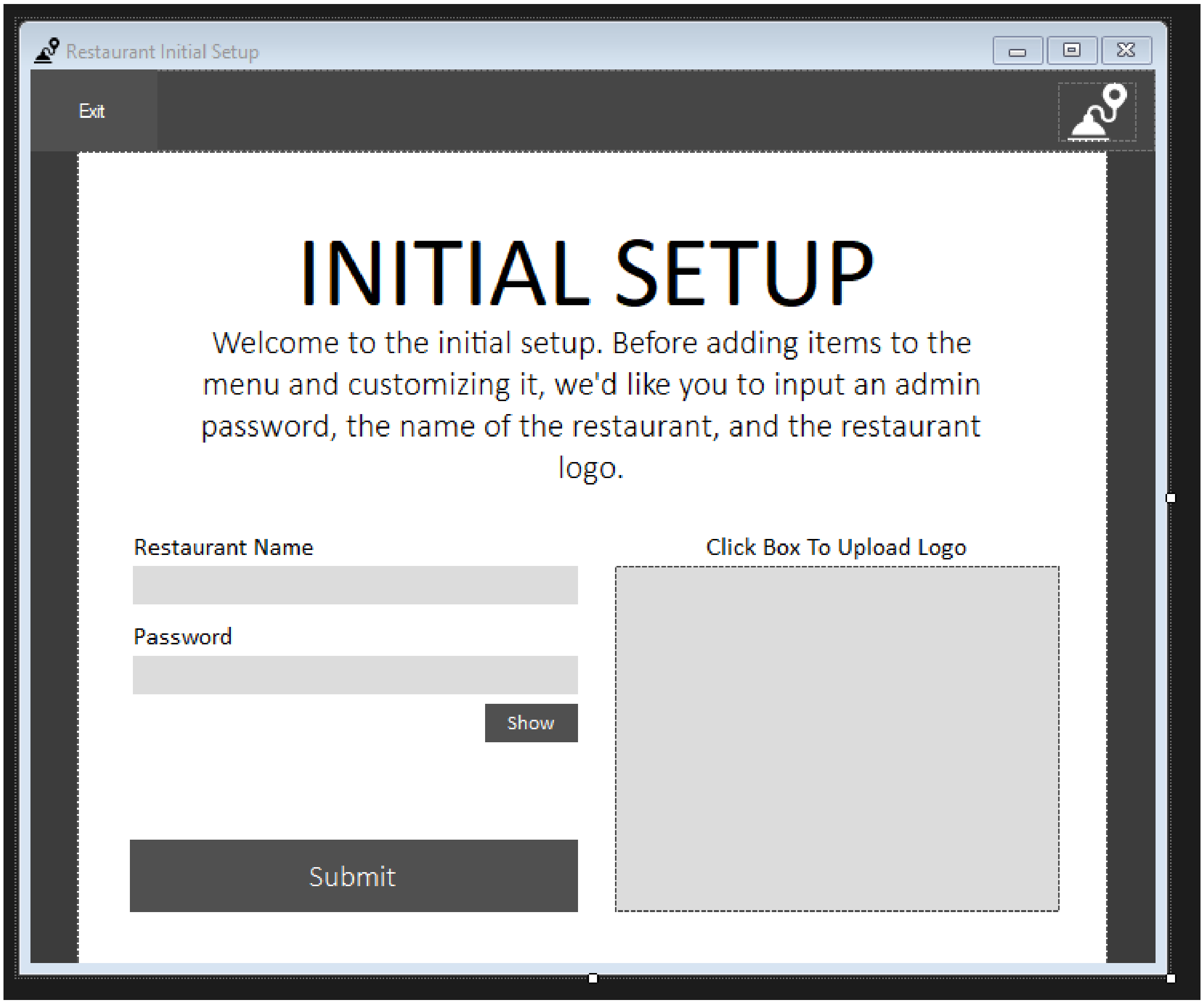
Contactless Dining System Report

Zaid Issa 20190148

Contents

GUI Documentation

Initial Setup

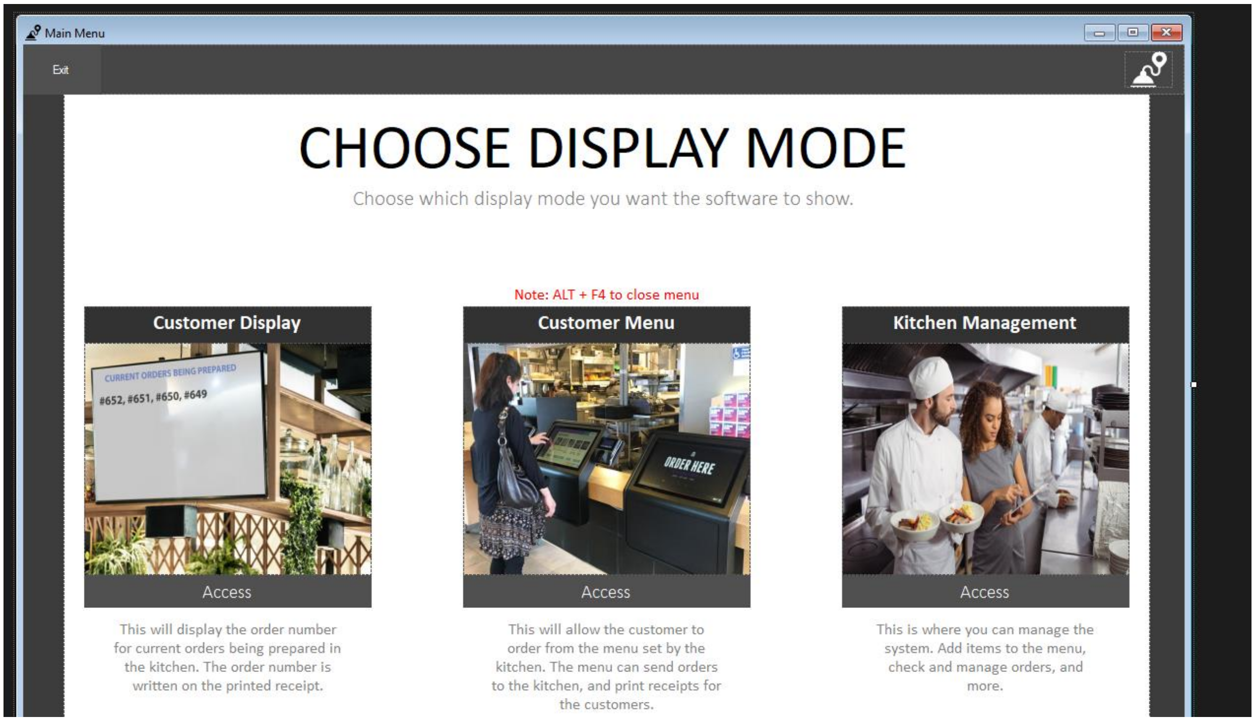


INITIAL SETUP

Welcome to the initial setup. Before adding items to the menu and customizing it, we'd like you to input an admin password, the name of the restaurant, and the restaurant logo.

In the initial setup, the restaurant can add their restaurant logo, name, and manager password. After submitting the proper details, they will be taken to the main menu.

Main Menu



CHOOSE DISPLAY MODE

Choose which display mode you want the software to show.

Note: ALT + F4 to close menu

Customer Display

This will display the order number for current orders being prepared in the kitchen. The order number is written on the printed receipt.

Customer Menu

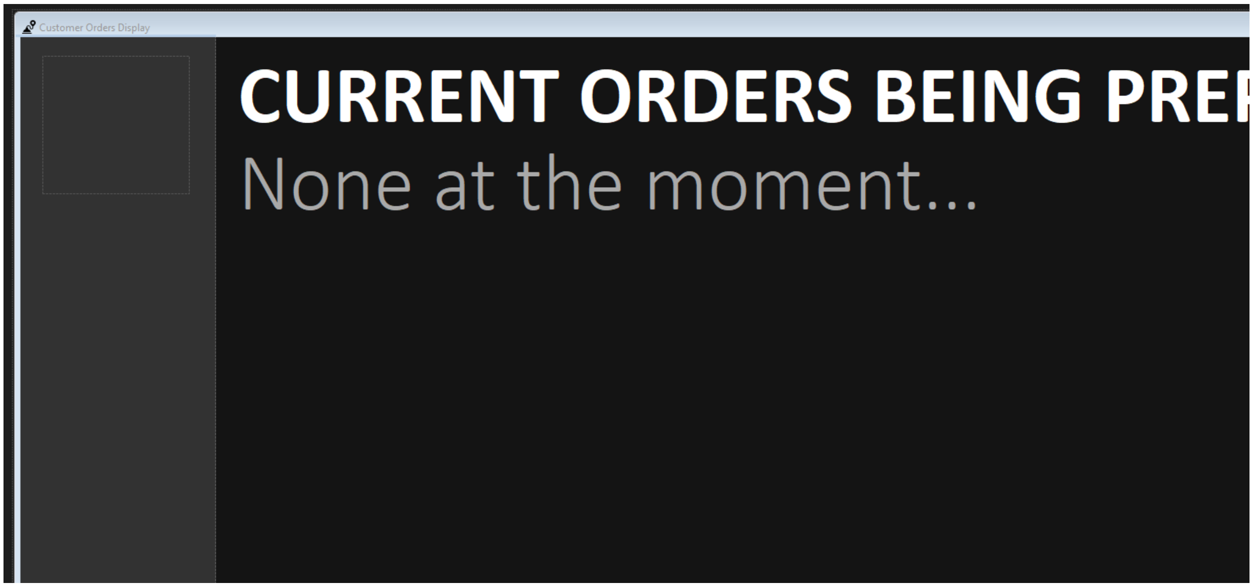
This will allow the customer to order from the menu set by the kitchen. The menu can send orders to the kitchen, and print receipts for the customers.

Kitchen Management

This is where you can manage the system. Add items to the menu, check and manage orders, and more.

In the main menu, the restaurant is able to access the different panels based on their needs. If customer display is accessed, then it'll take them to the display that shows the customers which order IDs are being prepared at the moment. If customer menu is clicked, then the customer menu will open allowing the customer to order from the restaurant's menu. Finally, if the kitchen management is clicked, then it'll ask for the manager password before allowing them in.

Orders being prepared display

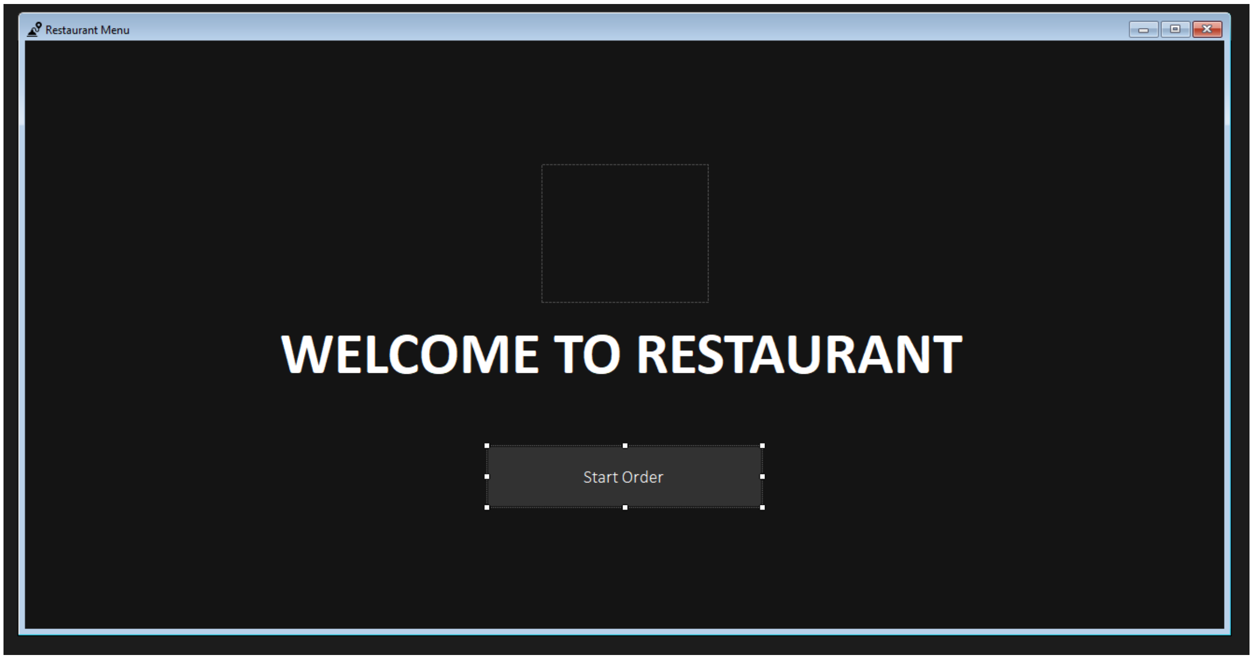


CURRENT ORDERS BEING PREP

None at the moment...

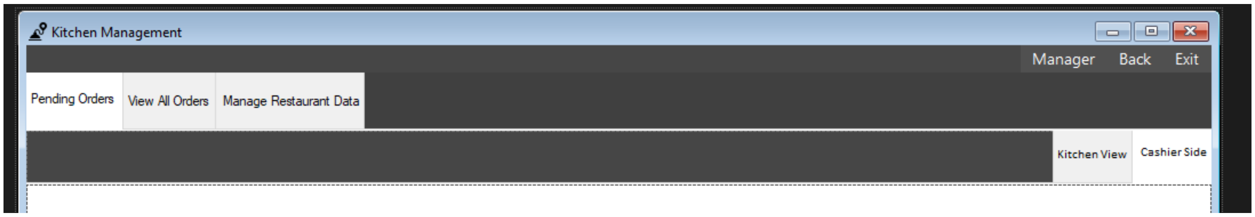
In the customer display, the logo of the restaurant is loaded on the left; while on the right, all the order IDs of orders being prepared will show up automatically, and refresh every 5 seconds.

Customer Ordering Menu Splash Screen



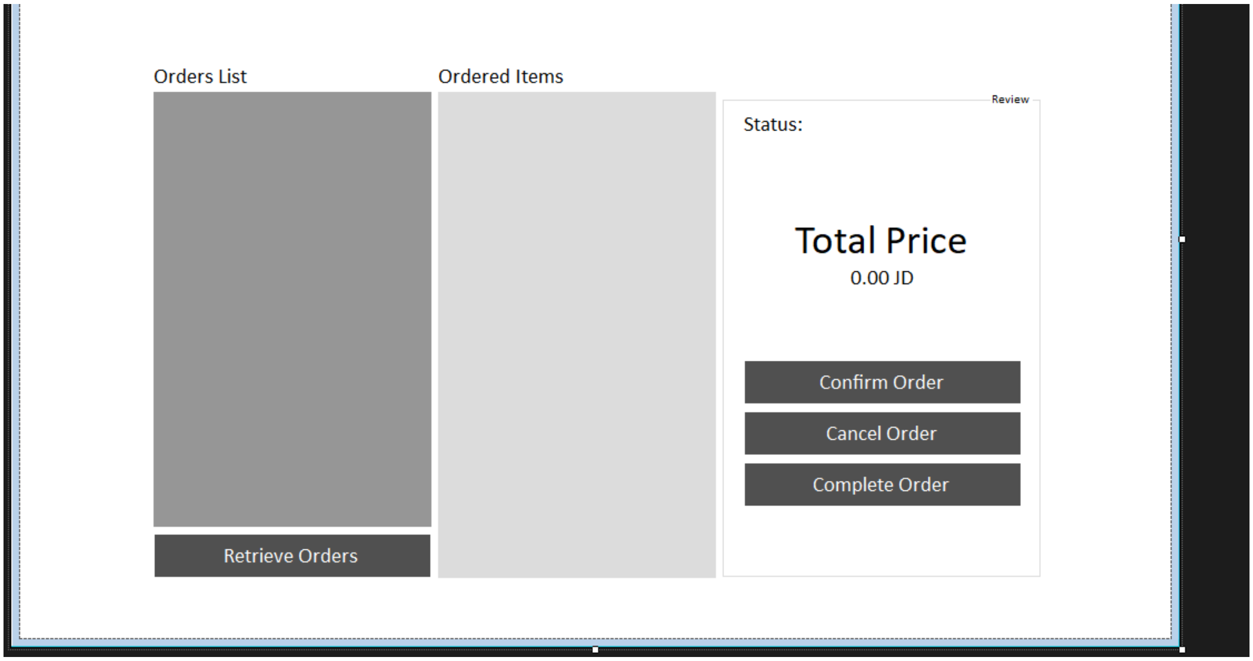
The splash screen displays the restaurant name and their logo. Once the customer clicks on start order it'll reveal the complete menu, where they could order.

Management Menu



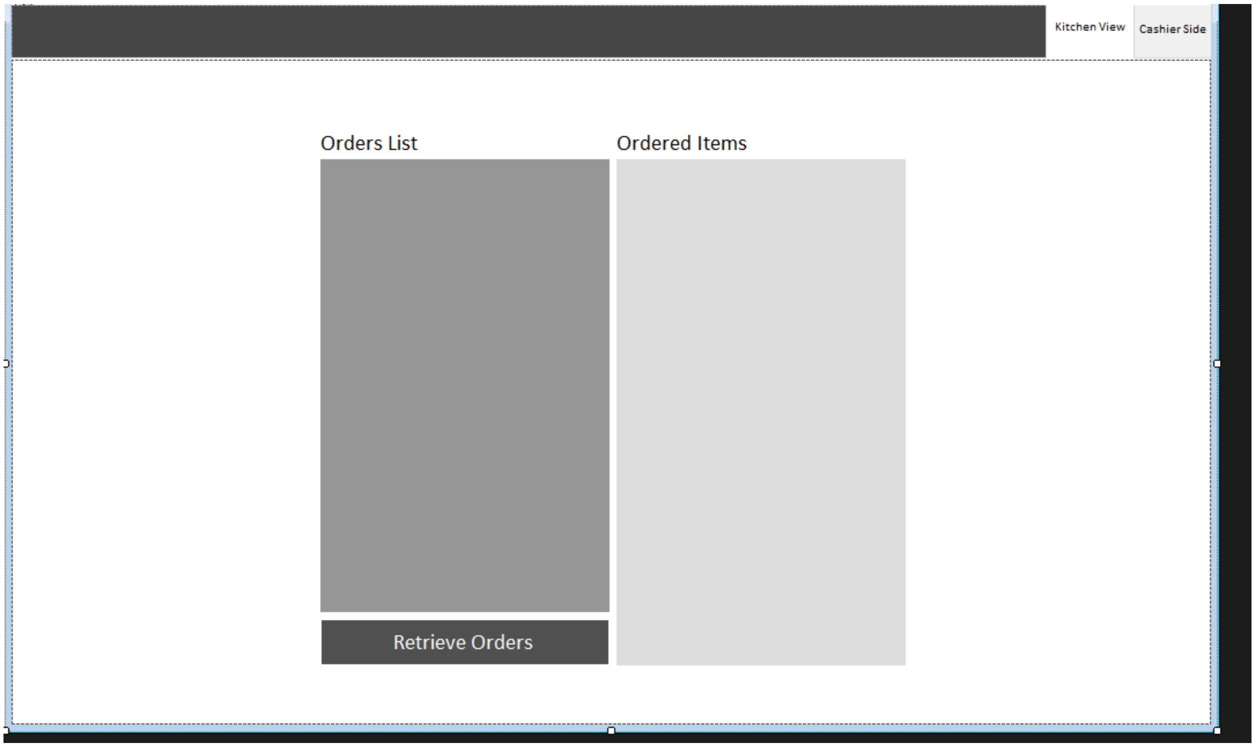
The kitchen management tabs allow us to navigate the kitchen management; in addition, in the top right corner, the manager could lock the current tab open, so employees don't change their designated tab; manager could also click back to go back to main menu, exit to close the software, or sign out from manager mode.

Cashier Side



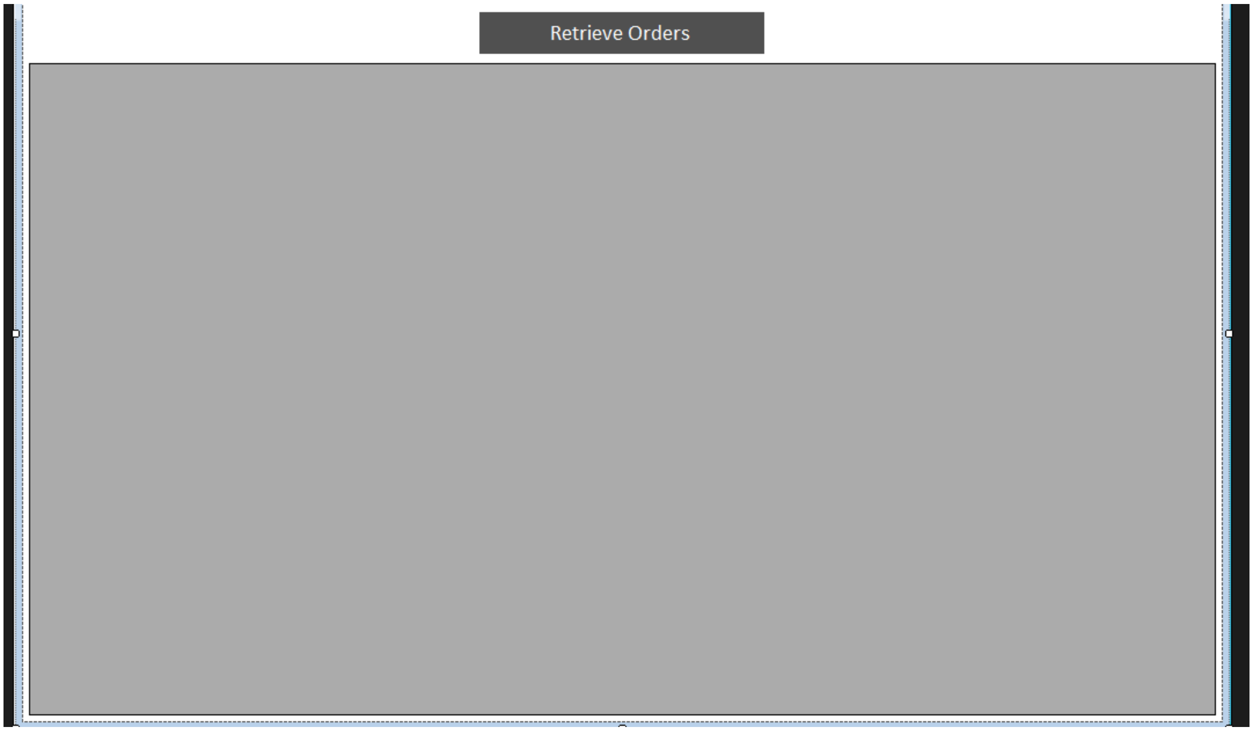
Cashier tab allows the employee to view the orders pending or being prepared. Orders viewed here can be modified as per the following: confirm an order is paid, cancel an order, or complete an order once customers receive it.

