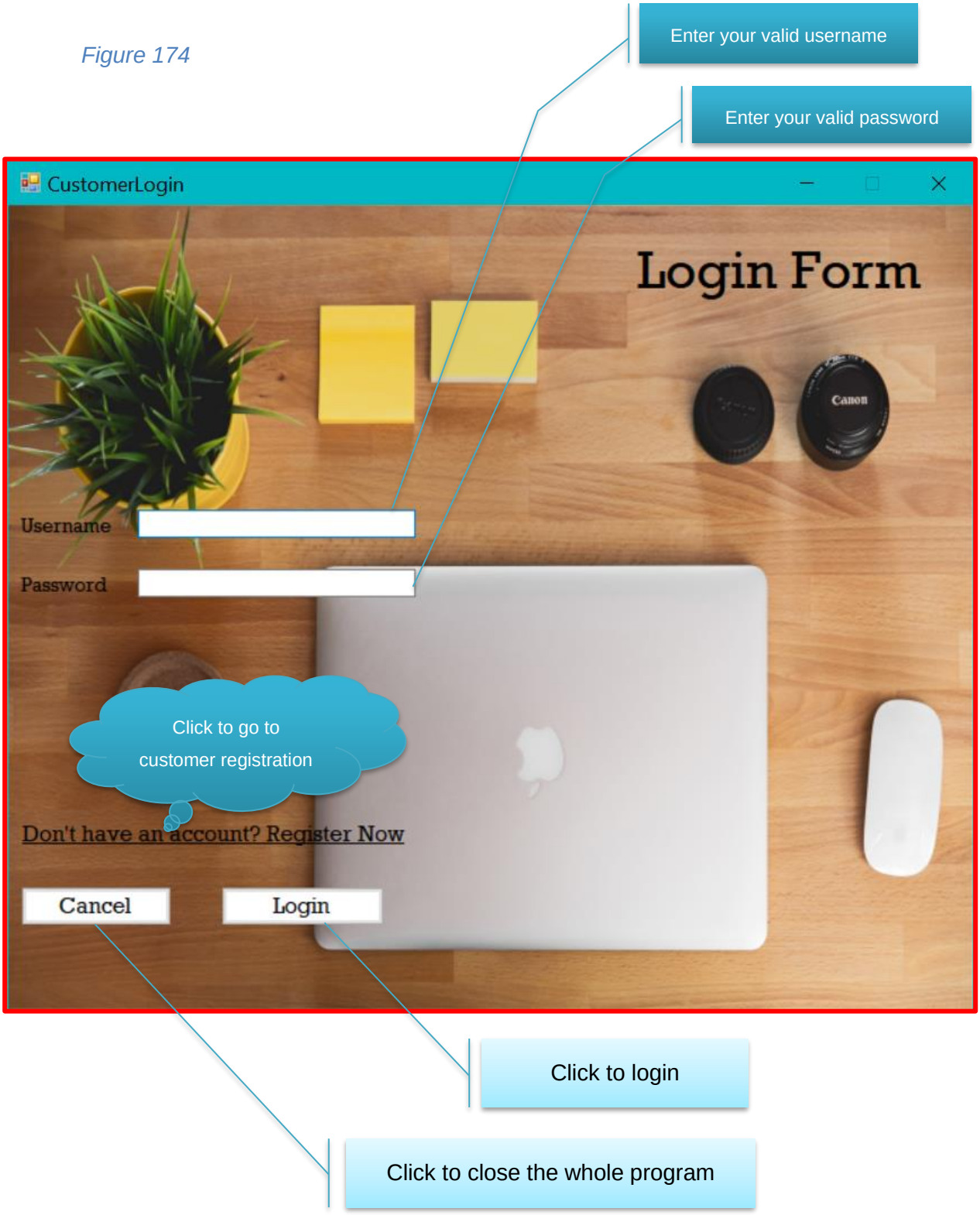
8. Report form page

Figure 173



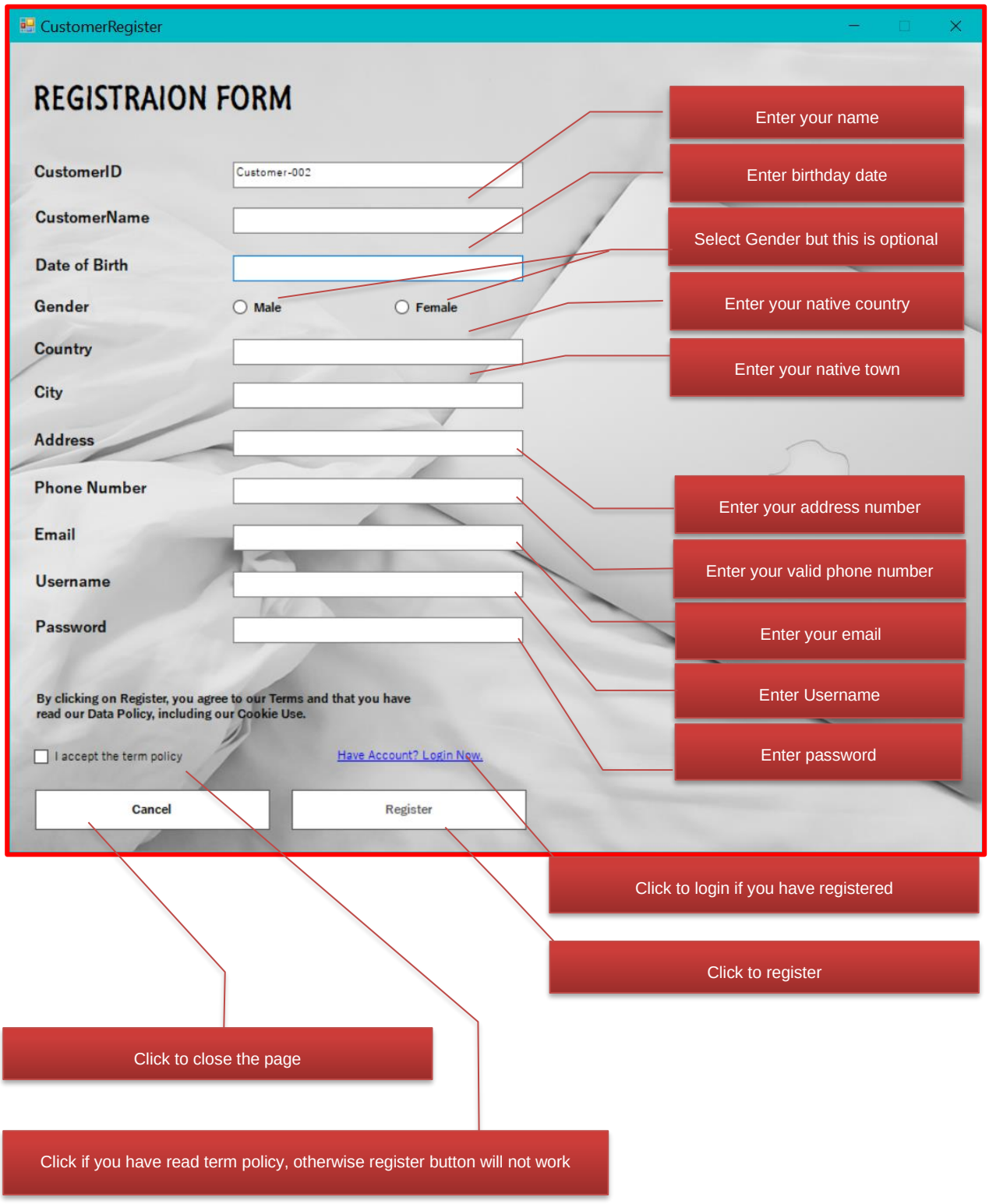
9. Customer login page

Figure 174



10. Customer registration page

Figure 175



11. Customer dashboard page

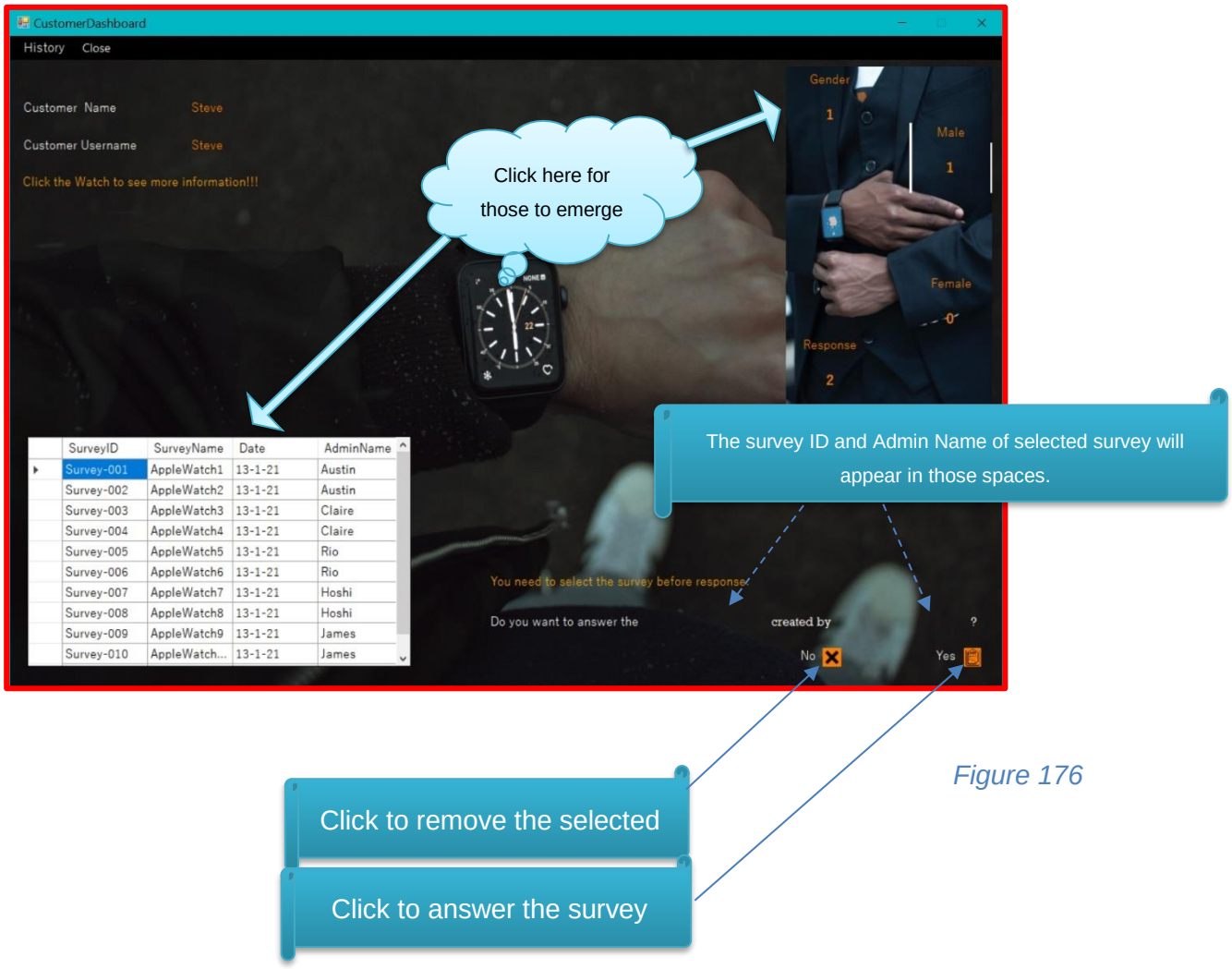


Figure 176

12. Customer Answer Survey

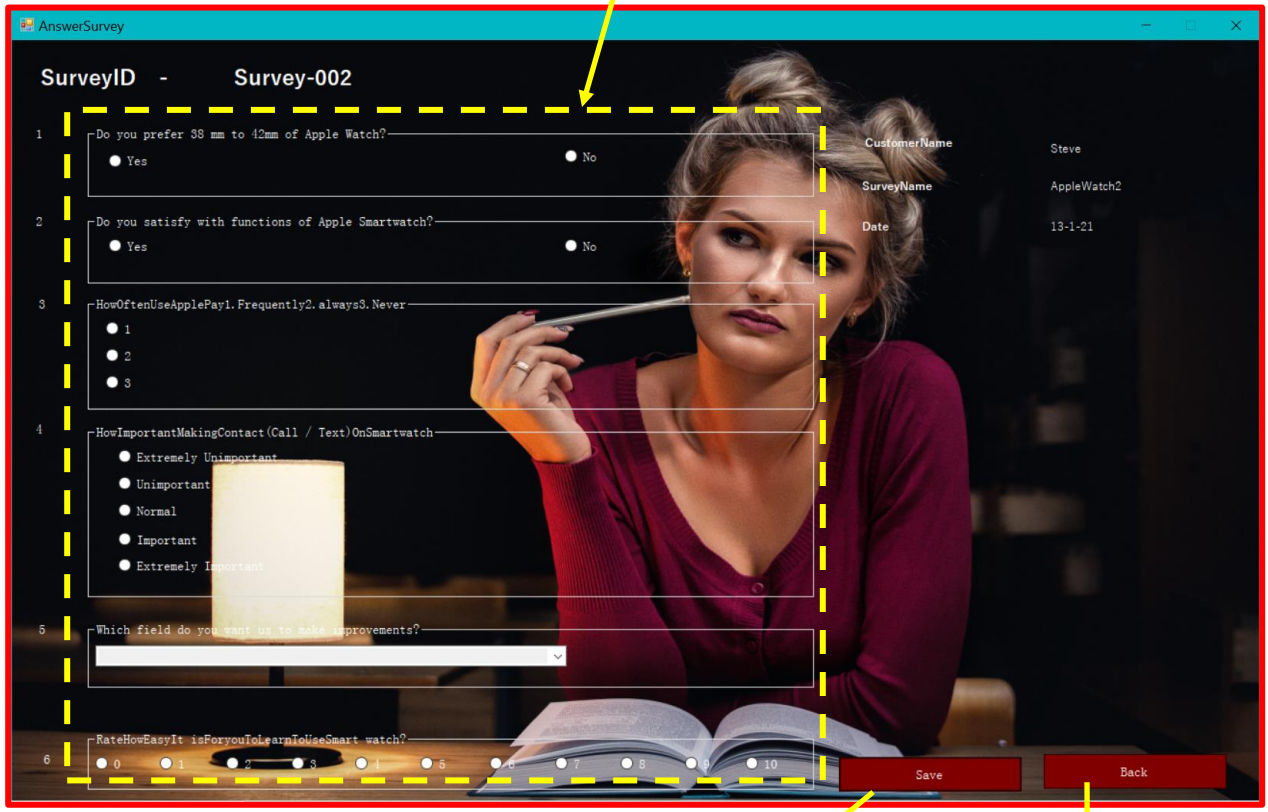


Figure 177