Kitchen Side



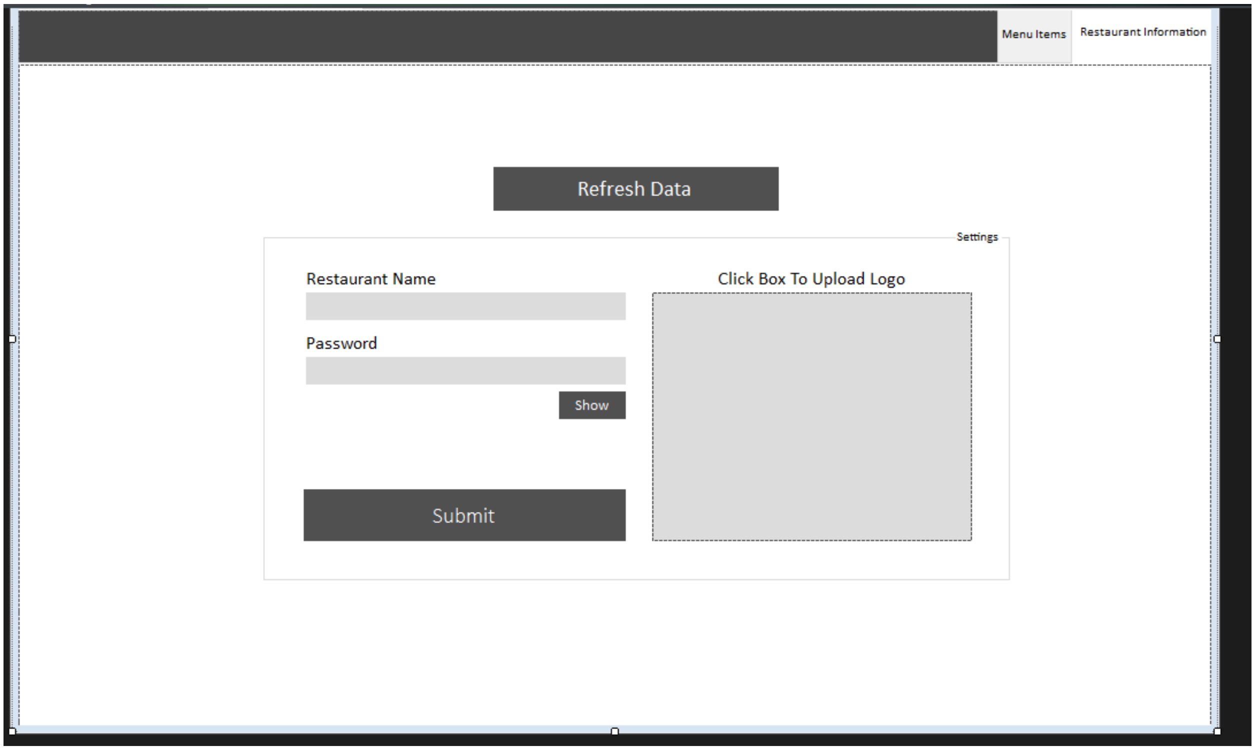
Kitchen tab shows the orders need to get prepared, while the ordered items show the item name, and the quantity.

Manager Retrieve All Orders



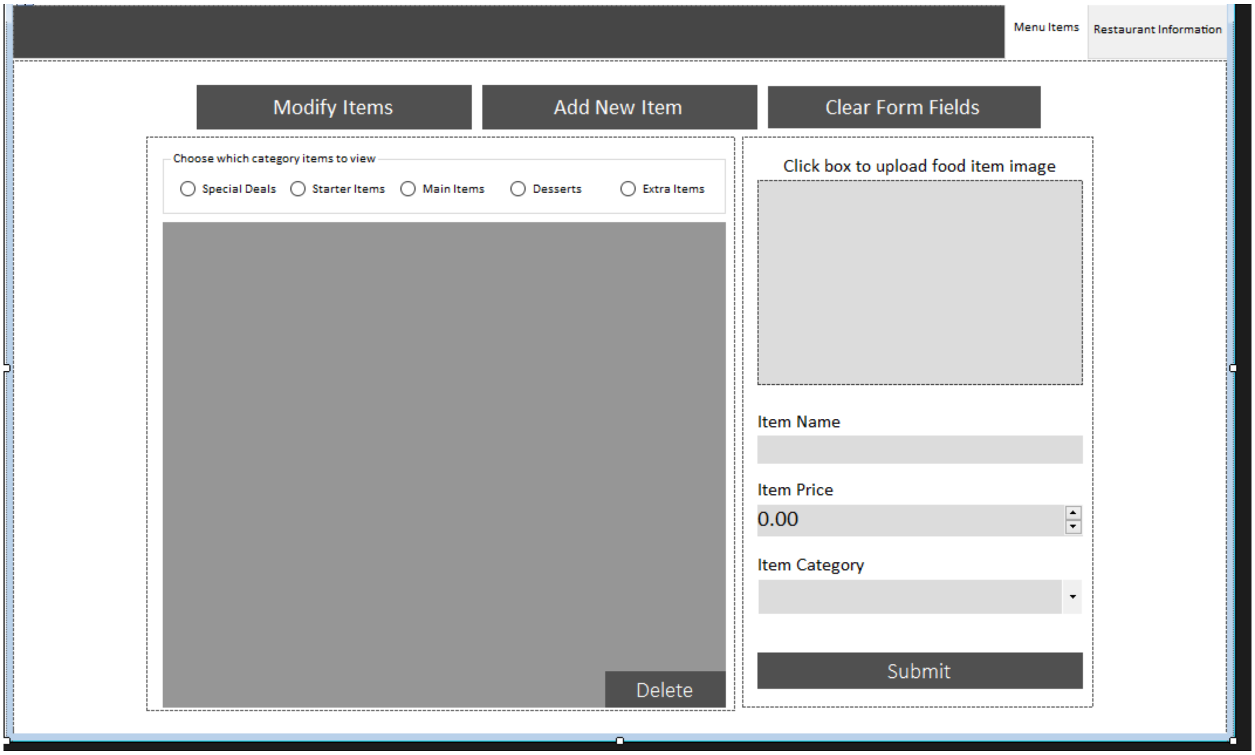
This tab allows the manager to view all orders with any status.

Restaurant Info Manager



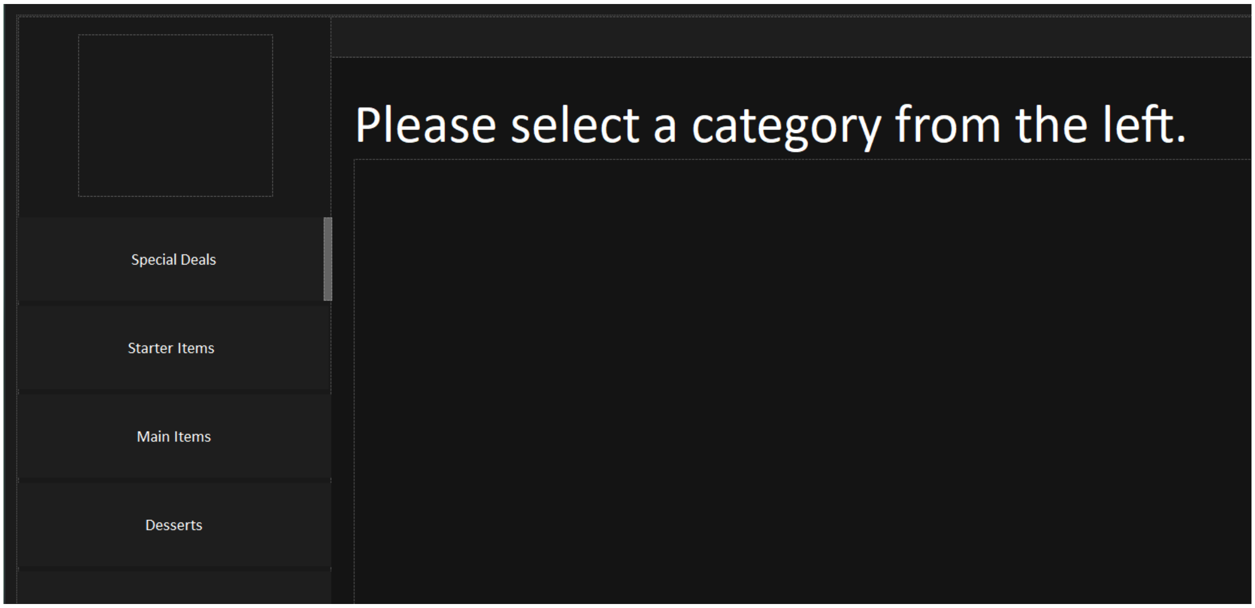
This tab allows the manager to change the restaurant name, password, or logo set before.

Manage Menu Items

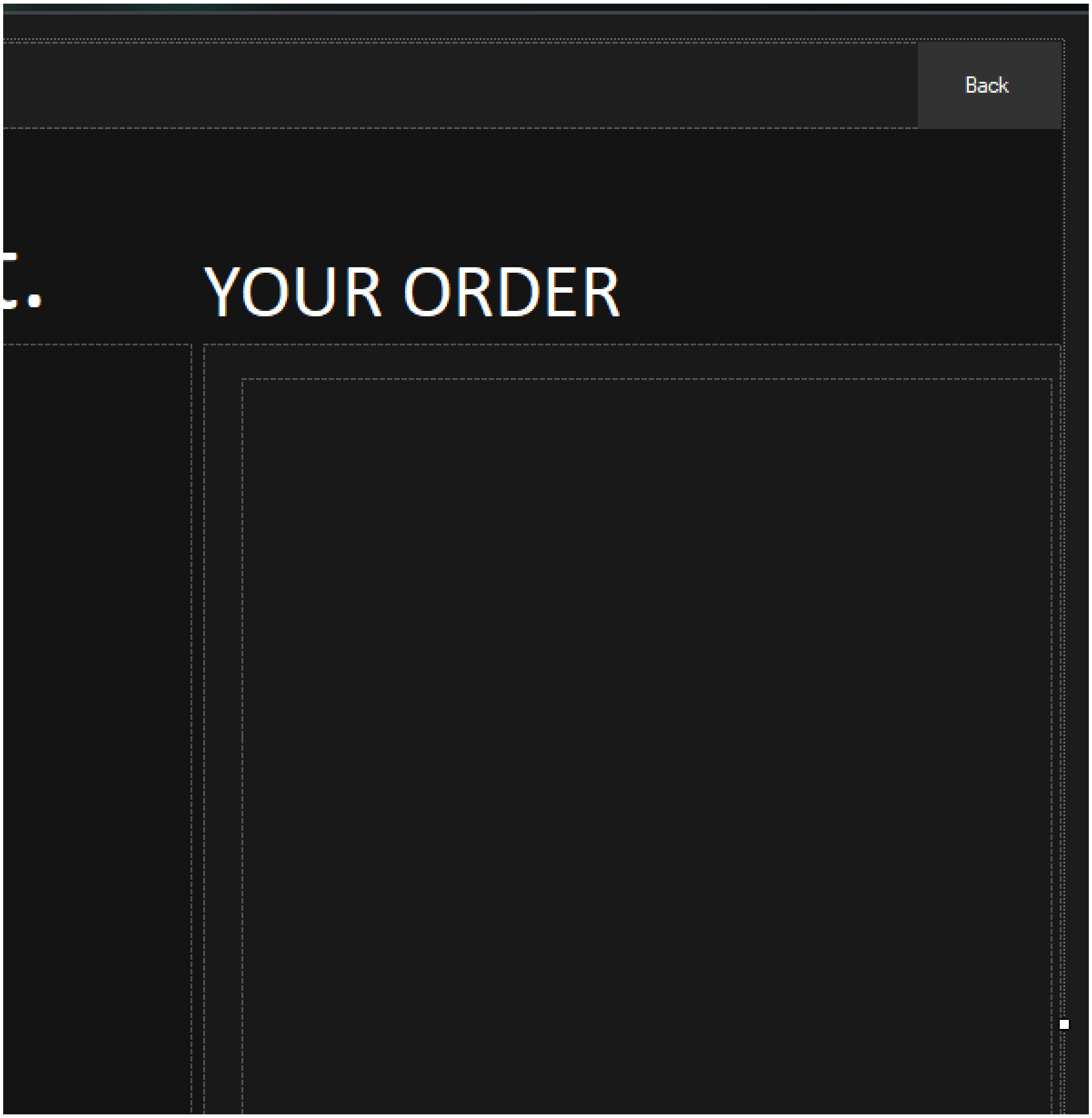


This tab allows the manager to delete, modify, and insert items on the restaurant menu.

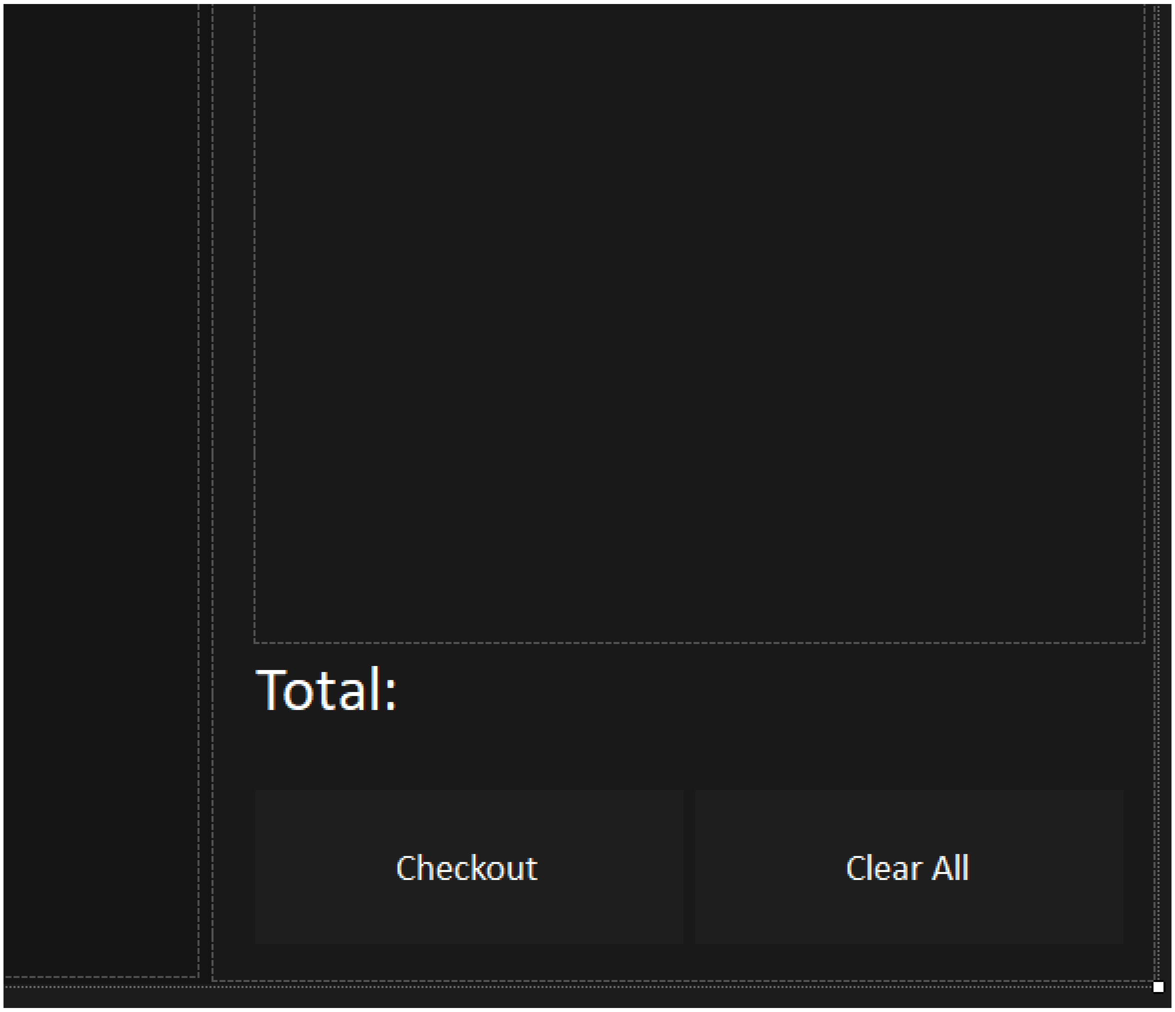
Customer Menu



The restaurant logo is loaded from the database on the left. Under that is the categories a customer can navigate through, and depending on which category is clicked, it'll display items under that category.

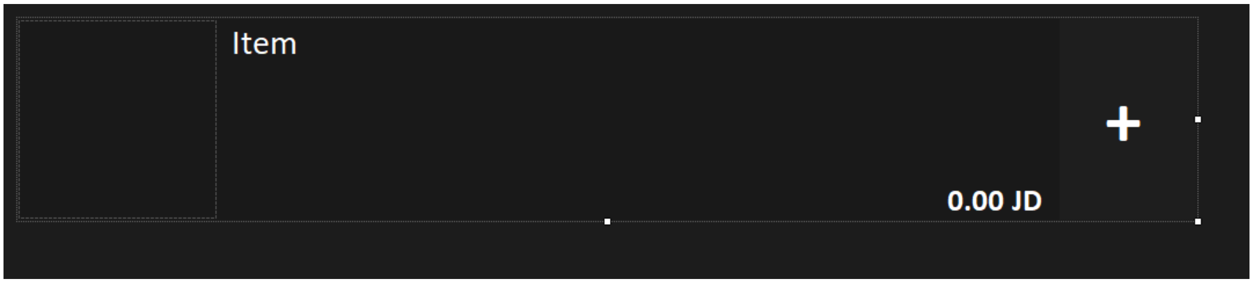


The back button allows the user to cancel their order, then the panel will go back to the splash screen. Under the “YOUR ORDER” label all the items in the cart would appear.

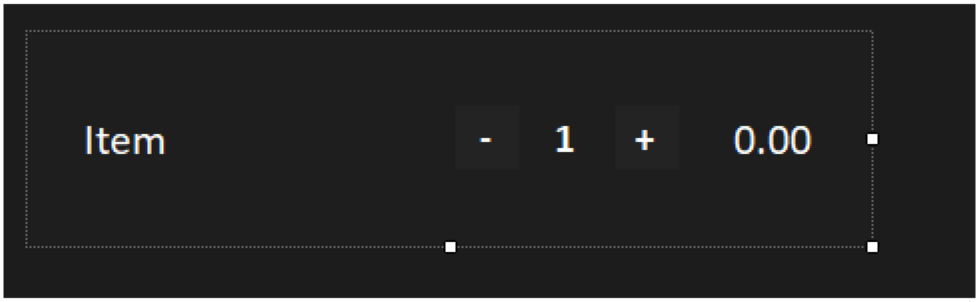


Total would automatically show the cart's total cost. The checkout allows the customer to print the receipt that they could show to cashier to pay and confirm their order. The clear all button allows the customer to delete all the items in their cart.

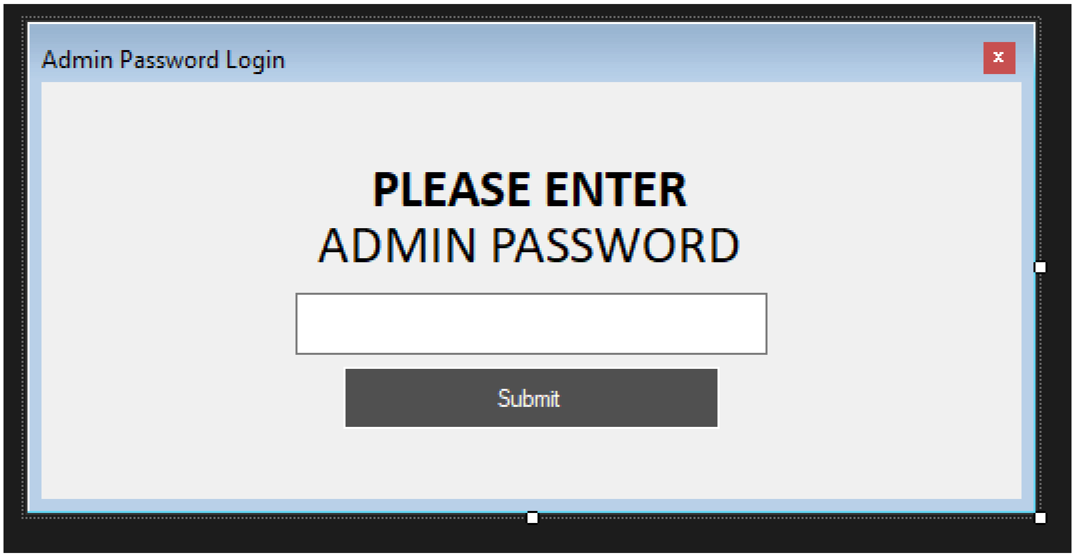
Extra Windows Forms



This user control loads the item name and its price. Once the plus button has been clicked, it'll be added to the cart.



This user control loads the item name, quantity, and price from the cart.

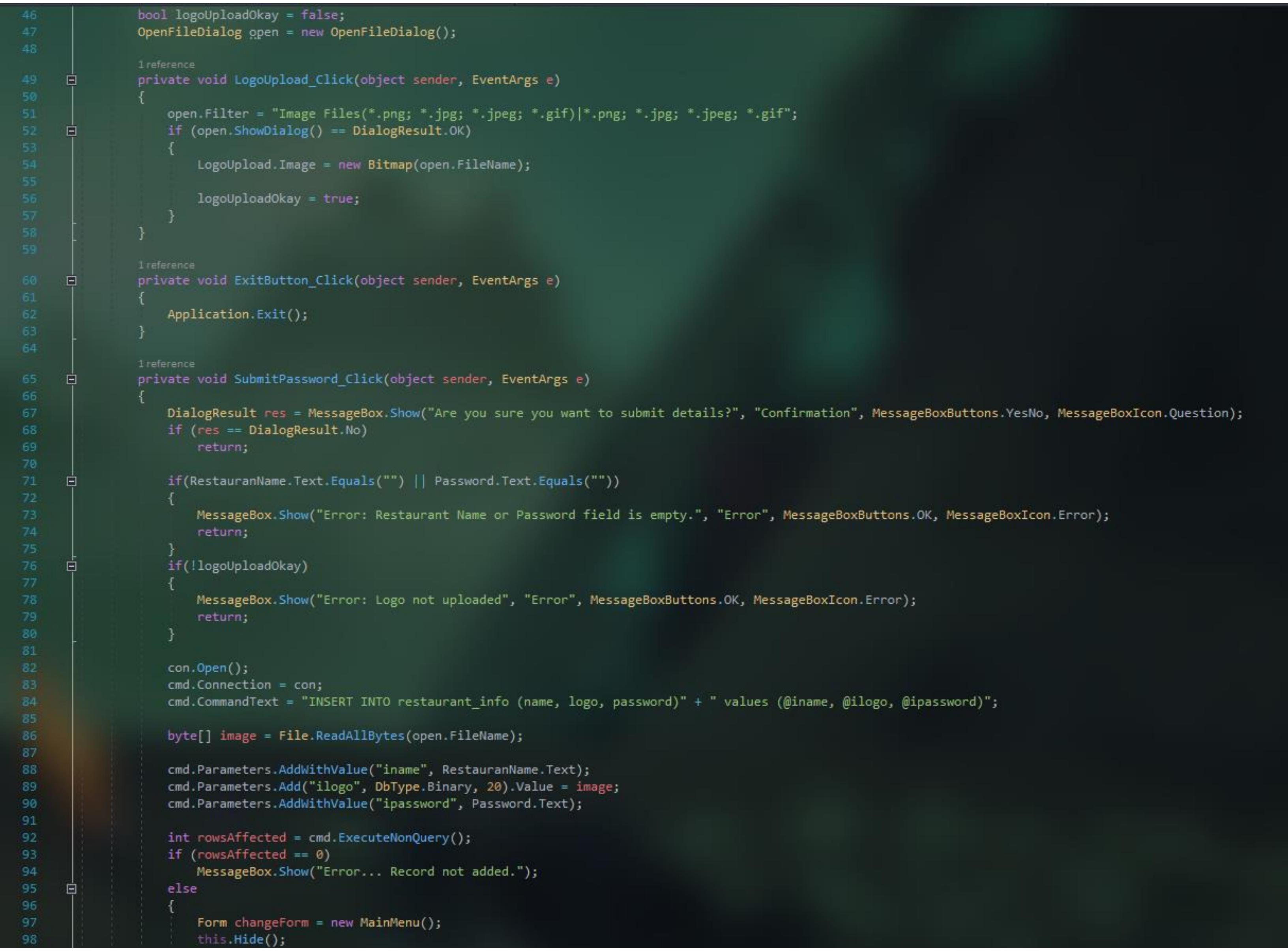

Admin Password Login

PLEASE ENTER
ADMIN PASSWORD

This form is a custom dialog where it checks if the user is able to access manager level forms.

Code Implementation

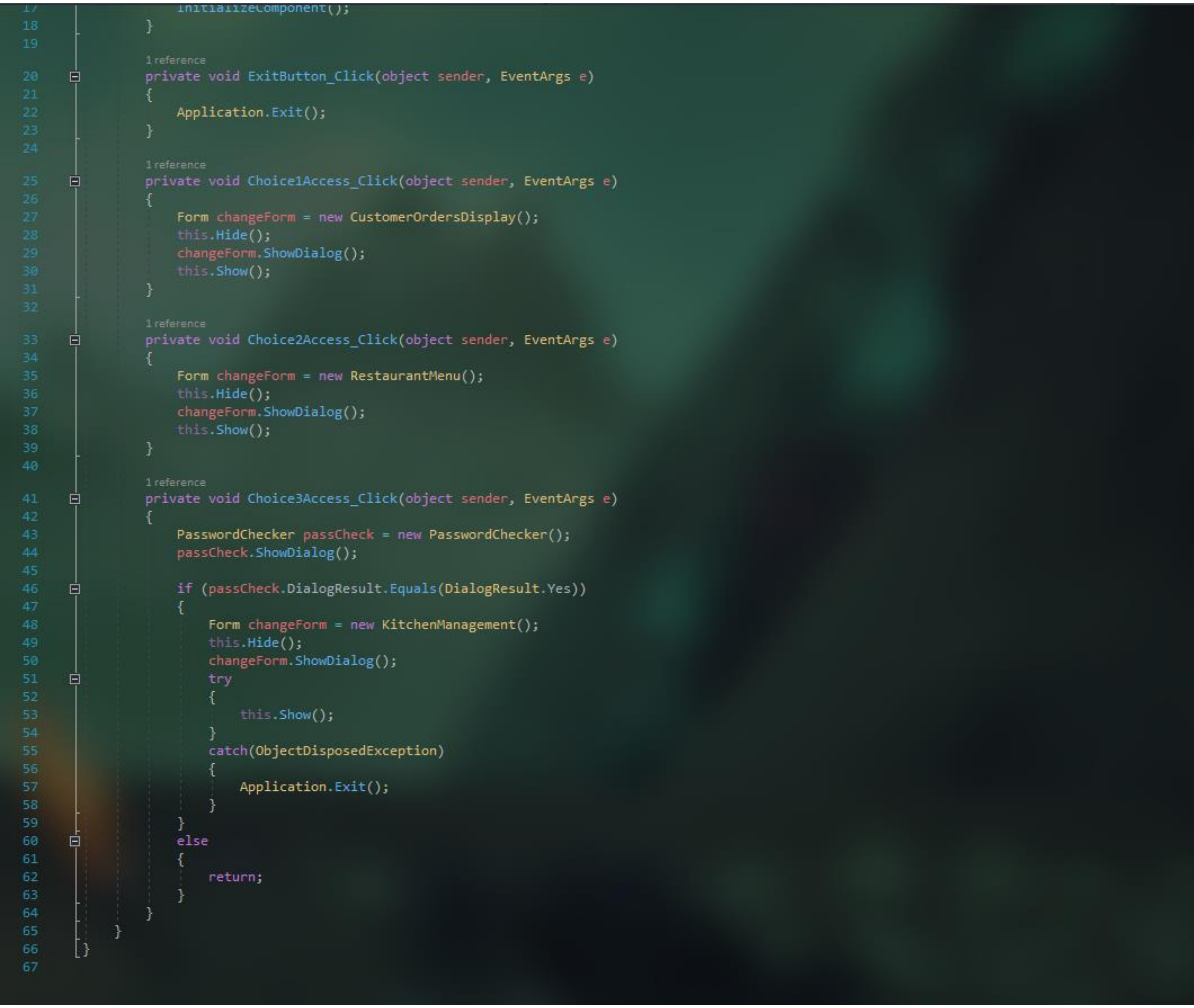
Form1.cs



```
        bool logoUploadOkay = false;
        OpenFileDialog open = new OpenFileDialog();

        1 reference
        private void LogoUpload_Click(object sender, EventArgs e)
        {
            open.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
            if (open.ShowDialog() == DialogResult.OK)
            {
                LogoUpload.Image = new Bitmap(open.FileName);

                logoUploadOkay = true;
            }
        }

        1 reference
        private void ExitButton_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        1 reference
        private void SubmitPassword_Click(object sender, EventArgs e)
        {
            DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (res == DialogResult.No)
                return;

            if(RestauranName.Text.Equals("") || Password.Text.Equals(""))
            {
                MessageBox.Show("Error: Restaurant Name or Password field is empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if(!logoUploadOkay)
            {
                MessageBox.Show("Error: Logo not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "INSERT INTO restaurant_info (name, logo, password)" + " values (@iname, @ilogo, @ipassword)";

            byte[] image = File.ReadAllBytes(open.FileName);

            cmd.Parameters.AddWithValue("iname", RestauranName.Text);
            cmd.Parameters.Add("ilogo", DbType.Binary, 20).Value = image;
            cmd.Parameters.AddWithValue("ipassword", Password.Text);

            int rowsAffected = cmd.ExecuteNonQuery();
            if (rowsAffected == 0)
                MessageBox.Show("Error... Record not added.");
            else
            {
                Form changeForm = new MainMenu();
                this.Hide();
```

- LogoUpload_Click opens an OpenFileDialog that only accepts image type files. If image is received, we update the box with the uploaded image.
- ExitButton_Click closes the application.
- SubmitPassword_Click checks if user wants to submit details, if no, return. We check if text is valid and logo has been uploaded; then, we connect to the database and upload the details using insert.
- Logo has to be converted into a byte array in order to be sent to database.

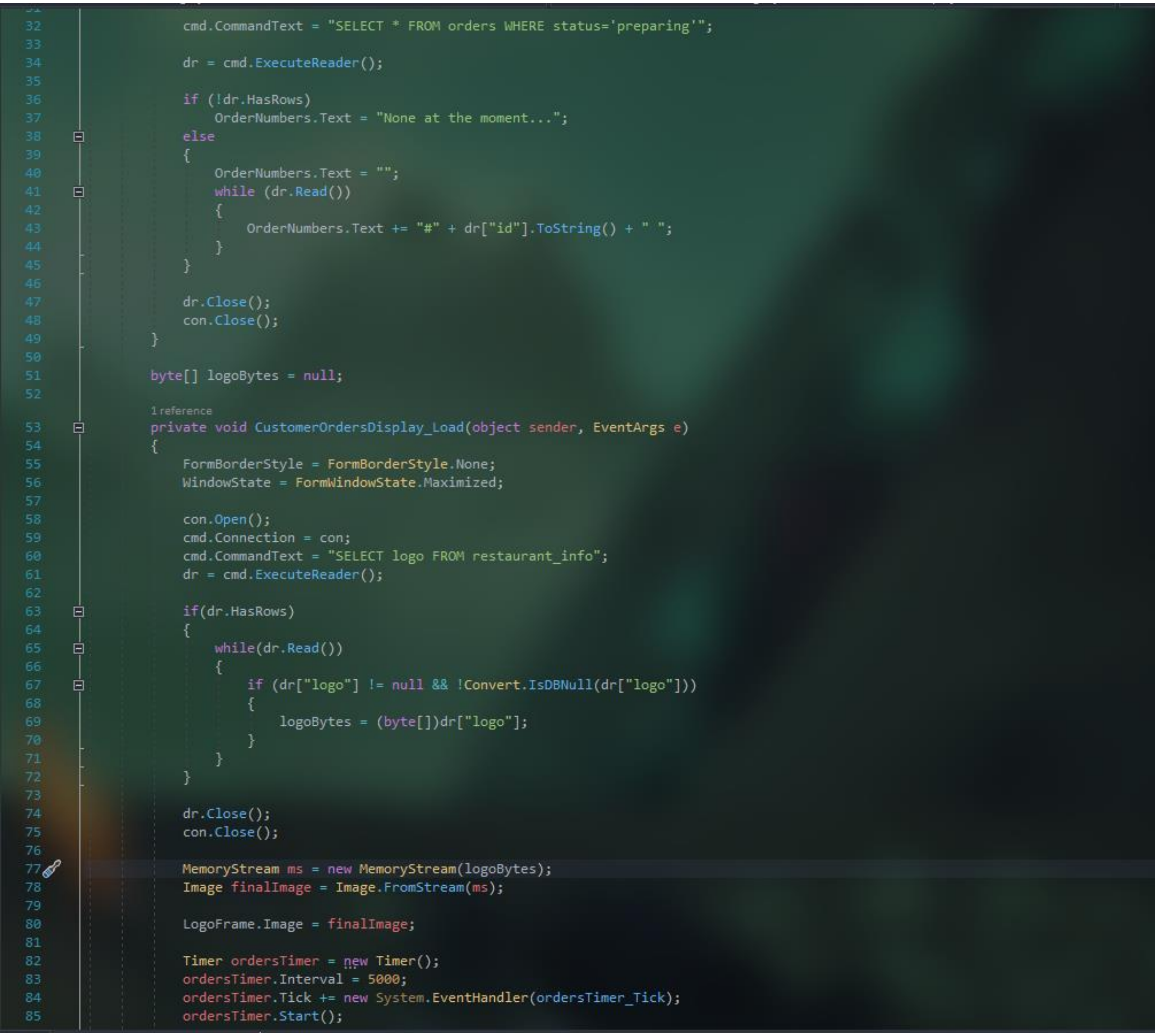
Form2.cs



```
            InitializeComponent();
        }

        1 reference
        private void ExitButton_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        1 reference
        private void Choice1Access_Click(object sender, EventArgs e)
        {
            Form changeForm = new CustomerOrdersDisplay();
            this.Hide();
            changeForm.ShowDialog();
            this.Show();
        }

        1 reference
        private void Choice2Access_Click(object sender, EventArgs e)
        {
            Form changeForm = new RestaurantMenu();
            this.Hide();
            changeForm.ShowDialog();
            this.Show();
        }

        1 reference
        private void Choice3Access_Click(object sender, EventArgs e)
        {
            PasswordChecker passCheck = new PasswordChecker();
            passCheck.ShowDialog();

            if (passCheck.DialogResult.Equals(DialogResult.Yes))
            {
                Form changeForm = new KitchenManagement();
                this.Hide();
                changeForm.ShowDialog();
                try
                {
                    this.Show();
                }
                catch(ObjectDisposedException)
                {
                    Application.Exit();
                }
            }
            else
            {
                return;
            }
        }
    }
}
```

- ExitButton_Click exits application.
- Choice1Access_Click and Choice2Access_Click creates a new form object, hides the main menu form, then shows the new form object created (depending on choice).
- Choice3Access_Click checks for password using the PasswordChecker form before giving access to kitchen management form. A try catch operation has been used here due to deleted object might being used.

Form3.cs

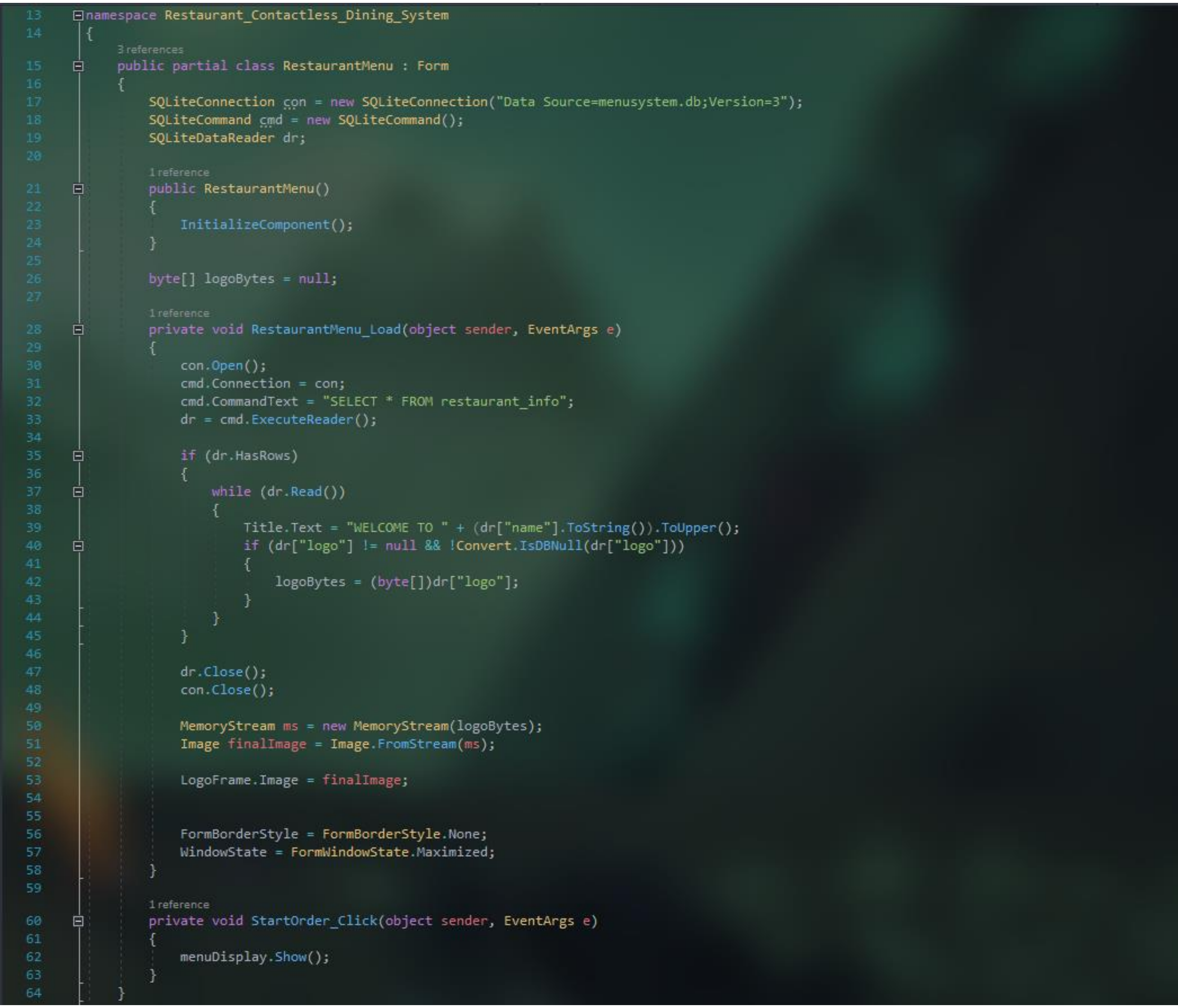


```
            cmd.CommandText = "SELECT * FROM orders WHERE status='preparing'";

            dr = cmd.ExecuteReader();

            if (!dr.HasRows)
                OrderNumbers.Text = "None at the moment...";
            else
            {
                OrderNumbers.Text = "";
                while (dr.Read())
                {
                    OrderNumbers.Text += "#" + dr["id"].ToString() + " ";
                }
            }

            dr.Close();
            con.Close();
        }

        byte[] logoBytes = null;

        1 reference
        private void CustomerOrdersDisplay_Load(object sender, EventArgs e)
        {
            FormBorderStyle = FormBorderStyle.None;
            WindowState = FormWindowState.Maximized;

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT logo FROM restaurant_info";
            dr = cmd.ExecuteReader();

            if(dr.HasRows)
            {
                while(dr.Read())
                {
                    if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
                    {
                        logoBytes = (byte[])dr["logo"];
                    }
                }
            }

            dr.Close();
            con.Close();

            MemoryStream ms = new MemoryStream(logoBytes);
            Image finalImage = Image.FromStream(ms);

            LogoFrame.Image = finalImage;

            Timer ordersTimer = new Timer();
            ordersTimer.Interval = 5000;
            ordersTimer.Tick += new System.EventHandler(ordersTimer_Tick);
            ordersTimer.Start();
```

- CustomerOrdersDIsplay_Load retrieves restaurant logo from database and shows it in the picture box set. Then it creates an interval timer that keeps activating ordersTImer_Tick function.

- ordersTimer_Tick function is activated each 5 seconds to retrieve info from database. The info retrieved checks the order numbers for the orders being prepared at the moment, if none then it'll show that.