13. Customer history page

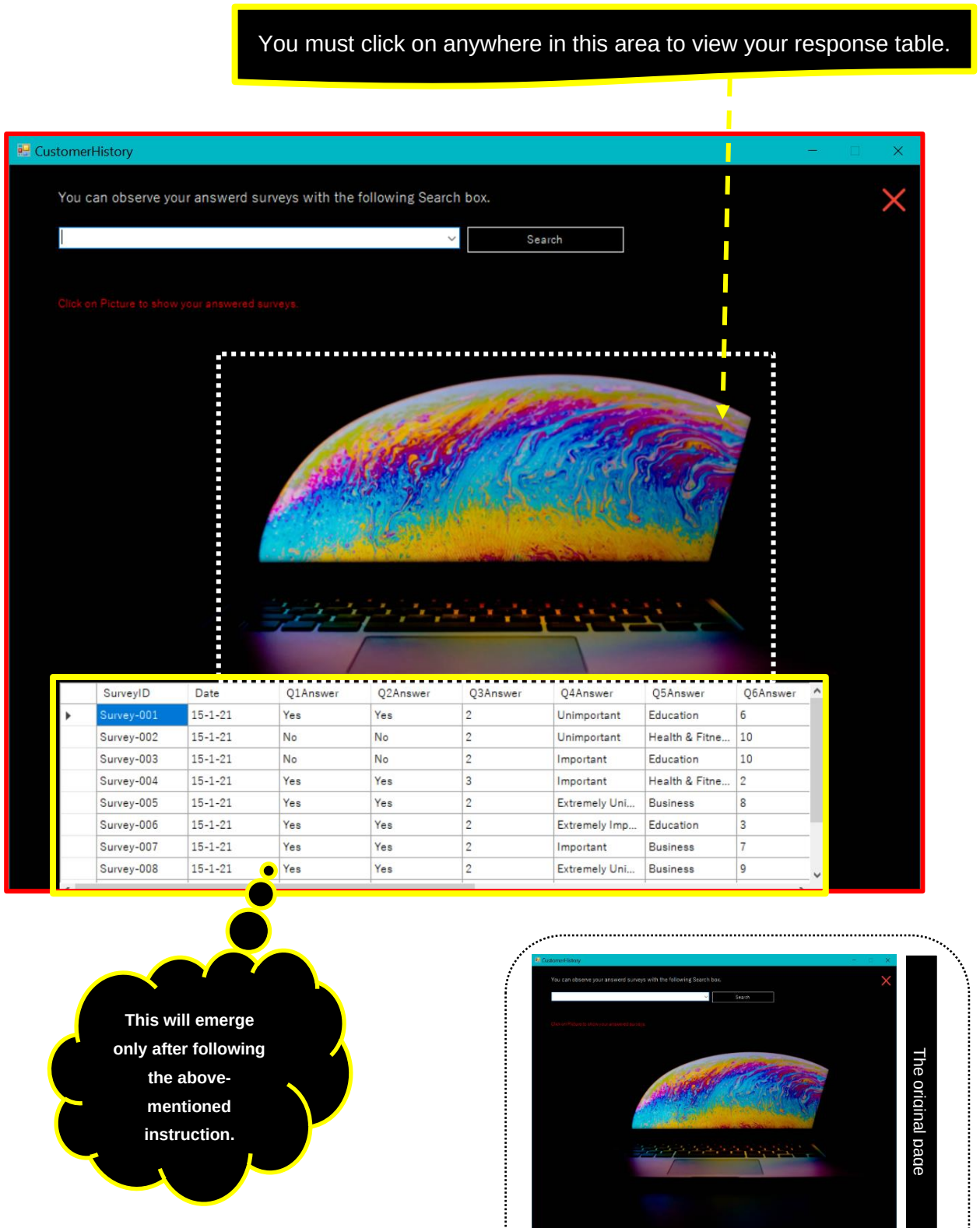


Figure 178

References

GeeksforGeeks. 2021. *Differences Between Black Box Testing Vs White Box Testing - Geeksforgeeks*. [online]
Available at: https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box testing/#:~:text=Black%20Box%20Testing%20is%20a,is%20known%20to%20the%20tester.
[Accessed 9 January 2021].

Guru99.com. 2021. *What Is WHITE Box Testing? Techniques, Example & Types*. [online]
Available at: https://www.guru99.com/white-box-testing.html

[Accessed 16 January 2021].

Learning Center. 2021. *What Is White Box Testing | Types & Techniques For Code Coverage | Imperva*. [online]
Available at: https://www.imperva.com/learn/application-security/white-box-testing/
[Accessed 16 January 2021].

Softwaretestinghelp.com. 2021. *Types Of Software Testing: Different Testing Types With Details*. [online]
Available at: https://www.softwaretestinghelp.com/types-of-software-testing/
[Accessed 25 January 2021].

Softwaretestingmaterial.com. 2021. *What Is Software Testing | Everything You Should Know*. [online]
Available at: https://www.softwaretestingmaterial.com/software-testing/
[Accessed 15 January 2021].

Writer, S., 2021. *Black Box Testing Technique, Its Types & Approaches With Example*. [online] ReQtest.
Availableat:https://reqtest.com/testing-blog/black-box-testing/#:~:text=Black%20box%20testing%20is%20used,server%20logic%2C%20and%20development%20method.
[Accessed 25 January 2021].