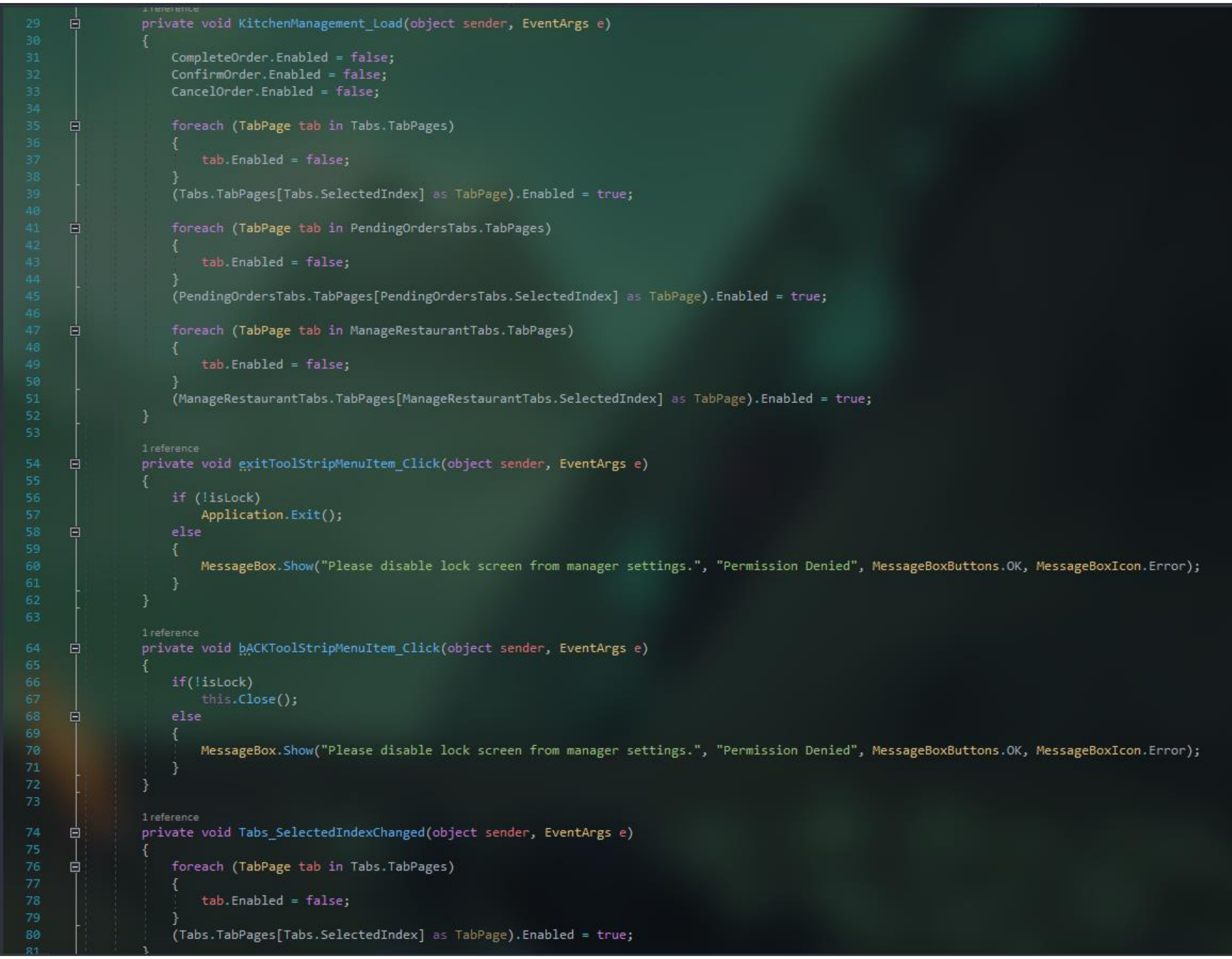
Form4.cs



```
namespace Restaurant_Contactless_Dining_System
{
    3 references
    public partial class RestaurantMenu : Form
    {
        SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
        SQLiteCommand cmd = new SQLiteCommand();
        SQLiteDataReader dr;

        1 reference
        public RestaurantMenu()
        {
            InitializeComponent();
        }

        byte[] logoBytes = null;

        1 reference
        private void RestaurantMenu_Load(object sender, EventArgs e)
        {
            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM restaurant_info";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    Title.Text = "WELCOME TO " + (dr["name"].ToString()).ToUpper();
                    if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
                    {
                        logoBytes = (byte[])dr["logo"];
                    }
                }
            }

            dr.Close();
            con.Close();

            MemoryStream ms = new MemoryStream(logoBytes);
            Image finalImage = Image.FromStream(ms);

            LogoFrame.Image = finalImage;


            FormBorderStyle = FormBorderStyle.None;
            WindowState = FormWindowState.Maximized;
        }

        1 reference
        private void StartOrder_Click(object sender, EventArgs e)
        {
            menuDisplay.Show();
        }
    }
```

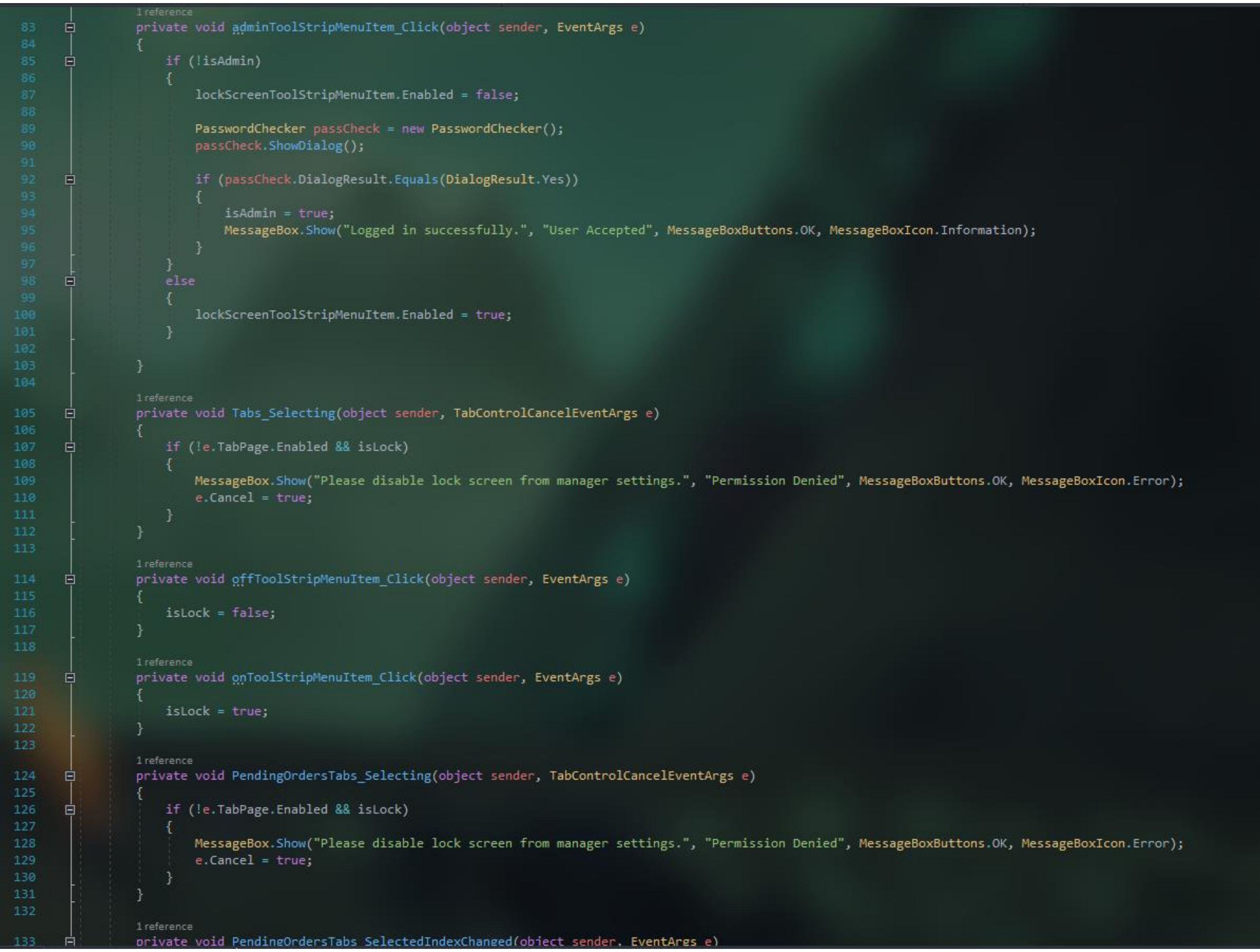
- Load event gets the restaurant logo and name from database then sets the image in the picture box and changes the splash screen welcome text depending on restaurant name.
- StartOrder_Click starts the CompleteMenu user control that allows customers to order from.

Form5.cs



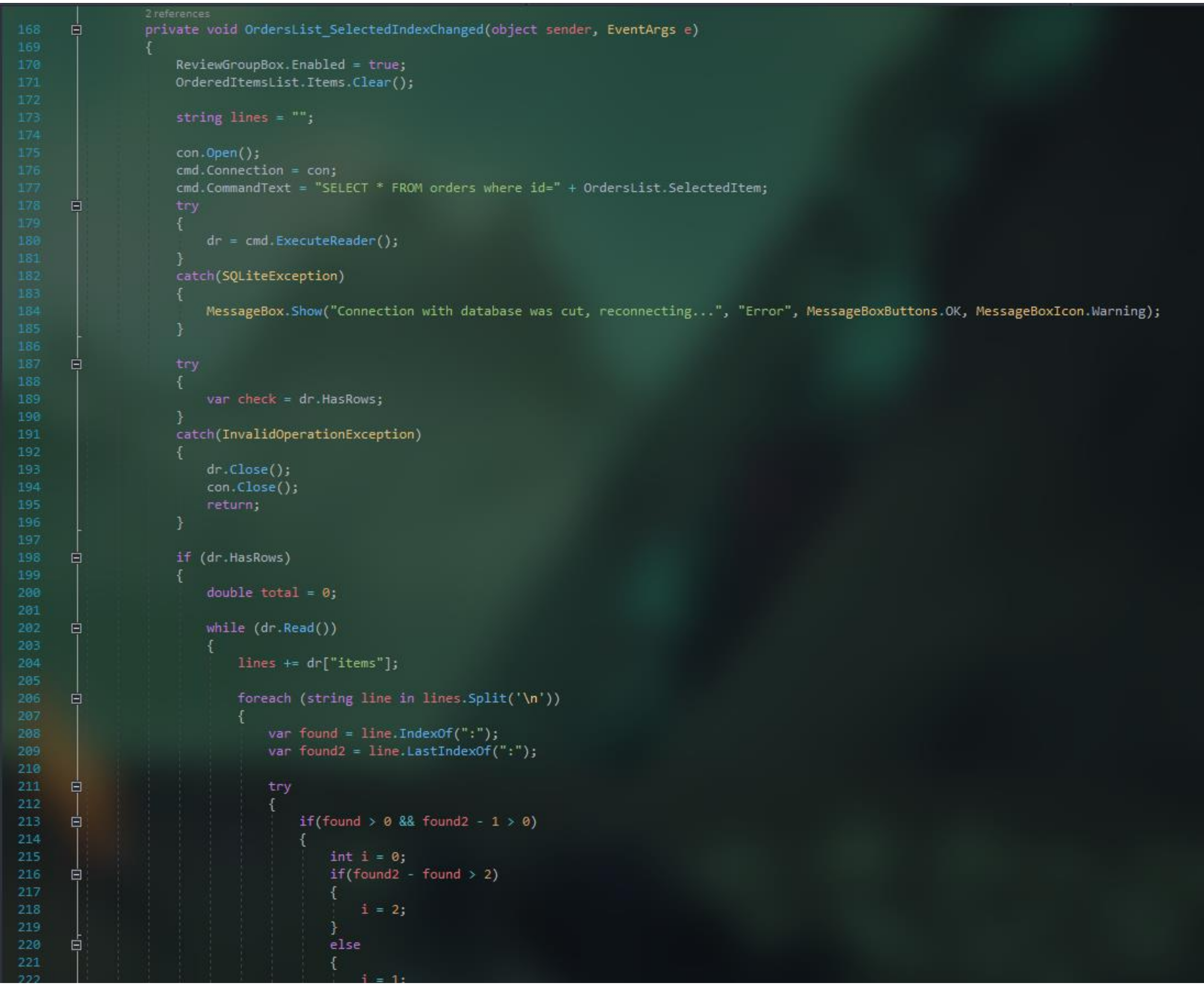
```
private void KitchenManagement_Load(object sender, EventArgs e)
{
    CompleteOrder.Enabled = false;
    ConfirmOrder.Enabled = false;
    CancelOrder.Enabled = false;

    foreach (TabPage tab in Tabs.TabPages)
    {
        tab.Enabled = false;
    }
    (Tabs.TabPages[Tabs.SelectedIndex] as TabPage).Enabled = true;

    foreach (TabPage tab in PendingOrdersTabs.TabPages)
    {
        tab.Enabled = false;
    }
    (PendingOrdersTabs.TabPages[PendingOrdersTabs.SelectedIndex] as TabPage).Enabled = true;

    foreach (TabPage tab in ManageRestaurantTabs.TabPages)
    {
        tab.Enabled = false;
    }
    (ManageRestaurantTabs.TabPages[ManageRestaurantTabs.SelectedIndex] as TabPage).Enabled = true;
}

1 reference
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (!isLock)
        Application.Exit();
    else
    {
        MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

1 reference
private void bACKToolStripMenuItem_Click(object sender, EventArgs e)
{
    if(!isLock)
        this.Close();
    else
    {
        MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

1 reference
private void Tabs_SelectedIndexChanged(object sender, EventArgs e)
{
    foreach (TabPage tab in Tabs.TabPages)
    {
        tab.Enabled = false;
    }
    (Tabs.TabPages[Tabs.SelectedIndex] as TabPage).Enabled = true;
}
```

- Load event disables buttons for cashier side, and disables all tab navigation (unless user is admin)
- exitToolStripMenuItem_Click checks if navigation is locked, if not user can exit.
- Tabs_SelectedIndexChanged changes the only enabled tabpage depending on which page the user on (and if it’s lock screen is on or not).



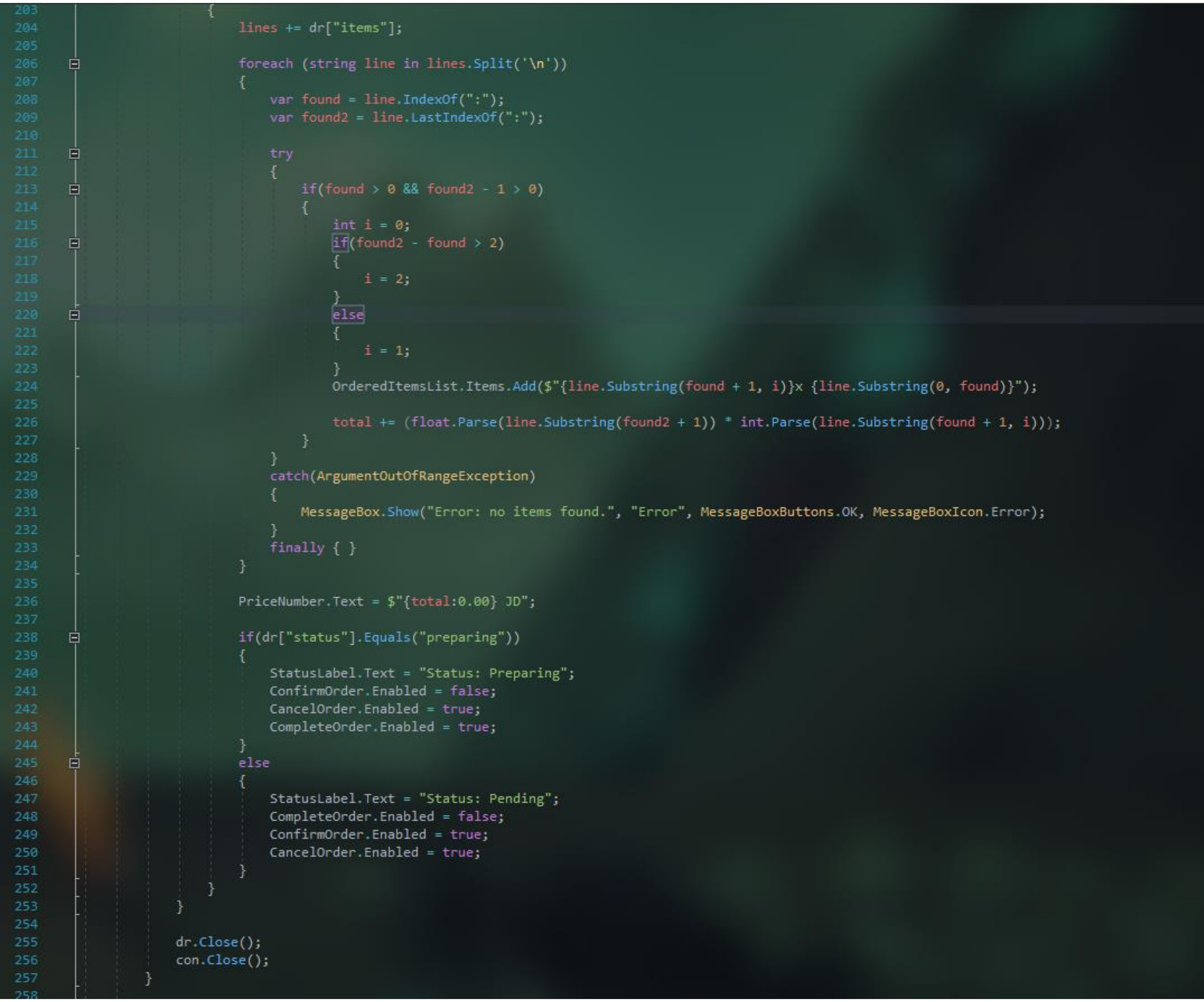
```
        1 reference
        private void adminToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if (!isAdmin)
            {
                lockScreenToolStripMenuItem.Enabled = false;

                PasswordChecker passCheck = new PasswordChecker();
                passCheck.ShowDialog();

                if (passCheck.DialogResult.Equals(DialogResult.Yes))
                {
                    isAdmin = true;
                    MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
            else
            {
                lockScreenToolStripMenuItem.Enabled = true;
            }

        }

        1 reference
        private void Tabs_Selecting(object sender, TabControlCancelEventArgs e)
        {
            if (!e.TabPage.Enabled && isLock)
            {
                MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
                e.Cancel = true;
            }
        }

        1 reference
        private void offToolStripMenuItem_Click(object sender, EventArgs e)
        {
            isLock = false;
        }

        1 reference
        private void onToolStripMenuItem_Click(object sender, EventArgs e)
        {
            isLock = true;
        }

        1 reference
        private void PendingOrdersTabs_Selecting(object sender, TabControlCancelEventArgs e)
        {
            if (!e.TabPage.Enabled && isLock)
            {
                MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
                e.Cancel = true;
            }
        }

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

- adminToolStripMenuItem_Click checks if user is admin, if not create a prompt asking for password. If isAdmin true then they are able to sign out or enable/disable lock screen feature.
- Tabs_Selecting checks if lock screen is enabled/disabled and cancels tab navigation if lock screen is enabled.



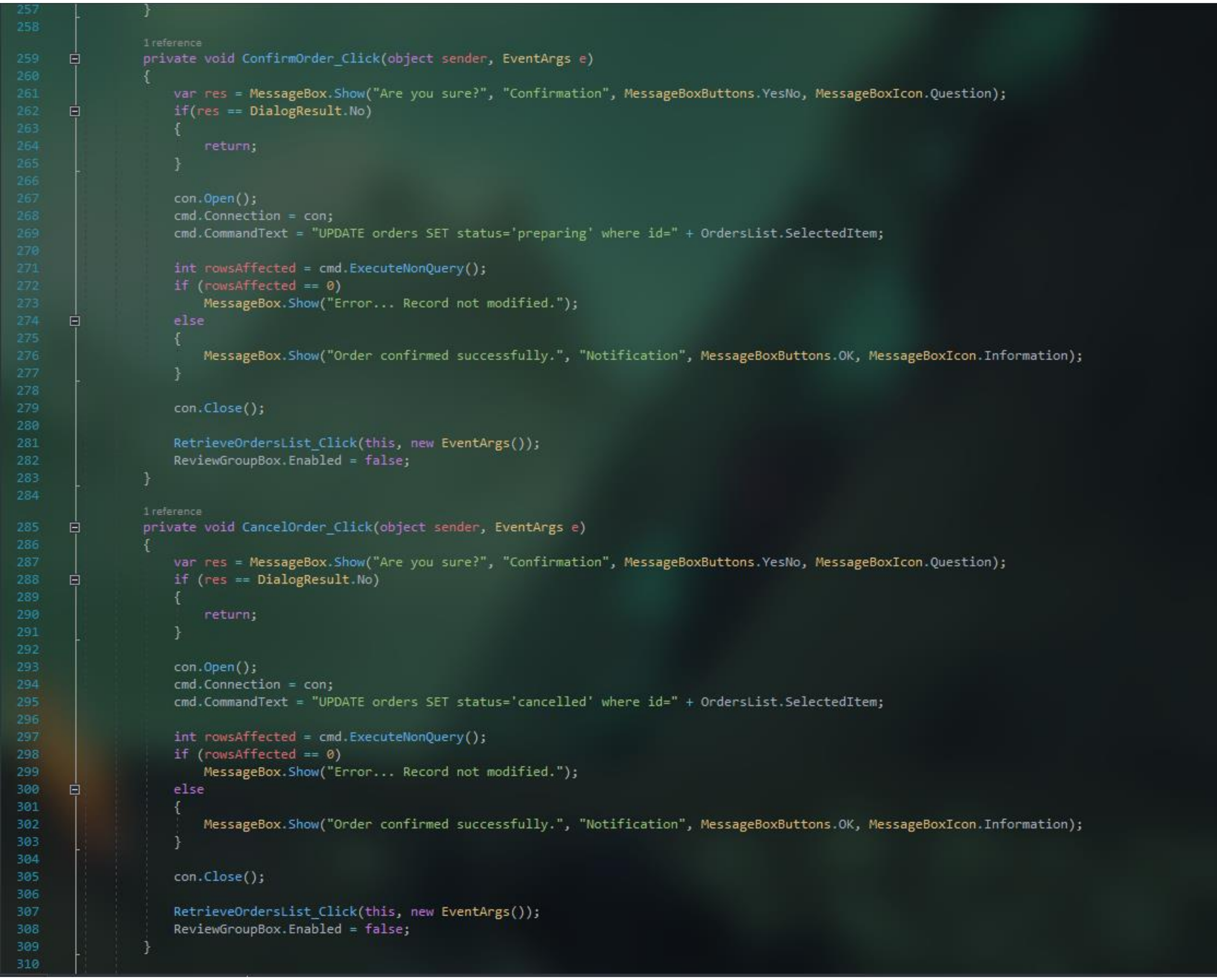
```
        2 references
        private void OrdersList_SelectedIndexChanged(object sender, EventArgs e)
        {
            ReviewGroupBox.Enabled = true;
            OrderedItemsList.Items.Clear();

            string lines = "";

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM orders where id=" + OrdersList.SelectedItem;
            try
            {
                dr = cmd.ExecuteReader();
            }
            catch(SQLiteException)
            {
                MessageBox.Show("Connection with database was cut, reconnecting...", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }

            try
            {
                var check = dr.HasRows;
            }
            catch(InvalidOperationException)
            {
                dr.Close();
                con.Close();
                return;
            }

            if (dr.HasRows)
            {
                double total = 0;

                while (dr.Read())
                {
                    lines += dr["items"];

                    foreach (string line in lines.Split('\n'))
                    {
                        var found = line.IndexOf(":");
                        var found2 = line.LastIndexOf(":");

                        try
                        {
                            if(found > 0 && found2 - 1 > 0)
                            {
                                int i = 0;
                                if(found2 - found > 2)
                                {
                                    i = 2;
                                }
                                else
                                {
                                    i = 1;
```

- Checks if the list item selected is changed, if yes then it connects to database and retrieves the items ordered depending on the order ID. Try and catch has been used here due to datareader method having connection errors. String methods have been used in order to get each item from the order (since it was boxed without nice styling).



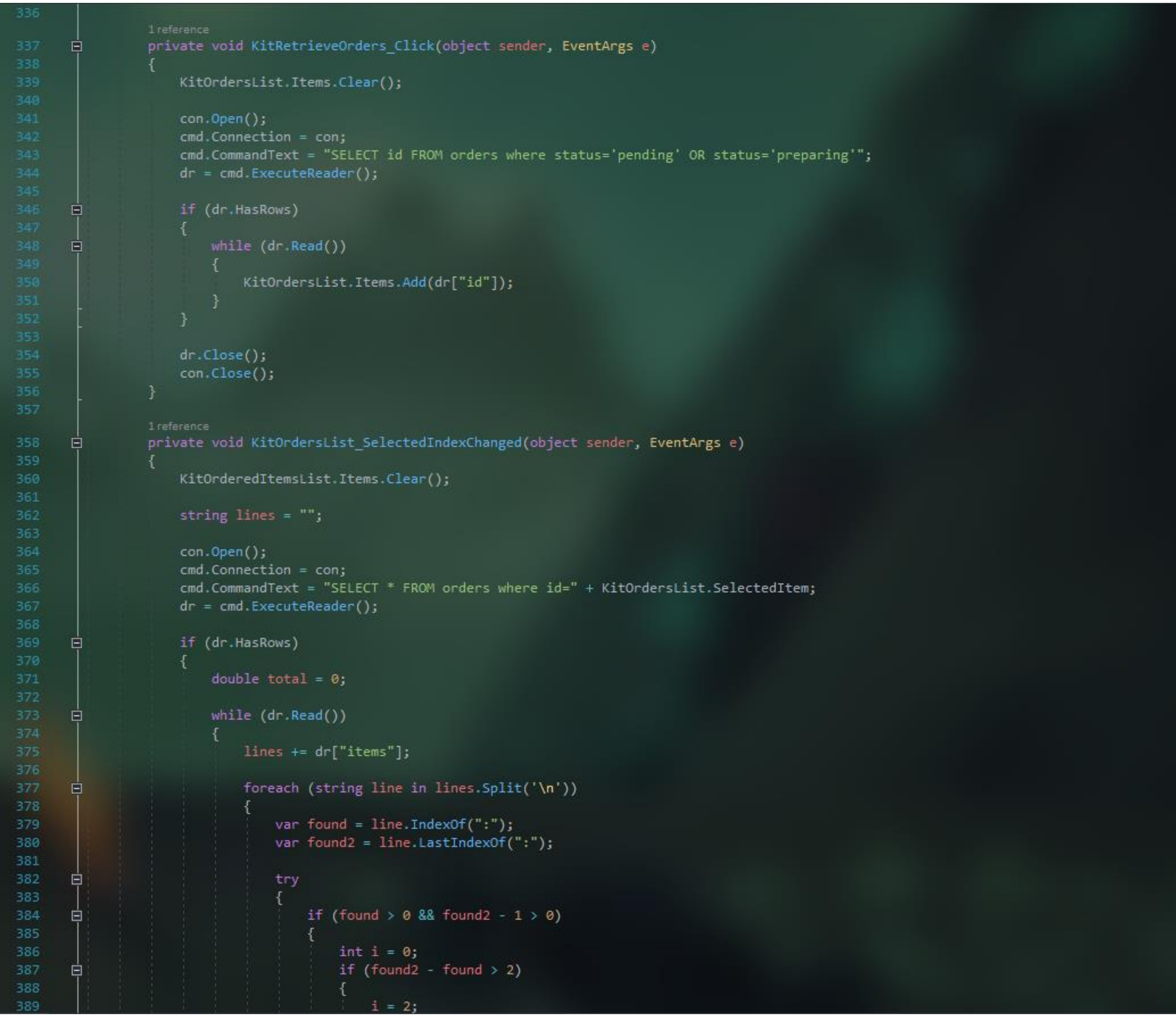
```
        {
            lines += dr["items"];

            foreach (string line in lines.Split('\n'))
            {
                var found = line.IndexOf(":");
                var found2 = line.LastIndexOf(":");

                try
                {
                    if(found > 0 && found2 - 1 > 0)
                    {
                        int i = 0;
                        if(found2 - found > 2)
                        {
                            i = 2;
                        }
                        else
                        {
                            i = 1;
                        }
                        OrderedItemsList.Items.Add($"{line.Substring(found + 1, i)}x {line.Substring(0, found)}");

                        total += (float.Parse(line.Substring(found2 + 1)) * int.Parse(line.Substring(found + 1, i)));
                    }
                }
                catch(ArgumentOutOfRangeException)
                {
                    MessageBox.Show("Error: no items found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
                finally { }
            }

            PriceNumber.Text = $"{total:0.00} JD";

            if(dr["status"].Equals("preparing"))
            {
                StatusLabel.Text = "Status: Preparing";
                ConfirmOrder.Enabled = false;
                CancelOrder.Enabled = true;
                CompleteOrder.Enabled = true;
            }
            else
            {
                StatusLabel.Text = "Status: Pending";
                CompleteOrder.Enabled = false;
                ConfirmOrder.Enabled = true;
                CancelOrder.Enabled = true;
            }
        }
    }

    dr.Close();
    con.Close();
}
```

- Total gets the total cost for the order and displays it in a label. It then checks which status the order is in, and depending on that the buttons enabled for the cashier side will change.



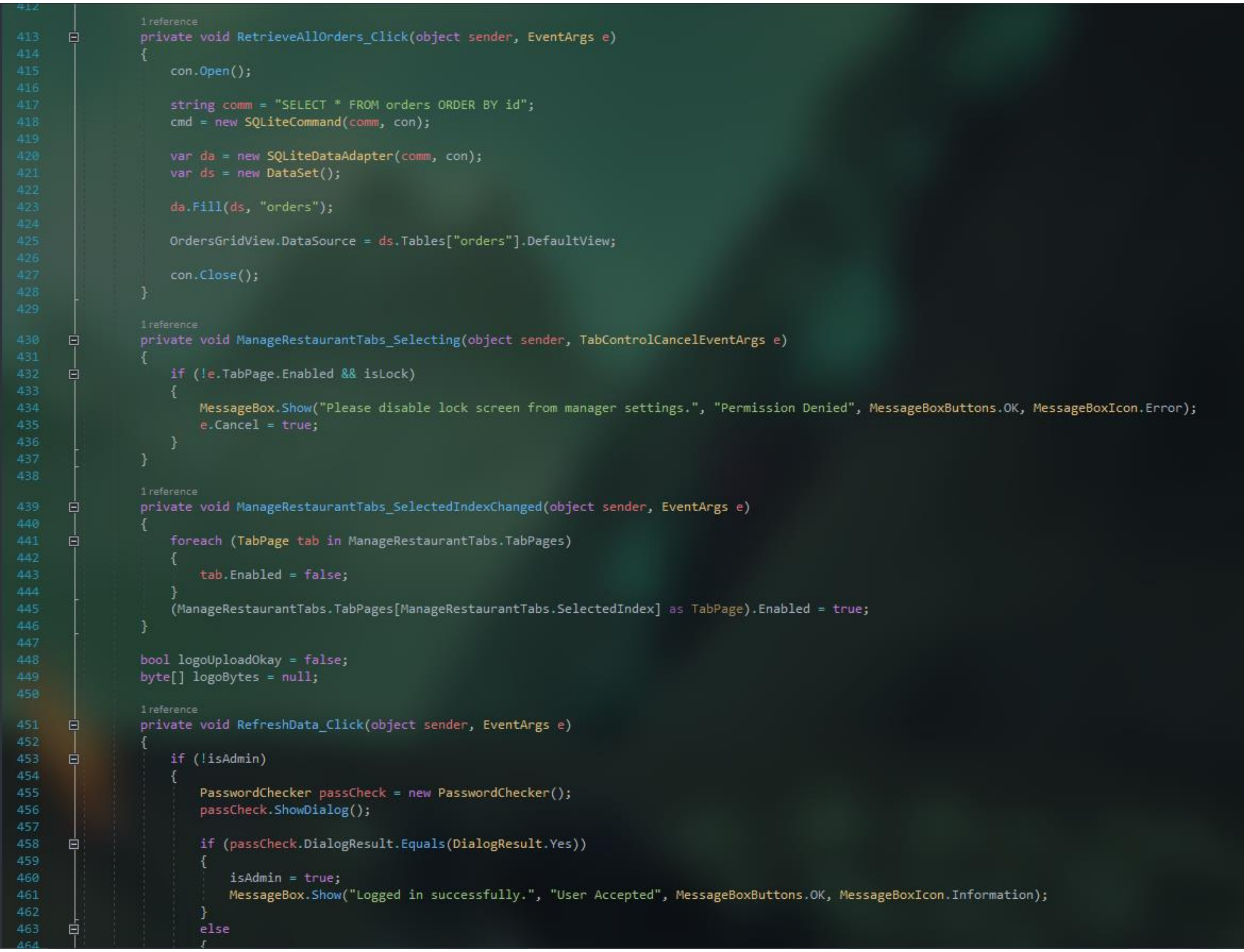
```
        }

        1 reference
        private void ConfirmOrder_Click(object sender, EventArgs e)
        {
            var res = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if(res == DialogResult.No)
            {
                return;
            }

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "UPDATE orders SET status='preparing' where id=" + OrdersList.SelectedItem;

            int rowsAffected = cmd.ExecuteNonQuery();
            if (rowsAffected == 0)
                MessageBox.Show("Error... Record not modified.");
            else
            {
                MessageBox.Show("Order confirmed successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            con.Close();

            RetrieveOrdersList_Click(this, new EventArgs());
            ReviewGroupBox.Enabled = false;
        }

        1 reference
        private void CancelOrder_Click(object sender, EventArgs e)
        {
            var res = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (res == DialogResult.No)
            {
                return;
            }

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "UPDATE orders SET status='cancelled' where id=" + OrdersList.SelectedItem;

            int rowsAffected = cmd.ExecuteNonQuery();
            if (rowsAffected == 0)
                MessageBox.Show("Error... Record not modified.");
            else
            {
                MessageBox.Show("Order confirmed successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            con.Close();

            RetrieveOrdersList_Click(this, new EventArgs());
            ReviewGroupBox.Enabled = false;
        }
```

- Both click events act the same, it checks if the user is sure with their action, if not return. After that it connects to the databse and updates the order record to the required status (depending on the order id). Finally, it disables the modification groupbox, and refreshes the orders view list.



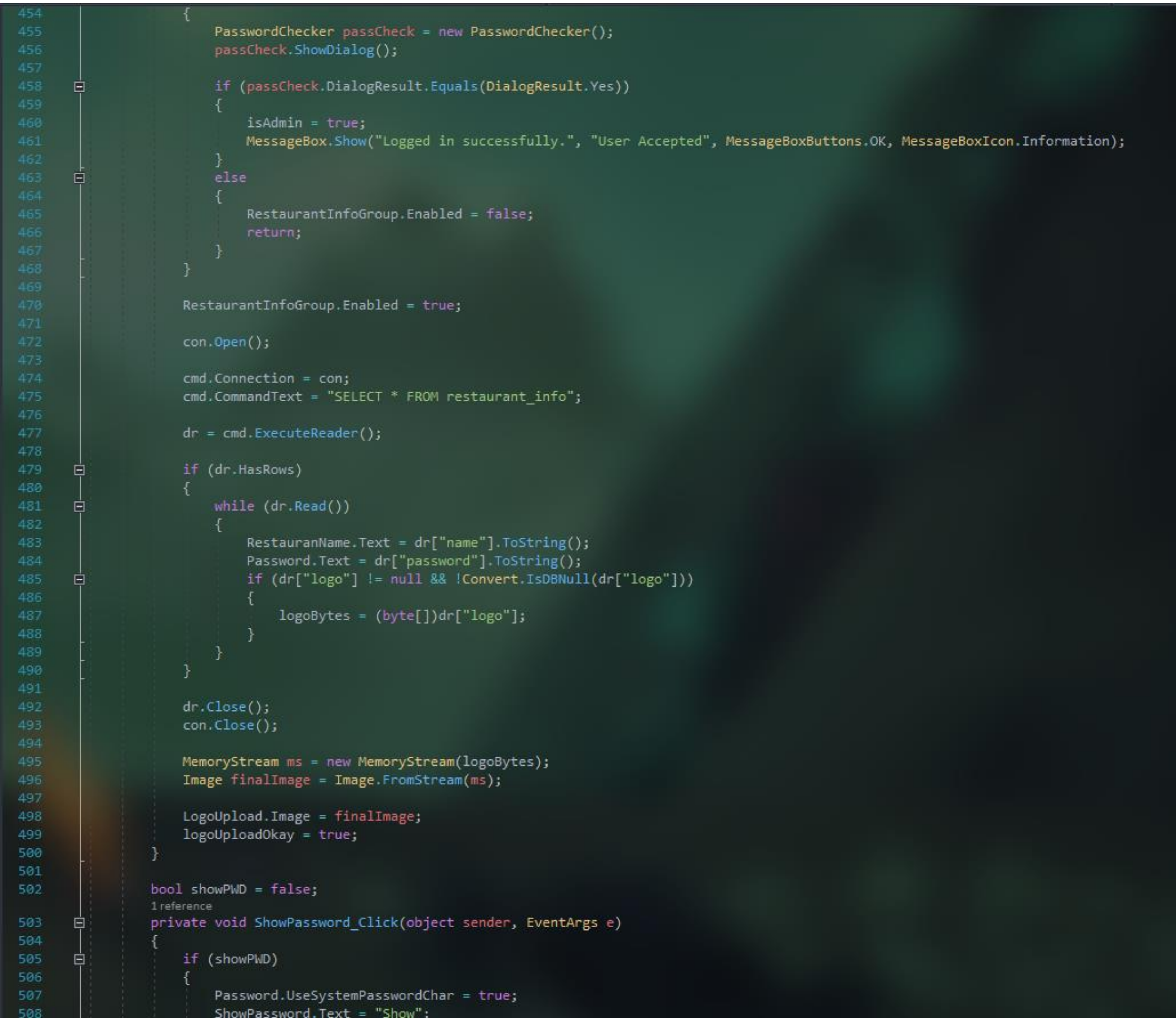
```
        1 reference
        private void KitRetrieveOrders_Click(object sender, EventArgs e)
        {
            KitOrdersList.Items.Clear();

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT id FROM orders where status='pending' OR status='preparing'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    KitOrdersList.Items.Add(dr["id"]);
                }
            }

            dr.Close();
            con.Close();
        }

        1 reference
        private void KitOrdersList_SelectedIndexChanged(object sender, EventArgs e)
        {
            KitOrderedItemsList.Items.Clear();

            string lines = "";

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM orders where id=" + KitOrdersList.SelectedItem;
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                double total = 0;

                while (dr.Read())
                {
                    lines += dr["items"];

                    foreach (string line in lines.Split('\n'))
                    {
                        var found = line.IndexOf(":");
                        var found2 = line.LastIndexOf(":");

                        try
                        {
                            if (found > 0 && found2 - 1 > 0)
                            {
                                int i = 0;
                                if (found2 - found > 2)
                                {
                                    i = 2;
```

- KitRetrieveOrders_Click gets all the orders records that's have a status of "preparing" and display it for the kitchen staff.
- Selected index changed event works the same way as the cashier side event.



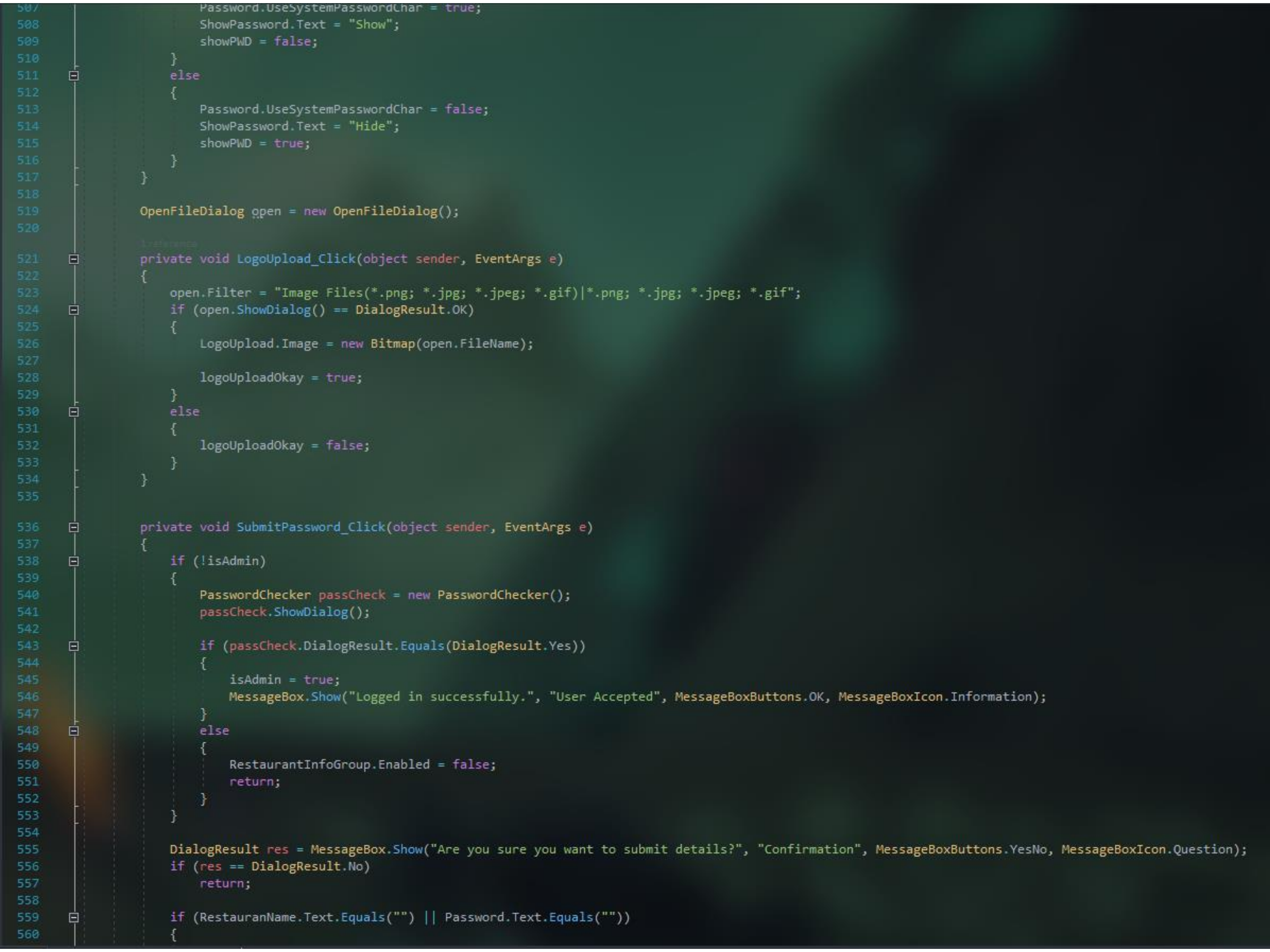
```
        1 reference
        private void RetrieveAllOrders_Click(object sender, EventArgs e)
        {
            con.Open();

            string comm = "SELECT * FROM orders ORDER BY id";
            cmd = new SQLiteCommand(comm, con);

            var da = new SQLiteDataAdapter(comm, con);
            var ds = new DataSet();

            da.Fill(ds, "orders");

            OrdersGridView.DataSource = ds.Tables["orders"].DefaultView;

            con.Close();
        }

        1 reference
        private void ManageRestaurantTabs_Selecting(object sender, TabControlCancelEventArgs e)
        {
            if (!e.TabPage.Enabled && isLock)
            {
                MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
                e.Cancel = true;
            }
        }

        1 reference
        private void ManageRestaurantTabs_SelectedIndexChanged(object sender, EventArgs e)
        {
            foreach (TabPage tab in ManageRestaurantTabs.TabPages)
            {
                tab.Enabled = false;
            }
            (ManageRestaurantTabs.TabPages[ManageRestaurantTabs.SelectedIndex] as TabPage).Enabled = true;
        }

        bool logoUploadOkay = false;
        byte[] logoBytes = null;

        1 reference
        private void RefreshData_Click(object sender, EventArgs e)
        {
            if (!isAdmin)
            {
                PasswordChecker passCheck = new PasswordChecker();
                passCheck.ShowDialog();

                if (passCheck.DialogResult.Equals(DialogResult.Yes))
                {
                    isAdmin = true;
                    MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
```

- RetrieveAllOrders_Click gets all the orders in the database with any status for the managers to view, a potential option could be to output results in a excel sheet.
- RefreshData_Click checks if user an admin, if not then it'll ask for user to login with manager password.



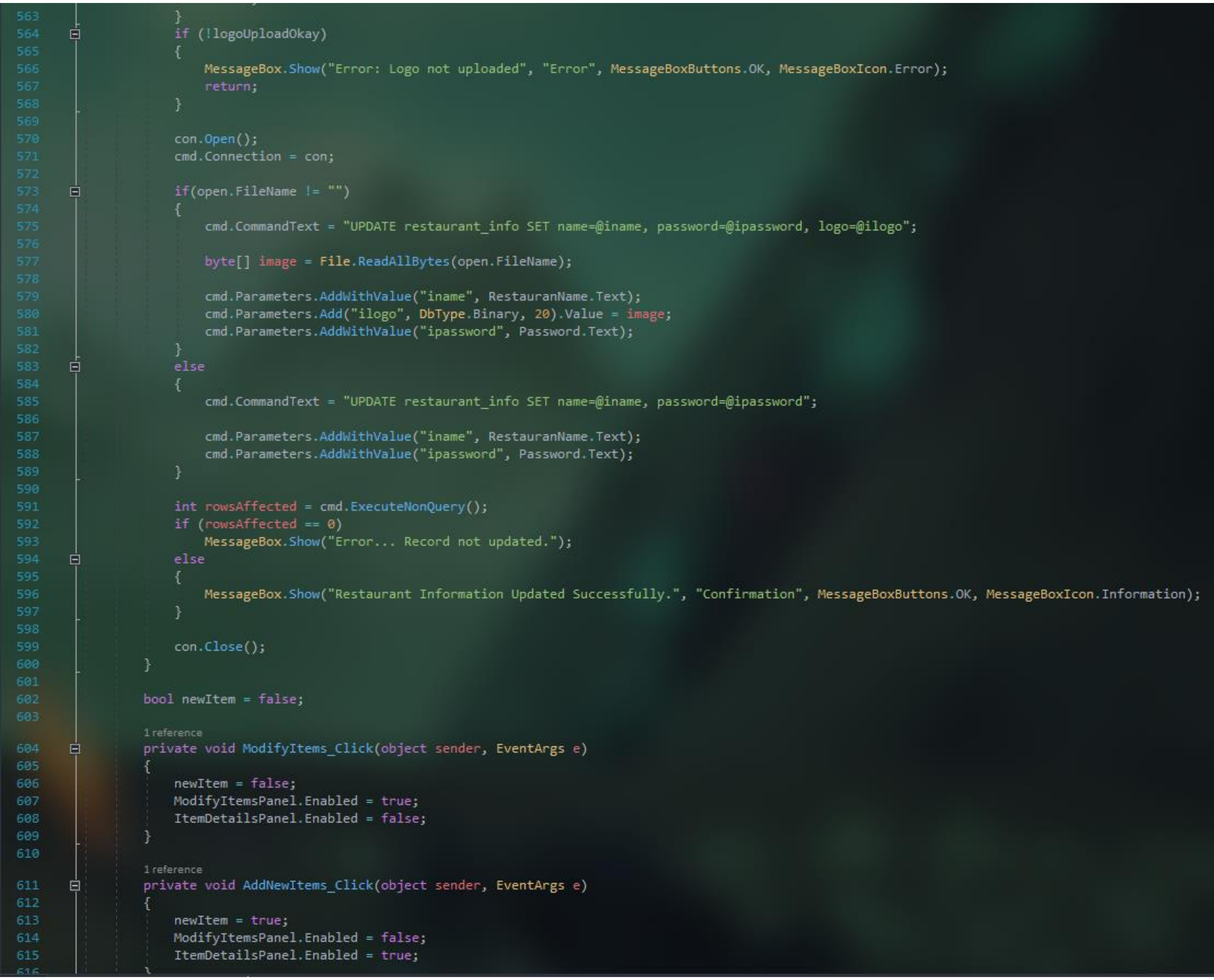
```
            {
                PasswordChecker passCheck = new PasswordChecker();
                passCheck.ShowDialog();

                if (passCheck.DialogResult.Equals(DialogResult.Yes))
                {
                    isAdmin = true;
                    MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    RestaurantInfoGroup.Enabled = false;
                    return;
                }
            }

            RestaurantInfoGroup.Enabled = true;

            con.Open();

            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM restaurant_info";

            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    RestauranName.Text = dr["name"].ToString();
                    Password.Text = dr["password"].ToString();
                    if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
                    {
                        logoBytes = (byte[])dr["logo"];
                    }
                }
            }

            dr.Close();
            con.Close();

            MemoryStream ms = new MemoryStream(logoBytes);
            Image finalImage = Image.FromStream(ms);

            LogoUpload.Image = finalImage;
            logoUploadOkay = true;
        }

        bool showPWD = false;
        1 reference
        private void ShowPassword_Click(object sender, EventArgs e)
        {
            if (showPWD)
            {
                Password.UseSystemPasswordChar = true;
                ShowPassword.Text = "Show";
```

- RefreshData_Click continuation: if user is an admin, we get restaurant info from database and display it in the fields given; the user then can change any data and submit it to database.



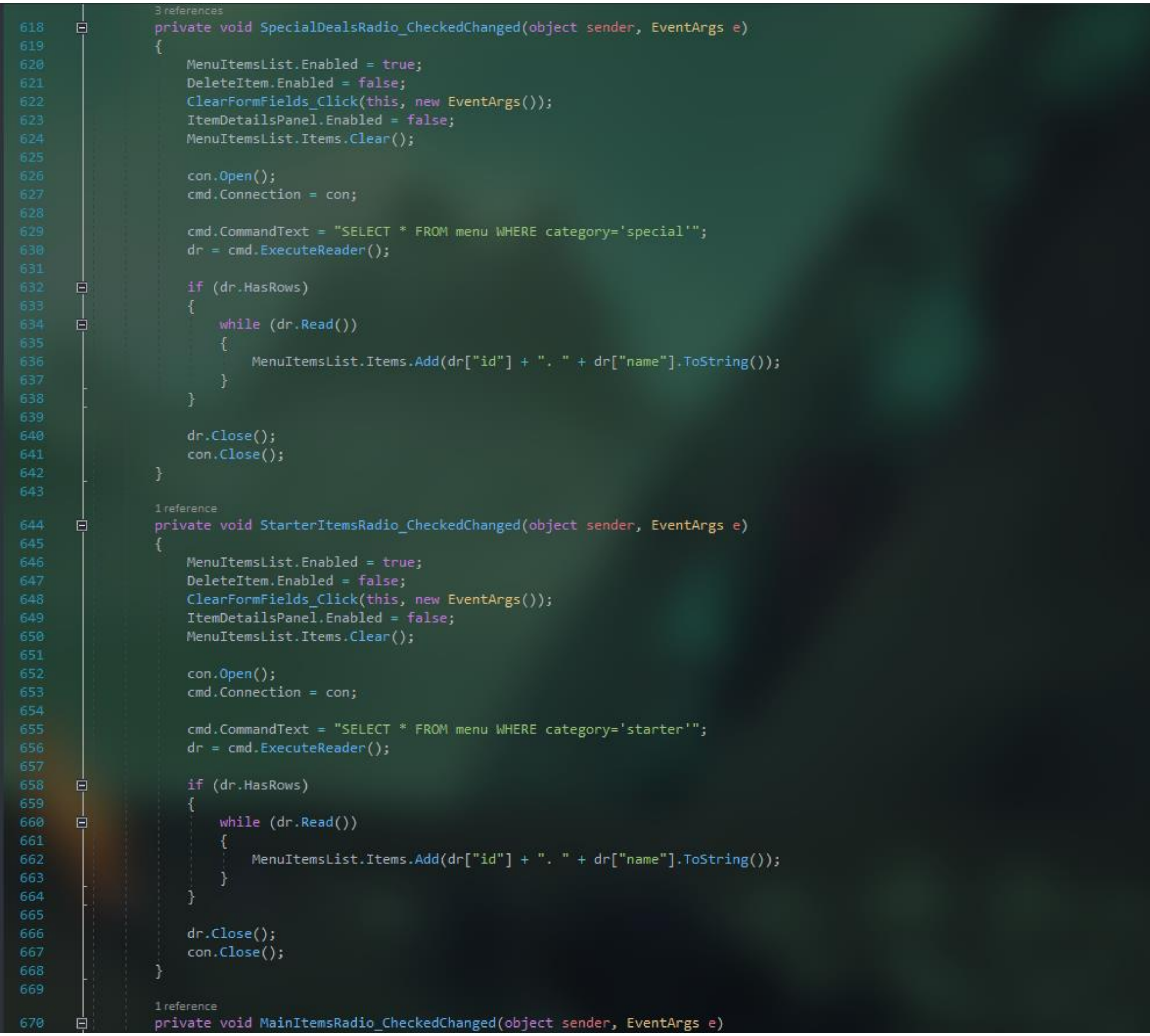
```
                Password.UseSystemPasswordChar = true;
                ShowPassword.Text = "Show";
                showPWD = false;
            }
            else
            {
                Password.UseSystemPasswordChar = false;
                ShowPassword.Text = "Hide";
                showPWD = true;
            }
        }

        OpenFileDialog open = new OpenFileDialog();

        private void LogoUpload_Click(object sender, EventArgs e)
        {
            open.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
            if (open.ShowDialog() == DialogResult.OK)
            {
                LogoUpload.Image = new Bitmap(open.FileName);

                logoUploadOkay = true;
            }
            else
            {
                logoUploadOkay = false;
            }
        }

        private void SubmitPassword_Click(object sender, EventArgs e)
        {
            if (!isAdmin)
            {
                PasswordChecker passCheck = new PasswordChecker();
                passCheck.ShowDialog();

                if (passCheck.DialogResult.Equals(DialogResult.Yes))
                {
                    isAdmin = true;
                    MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    RestaurantInfoGroup.Enabled = false;
                    return;
                }
            }

            DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (res == DialogResult.No)
                return;

            if (RestauranName.Text.Equals("") || Password.Text.Equals(""))
            {
```

- SubmitPassword_Click checks if user an admin again. Then we check to make sure user is sure they want to submit details.



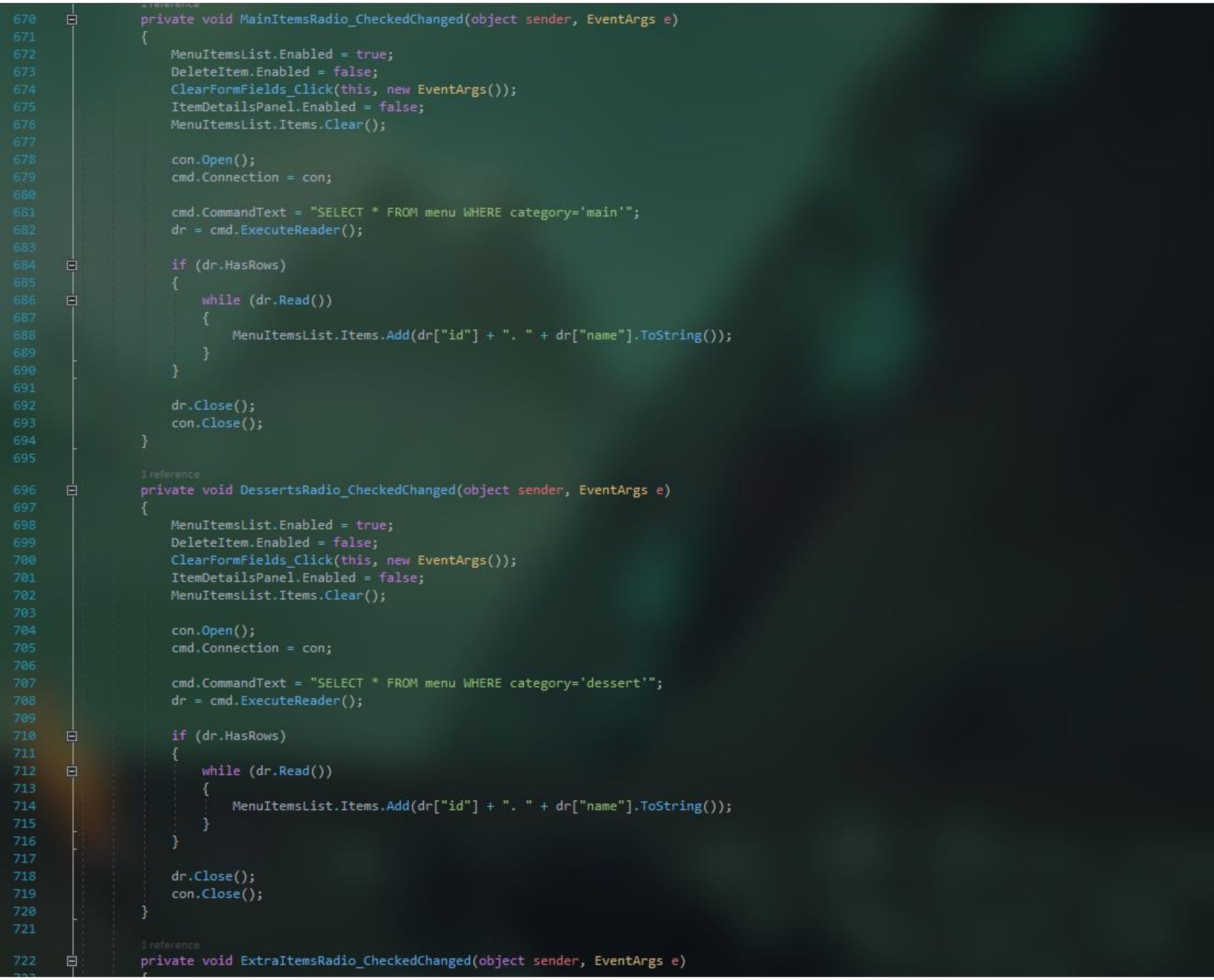
```
            }
            if (!logoUploadOkay)
            {
                MessageBox.Show("Error: Logo not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            con.Open();
            cmd.Connection = con;

            if(open.FileName != "")
            {
                cmd.CommandText = "UPDATE restaurant_info SET name=@iname, password=@ipassword, logo=@ilogo";

                byte[] image = File.ReadAllBytes(open.FileName);

                cmd.Parameters.AddWithValue("iname", RestauranName.Text);
                cmd.Parameters.Add("ilogo", DbType.Binary, 20).Value = image;
                cmd.Parameters.AddWithValue("ipassword", Password.Text);
            }
            else
            {
                cmd.CommandText = "UPDATE restaurant_info SET name=@iname, password=@ipassword";

                cmd.Parameters.AddWithValue("iname", RestauranName.Text);
                cmd.Parameters.AddWithValue("ipassword", Password.Text);
            }

            int rowsAffected = cmd.ExecuteNonQuery();
            if (rowsAffected == 0)
                MessageBox.Show("Error... Record not updated.");
            else
            {
                MessageBox.Show("Restaurant Information Updated Successfully.", "Confirmation", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }

            con.Close();
        }

        bool newItem = false;

        1 reference
        private void ModifyItems_Click(object sender, EventArgs e)
        {
            newItem = false;
            ModifyItemsPanel.Enabled = true;
            ItemDetailsPanel.Enabled = false;
        }

        1 reference
        private void AddNewItems_Click(object sender, EventArgs e)
        {
            newItem = true;
            ModifyItemsPanel.Enabled = false;
            ItemDetailsPanel.Enabled = true;
```

- Before submitting, we check if user uploaded a new logo or wants to keep the old one, then send it to database accordingly.
- ModifyItems_Click enables medication panel
- AddNewItems_Click enables new item form fields to submit.



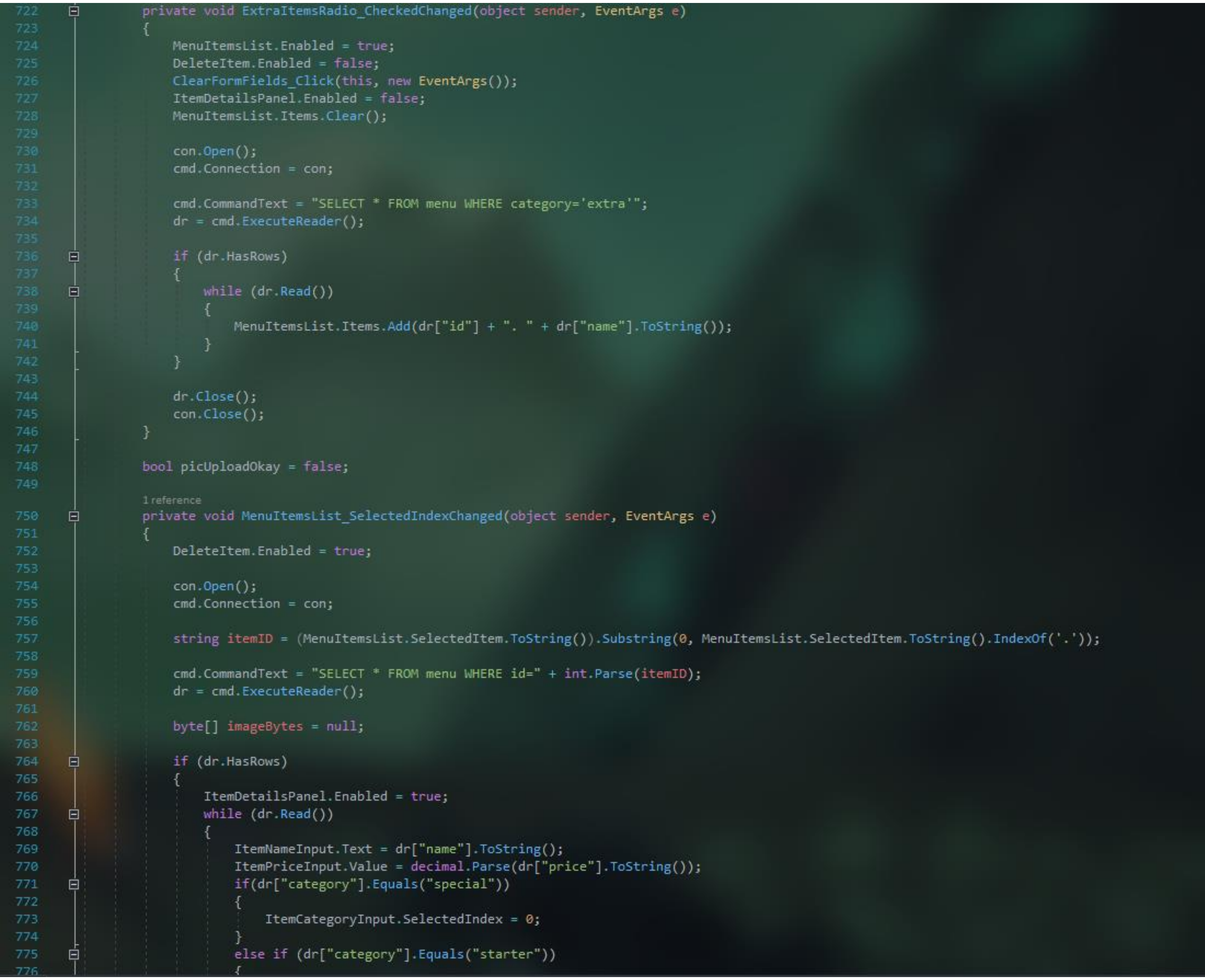
```
        3 references
        private void SpecialDealsRadio_CheckedChanged(object sender, EventArgs e)
        {
            MenuItemsList.Enabled = true;
            DeleteItem.Enabled = false;
            ClearFormFields_Click(this, new EventArgs());
            ItemDetailsPanel.Enabled = false;
            MenuItemsList.Items.Clear();

            con.Open();
            cmd.Connection = con;

            cmd.CommandText = "SELECT * FROM menu WHERE category='special'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItemsList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
                }
            }

            dr.Close();
            con.Close();
        }

        1 reference
        private void StarterItemsRadio_CheckedChanged(object sender, EventArgs e)
        {
            MenuItemsList.Enabled = true;
            DeleteItem.Enabled = false;
            ClearFormFields_Click(this, new EventArgs());
            ItemDetailsPanel.Enabled = false;
            MenuItemsList.Items.Clear();

            con.Open();
            cmd.Connection = con;

            cmd.CommandText = "SELECT * FROM menu WHERE category='starter'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItemsList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
                }
            }

            dr.Close();
            con.Close();
        }

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

- Radio button shows menu items in the list depending on which category was chosen from the group box.



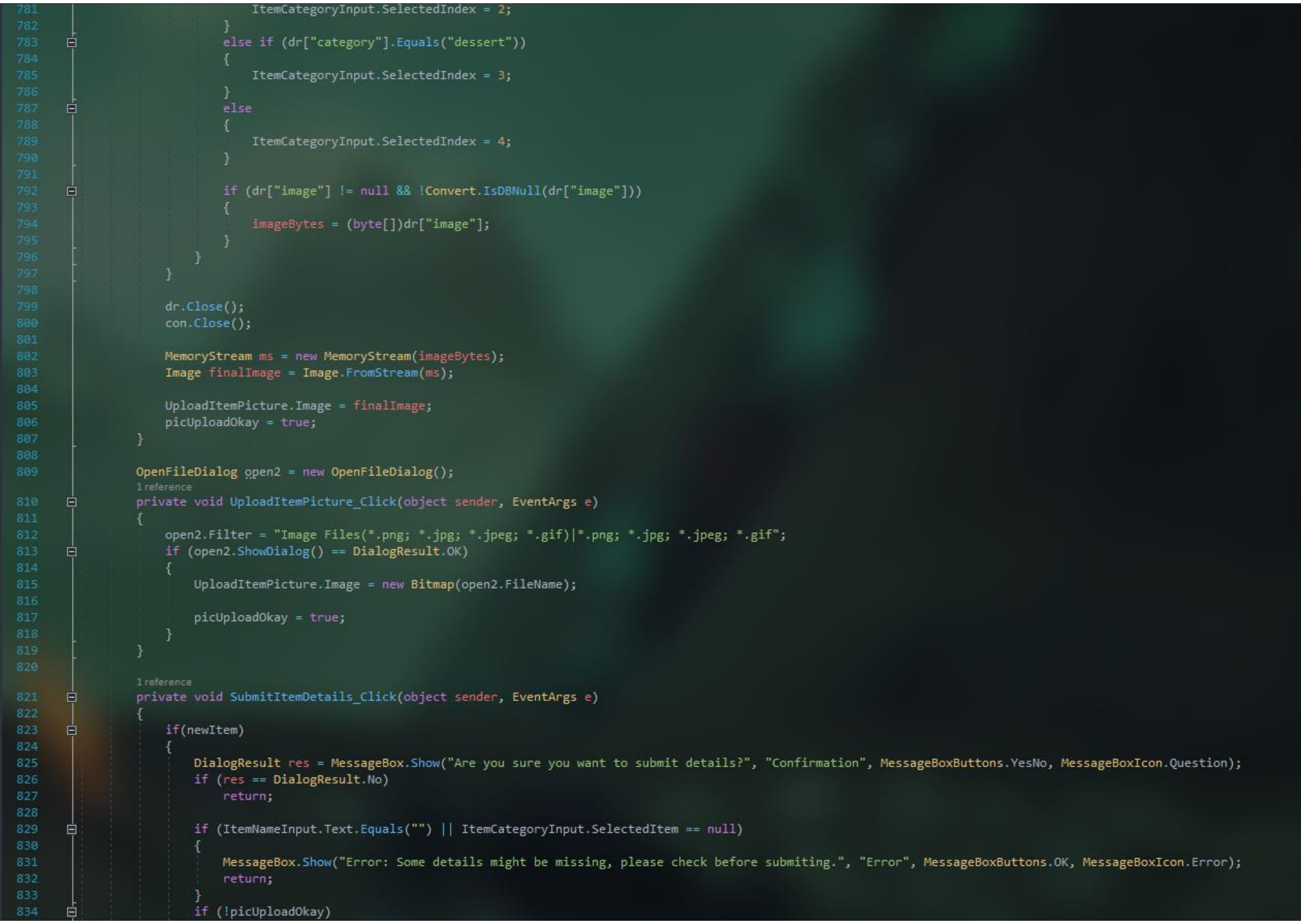
```
        private void MainItemsRadio_CheckedChanged(object sender, EventArgs e)
        {
            MenuItemsList.Enabled = true;
            DeleteItem.Enabled = false;
            ClearFormFields_Click(this, new EventArgs());
            ItemDetailsPanel.Enabled = false;
            MenuItemsList.Items.Clear();

            con.Open();
            cmd.Connection = con;

            cmd.CommandText = "SELECT * FROM menu WHERE category='main'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItemsList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
                }
            }

            dr.Close();
            con.Close();
        }

        1 reference
        private void DessertsRadio_CheckedChanged(object sender, EventArgs e)
        {
            MenuItemsList.Enabled = true;
            DeleteItem.Enabled = false;
            ClearFormFields_Click(this, new EventArgs());
            ItemDetailsPanel.Enabled = false;
            MenuItemsList.Items.Clear();

            con.Open();
            cmd.Connection = con;

            cmd.CommandText = "SELECT * FROM menu WHERE category='dessert'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItemsList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
                }
            }

            dr.Close();
            con.Close();
        }

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

- Continuation of the above.



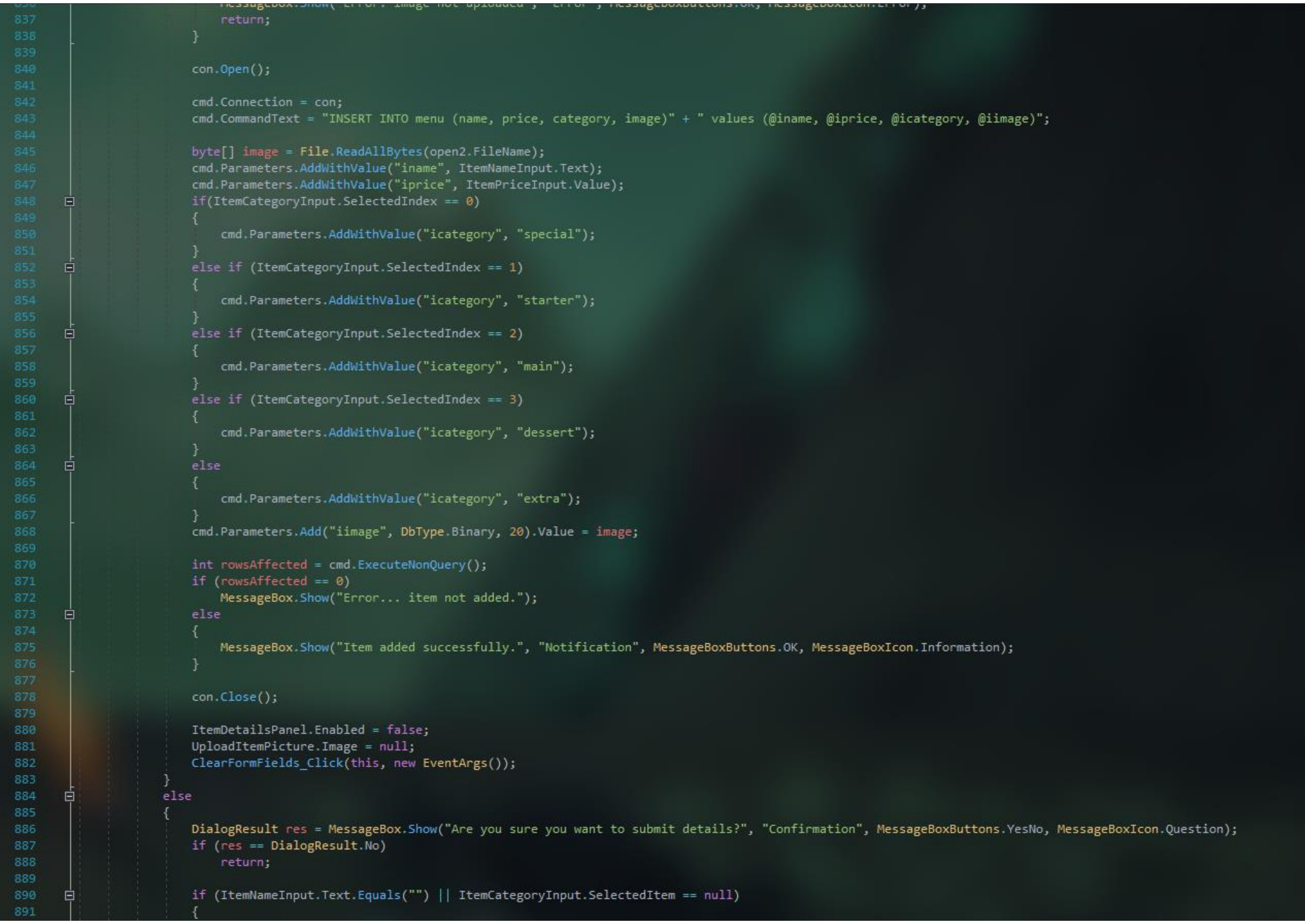
```
        private void ExtraItemsRadio_CheckedChanged(object sender, EventArgs e)
        {
            MenuItemsList.Enabled = true;
            DeleteItem.Enabled = false;
            ClearFormFields_Click(this, new EventArgs());
            ItemDetailsPanel.Enabled = false;
            MenuItemsList.Items.Clear();

            con.Open();
            cmd.Connection = con;

            cmd.CommandText = "SELECT * FROM menu WHERE category='extra'";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItemsList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
                }
            }

            dr.Close();
            con.Close();
        }

        bool picUploadOkay = false;

        1 reference
        private void MenuItemsList_SelectedIndexChanged(object sender, EventArgs e)
        {
            DeleteItem.Enabled = true;

            con.Open();
            cmd.Connection = con;

            string itemID = (MenuItemsList.SelectedItem.ToString()).Substring(0, MenuItemsList.SelectedItem.ToString().IndexOf('.'));

            cmd.CommandText = "SELECT * FROM menu WHERE id=" + int.Parse(itemID);
            dr = cmd.ExecuteReader();

            byte[] imageBytes = null;

            if (dr.HasRows)
            {
                ItemDetailsPanel.Enabled = true;
                while (dr.Read())
                {
                    ItemNameInput.Text = dr["name"].ToString();
                    ItemPriceInput.Value = decimal.Parse(dr["price"].ToString());
                    if(dr["category"].Equals("special"))
                    {
                        ItemCategoryInput.SelectedIndex = 0;
                    }
                    else if (dr["category"].Equals("starter"))
                    {
```

- If one of the menu items were selected from the list, then it takes the item id from the string in list and searches for it in the database. The database then gets it in order for user to view the old details they could change in the form field.



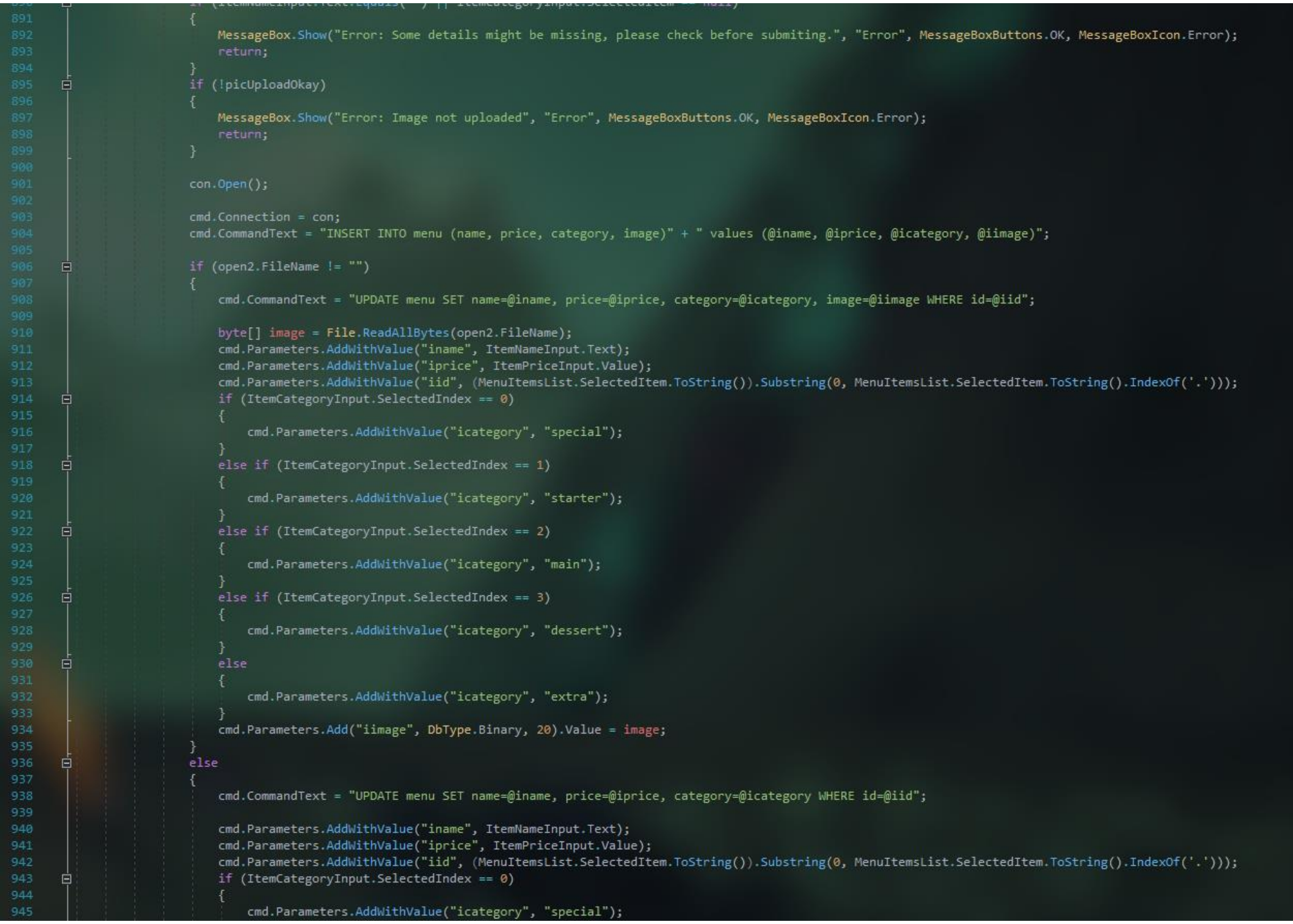
```
                        ItemCategoryInput.SelectedIndex = 2;
                    }
                    else if (dr["category"].Equals("dessert"))
                    {
                        ItemCategoryInput.SelectedIndex = 3;
                    }
                    else
                    {
                        ItemCategoryInput.SelectedIndex = 4;
                    }

                    if (dr["image"] != null && !Convert.IsDBNull(dr["image"]))
                    {
                        imageBytes = (byte[])dr["image"];
                    }
                }
            }

            dr.Close();
            con.Close();

            MemoryStream ms = new MemoryStream(imageBytes);
            Image finalImage = Image.FromStream(ms);

            UploadItemPicture.Image = finalImage;
            picUploadOkay = true;
        }

        OpenFileDialog open2 = new OpenFileDialog();
        1 reference
        private void UploadItemPicture_Click(object sender, EventArgs e)
        {
            open2.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
            if (open2.ShowDialog() == DialogResult.OK)
            {
                UploadItemPicture.Image = new Bitmap(open2.FileName);

                picUploadOkay = true;
            }
        }

        1 reference
        private void SubmitItemDetails_Click(object sender, EventArgs e)
        {
            if(newItem)
            {
                DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
                if (res == DialogResult.No)
                    return;

                if (ItemNameInput.Text.Equals("") || ItemCategoryInput.SelectedItem == null)
                {
                    MessageBox.Show("Error: Some details might be missing, please check before submiting.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                if (!picUploadOkay)
```

- User can upload an image file type for the menu item they’re adding.
- User can submit details; we check if they’re sure with their actions.



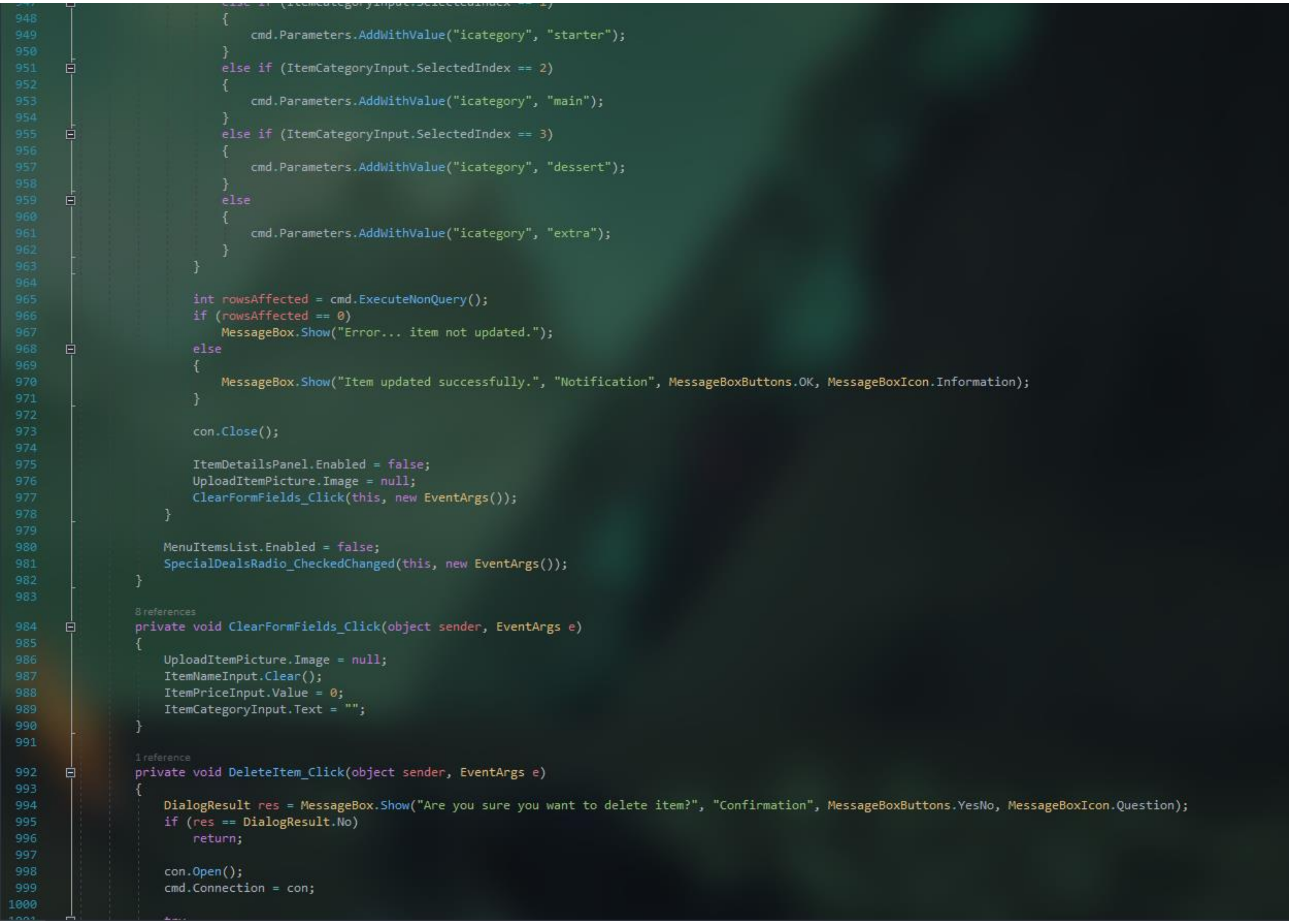
```
        MessageBox.Show("Error: Image not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    con.Open();

    cmd.Connection = con;
    cmd.CommandText = "INSERT INTO menu (name, price, category, image)" + " values (@iname, @iprice, @icategory, @iimage)";

    byte[] image = File.ReadAllBytes(open2.FileName);
    cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
    cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
    if(ItemCategoryInput.SelectedIndex == 0)
    {
        cmd.Parameters.AddWithValue("icategory", "special");
    }
    else if (ItemCategoryInput.SelectedIndex == 1)
    {
        cmd.Parameters.AddWithValue("icategory", "starter");
    }
    else if (ItemCategoryInput.SelectedIndex == 2)
    {
        cmd.Parameters.AddWithValue("icategory", "main");
    }
    else if (ItemCategoryInput.SelectedIndex == 3)
    {
        cmd.Parameters.AddWithValue("icategory", "dessert");
    }
    else
    {
        cmd.Parameters.AddWithValue("icategory", "extra");
    }
    cmd.Parameters.Add("iimage", DbType.Binary, 20).Value = image;

    int rowsAffected = cmd.ExecuteNonQuery();
    if (rowsAffected == 0)
        MessageBox.Show("Error... item not added.");
    else
    {
        MessageBox.Show("Item added successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    con.Close();

    ItemDetailsPanel.Enabled = false;
    UploadItemPicture.Image = null;
    ClearFormFields_Click(this, new EventArgs());
}
else
{
    DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (res == DialogResult.No)
        return;

    if (ItemNameInput.Text.Equals("") || ItemCategoryInput.SelectedItem == null)
    {
```

- If this is a new menu item then we'll insert the image and the other details directly to the database.
- If not, then we continue to else statement.



```
if (ItemNameInput.Text.Equals("") || ItemCategoryInput.SelectedItem == null)
{
    MessageBox.Show("Error: Some details might be missing, please check before submiting.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
if (!picUploadOkay)
{
    MessageBox.Show("Error: Image not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}

con.Open();

cmd.Connection = con;
cmd.CommandText = "INSERT INTO menu (name, price, category, image)" + " values (@iname, @iprice, @icategory, @iimage)";

if (open2.FileName != "")
{
    cmd.CommandText = "UPDATE menu SET name=@iname, price=@iprice, category=@icategory, image=@iimage WHERE id=@iid";

    byte[] image = File.ReadAllBytes(open2.FileName);
    cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
    cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
    cmd.Parameters.AddWithValue("iid", (MenuItemsList.SelectedItem.ToString()).Substring(0, MenuItemsList.SelectedItem.ToString().IndexOf('.')));
    if (ItemCategoryInput.SelectedIndex == 0)
    {
        cmd.Parameters.AddWithValue("icategory", "special");
    }
    else if (ItemCategoryInput.SelectedIndex == 1)
    {
        cmd.Parameters.AddWithValue("icategory", "starter");
    }
    else if (ItemCategoryInput.SelectedIndex == 2)
    {
        cmd.Parameters.AddWithValue("icategory", "main");
    }
    else if (ItemCategoryInput.SelectedIndex == 3)
    {
        cmd.Parameters.AddWithValue("icategory", "dessert");
    }
    else
    {
        cmd.Parameters.AddWithValue("icategory", "extra");
    }
    cmd.Parameters.Add("iimage", DbType.Binary, 20).Value = image;
}
else
{
    cmd.CommandText = "UPDATE menu SET name=@iname, price=@iprice, category=@icategory WHERE id=@iid";

    cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
    cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
    cmd.Parameters.AddWithValue("iid", (MenuItemsList.SelectedItem.ToString()).Substring(0, MenuItemsList.SelectedItem.ToString().IndexOf('.')));
    if (ItemCategoryInput.SelectedIndex == 0)
    {
        cmd.Parameters.AddWithValue("icategory", "special");
```

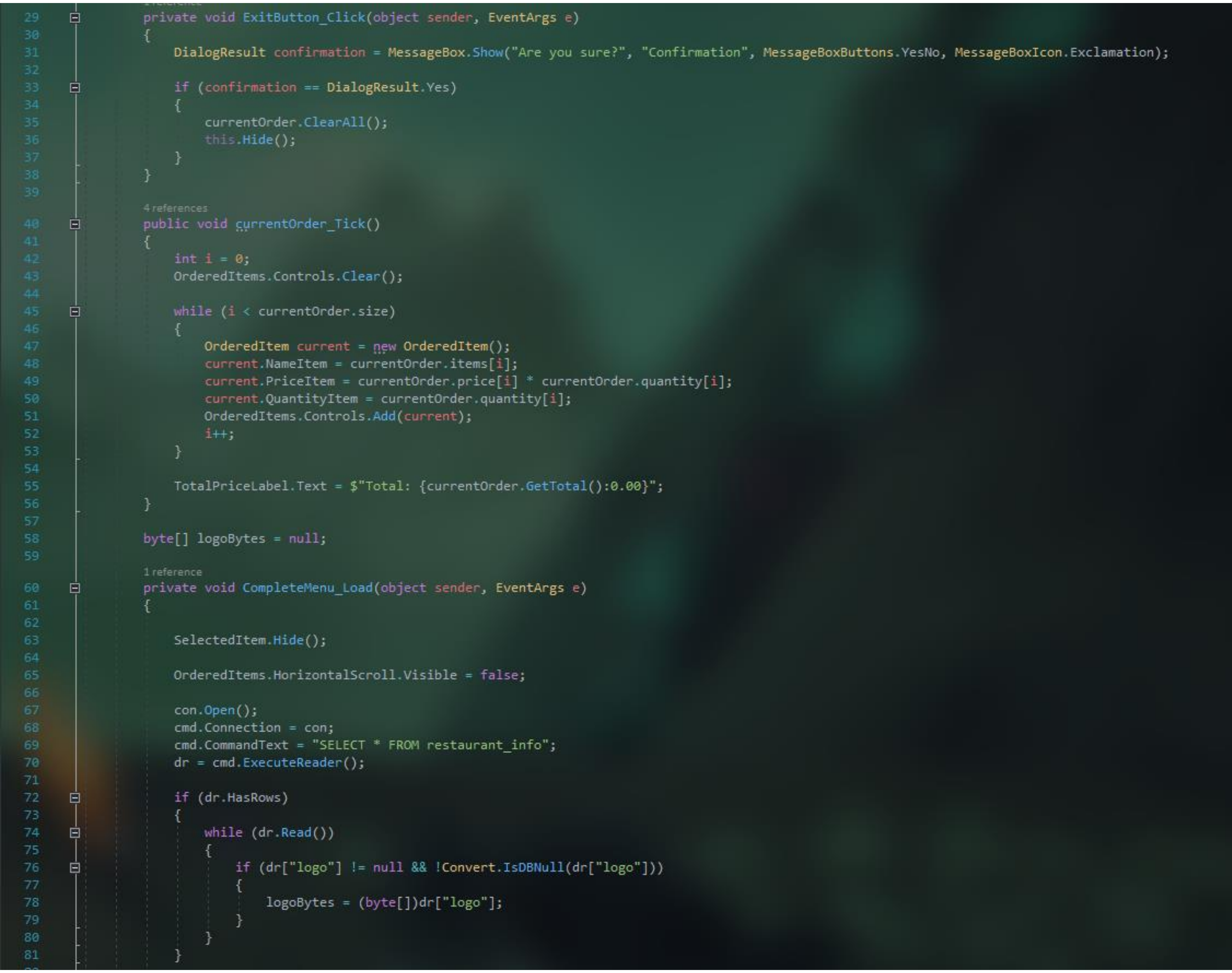
- Since database has old details, we have to check if the user inserted a new image; if not, then the details will be updated without the image.



```
            else if (ItemCategoryInput.SelectedIndex == 1)
            {
                cmd.Parameters.AddWithValue("icategory", "starter");
            }
            else if (ItemCategoryInput.SelectedIndex == 2)
            {
                cmd.Parameters.AddWithValue("icategory", "main");
            }
            else if (ItemCategoryInput.SelectedIndex == 3)
            {
                cmd.Parameters.AddWithValue("icategory", "dessert");
            }
            else
            {
                cmd.Parameters.AddWithValue("icategory", "extra");
            }
        }

        int rowsAffected = cmd.ExecuteNonQuery();
        if (rowsAffected == 0)
            MessageBox.Show("Error... item not updated.");
        else
        {
            MessageBox.Show("Item updated successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        con.Close();

        ItemDetailsPanel.Enabled = false;
        UploadItemPicture.Image = null;
        ClearFormFields_Click(this, new EventArgs());
    }

    MenuItemsList.Enabled = false;
    SpecialDealsRadio_CheckedChanged(this, new EventArgs());
}

8 references
private void ClearFormFields_Click(object sender, EventArgs e)
{
    UploadItemPicture.Image = null;
    ItemNameInput.Clear();
    ItemPriceInput.Value = 0;
    ItemCategoryInput.Text = "";
}

1 reference
private void DeleteItem_Click(object sender, EventArgs e)
{
    DialogResult res = MessageBox.Show("Are you sure you want to delete item?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    if (res == DialogResult.No)
        return;

    con.Open();
    cmd.Connection = con;

    try
```

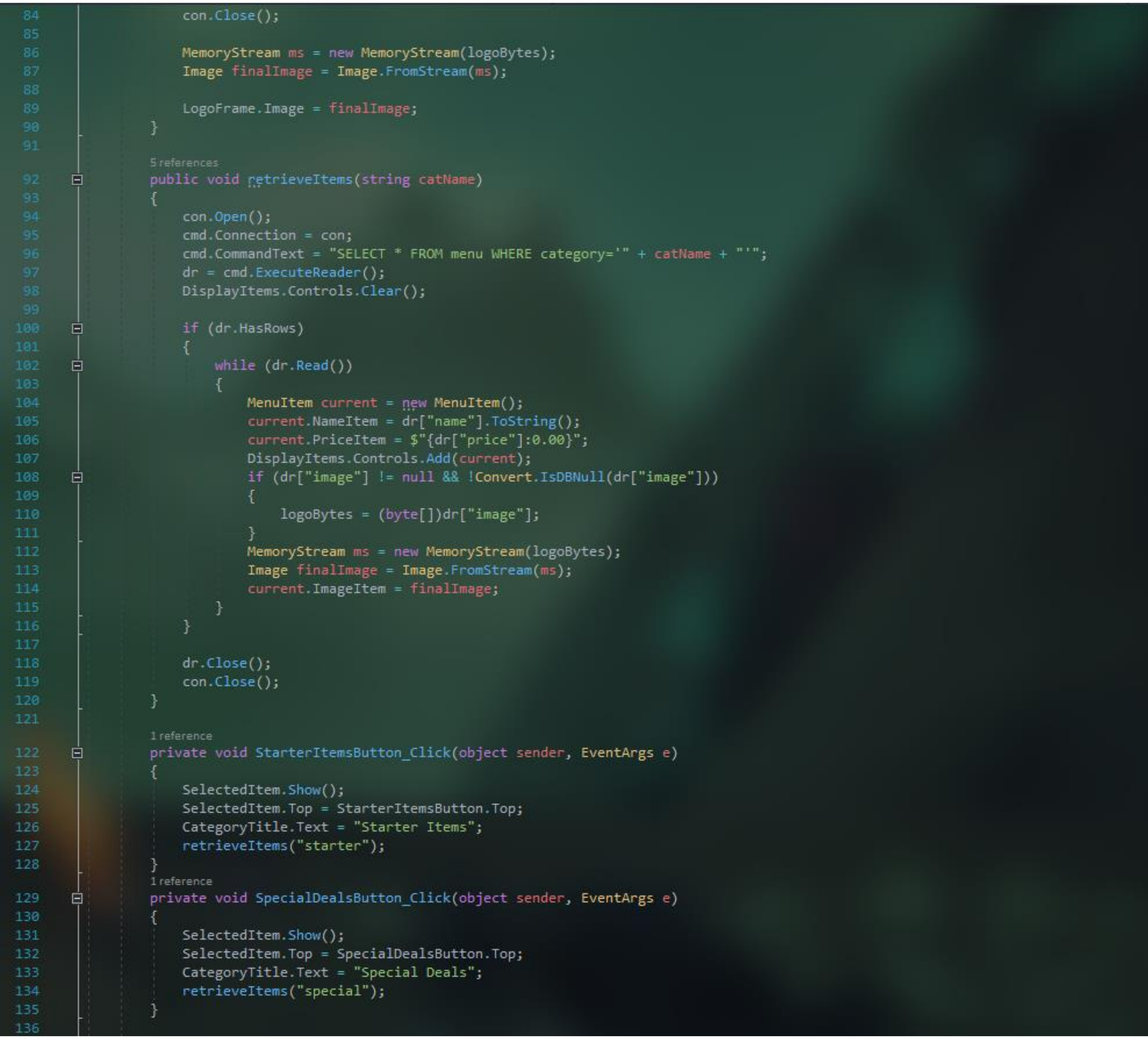
- After updating/inserting menu item record, we disable the form fields and update the menu items list.
- ClearFormFields_Click clears all the form fields for the user.
- DeleteItem_Click allows the user to delete the menu item from the database.

CompleteMenu.cs



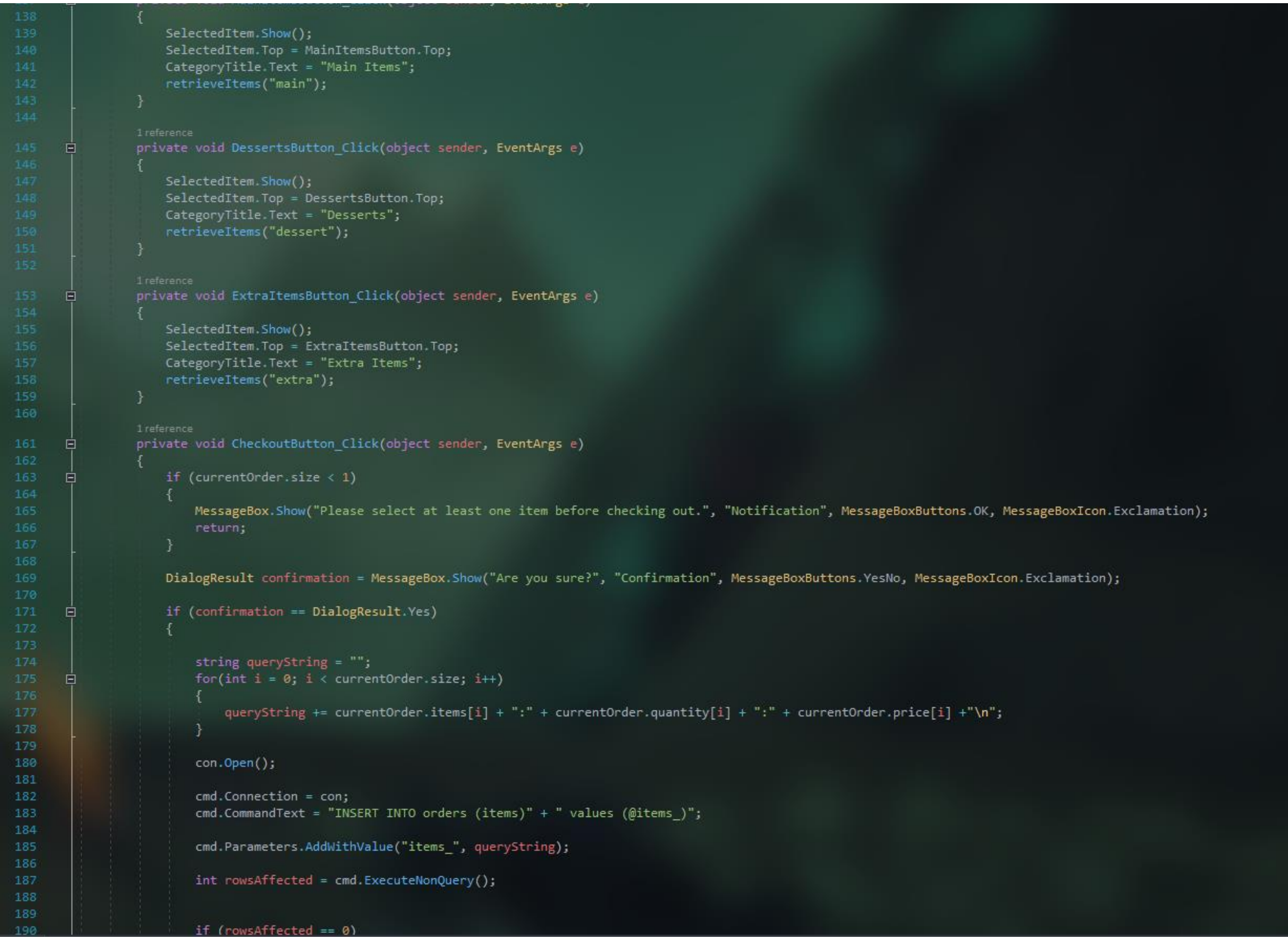
```
        private void ExitButton_Click(object sender, EventArgs e)
        {
            DialogResult confirmation = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

            if (confirmation == DialogResult.Yes)
            {
                currentOrder.ClearAll();
                this.Hide();
            }
        }

        4 references
        public void currentOrder_Tick()
        {
            int i = 0;
            OrderedItems.Controls.Clear();

            while (i < currentOrder.size)
            {
                OrderedItem current = new OrderedItem();
                current.NameItem = currentOrder.items[i];
                current.PriceItem = currentOrder.price[i] * currentOrder.quantity[i];
                current.QuantityItem = currentOrder.quantity[i];
                OrderedItems.Controls.Add(current);
                i++;
            }

            TotalPriceLabel.Text = $"Total: {currentOrder.GetTotal():0.00}";
        }

        byte[] logoBytes = null;

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

            SelectedItem.Hide();

            OrderedItems.HorizontalScroll.Visible = false;

            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM restaurant_info";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
                    {
                        logoBytes = (byte[])dr["logo"];
                    }
                }
            }
```

- ExitButton_Click checks if customer wants to cancel their order and return to splash screen.
- CurrentOrder_Tick gets the details of the current cart from the Order object created at the start, then displays it in an OrderedItem user control in the flow layout cart panel.
- CompleteMenu_Load gets the logo from database and display it in picture box.



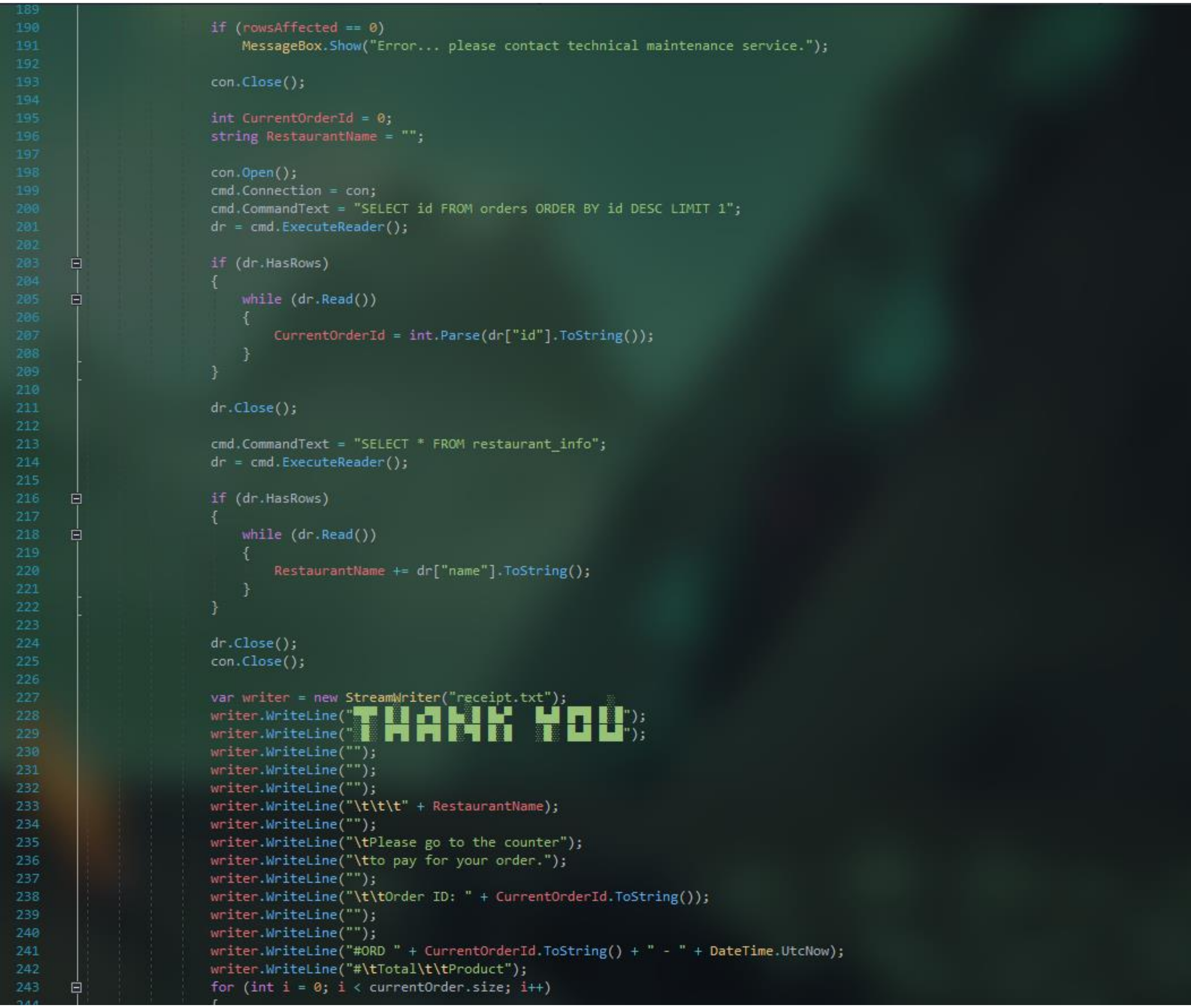
```
            con.Close();

            MemoryStream ms = new MemoryStream(logoBytes);
            Image finalImage = Image.FromStream(ms);

            LogoFrame.Image = finalImage;
        }

        5 references
        public void retrieveItems(string catName)
        {
            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM menu WHERE category='" + catName + "'";
            dr = cmd.ExecuteReader();
            DisplayItems.Controls.Clear();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    MenuItem current = new MenuItem();
                    current.NameItem = dr["name"].ToString();
                    current.PriceItem = $"{dr["price"]:0.00}";
                    DisplayItems.Controls.Add(current);
                    if (dr["image"] != null && !Convert.IsDBNull(dr["image"]))
                    {
                        logoBytes = (byte[])dr["image"];
                    }
                    MemoryStream ms = new MemoryStream(logoBytes);
                    Image finalImage = Image.FromStream(ms);
                    current.ImageItem = finalImage;
                }
            }

            dr.Close();
            con.Close();
        }

        1 reference
        private void StarterItemsButton_Click(object sender, EventArgs e)
        {
            SelectedItem.Show();
            SelectedItem.Top = StarterItemsButton.Top;
            CategoryTitle.Text = "Starter Items";
            retrieveItems("starter");
        }
        1 reference
        private void SpecialDealsButton_Click(object sender, EventArgs e)
        {
            SelectedItem.Show();
            SelectedItem.Top = SpecialDealsButton.Top;
            CategoryTitle.Text = "Special Deals";
            retrieveItems("special");
        }

```

- retrieveItems gets all the menu items depending on the selected category, and displays it in a MenuItem user control in a flow layout panel.
- Each button click event below activates the retrieveItems function that changes which menu items are displayed.



```
{
    SelectedItem.Show();
    SelectedItem.Top = MainItemsButton.Top;
    CategoryTitle.Text = "Main Items";
    retrieveItems("main");
}

1 reference
private void DessertsButton_Click(object sender, EventArgs e)
{
    SelectedItem.Show();
    SelectedItem.Top = DessertsButton.Top;
    CategoryTitle.Text = "Desserts";
    retrieveItems("dessert");
}

1 reference
private void ExtraItemsButton_Click(object sender, EventArgs e)
{
    SelectedItem.Show();
    SelectedItem.Top = ExtraItemsButton.Top;
    CategoryTitle.Text = "Extra Items";
    retrieveItems("extra");
}

1 reference
private void CheckoutButton_Click(object sender, EventArgs e)
{
    if (currentOrder.size < 1)
    {
        MessageBox.Show("Please select at least one item before checking out.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }

    DialogResult confirmation = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);

    if (confirmation == DialogResult.Yes)
    {

        string queryString = "";
        for(int i = 0; i < currentOrder.size; i++)
        {
            queryString += currentOrder.items[i] + ":" + currentOrder.quantity[i] + ":" + currentOrder.price[i] +"\n";
        }

        con.Open();

        cmd.Connection = con;
        cmd.CommandText = "INSERT INTO orders (items)" + " values (@items_)";

        cmd.Parameters.AddWithValue("items_", queryString);

        int rowsAffected = cmd.ExecuteNonQuery();


        if (rowsAffected == 0)
```

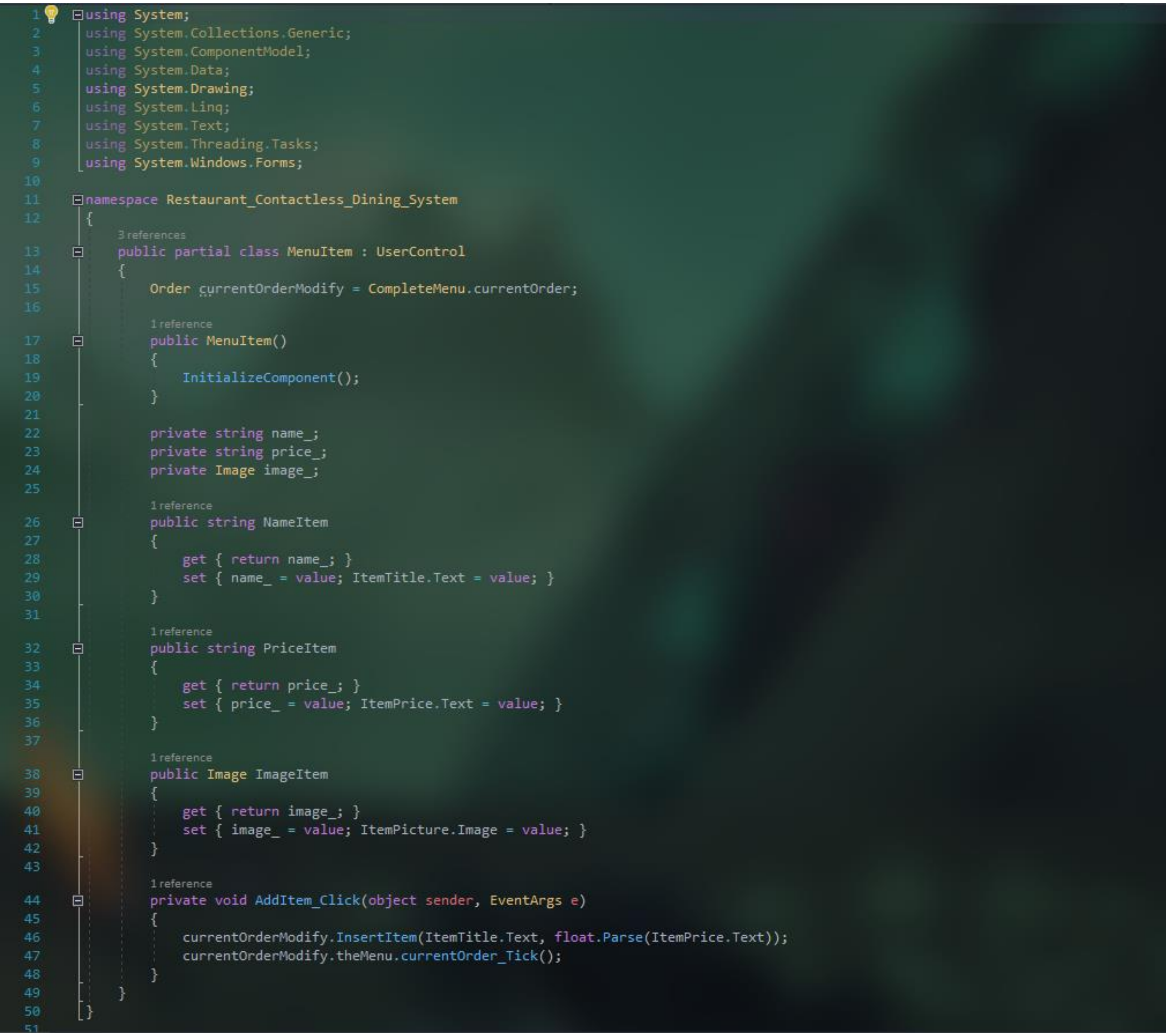
- Checkout button checks if cart has items, if not then show a message error.
- If all good, then it sends new order to database.



```
if (rowsAffected == 0)
    MessageBox.Show("Error... please contact technical maintenance service.");

con.Close();

int CurrentOrderId = 0;
string RestaurantName = "";

con.Open();
cmd.Connection = con;
cmd.CommandText = "SELECT id FROM orders ORDER BY id DESC LIMIT 1";
dr = cmd.ExecuteReader();

if (dr.HasRows)
{
    while (dr.Read())
    {
        CurrentOrderId = int.Parse(dr["id"].ToString());
    }
}

dr.Close();

cmd.CommandText = "SELECT * FROM restaurant_info";
dr = cmd.ExecuteReader();

if (dr.HasRows)
{
    while (dr.Read())
    {
        RestaurantName += dr["name"].ToString();
    }
}

dr.Close();
con.Close();

var writer = new StreamWriter("receipt.txt");
writer.WriteLine("THANK YOU");
writer.WriteLine("         ");
writer.WriteLine("");
writer.WriteLine("");
writer.WriteLine("");
writer.WriteLine("\t\t\t" + RestaurantName);
writer.WriteLine("");
writer.WriteLine("\tPlease go to the counter");
writer.WriteLine("\tto pay for your order.");
writer.WriteLine("");
writer.WriteLine("\t\tOrder ID: " + CurrentOrderId.ToString());
writer.WriteLine("");
writer.WriteLine("");
writer.WriteLine("#ORD " + CurrentOrderId.ToString() + " - " + DateTime.UtcNow);
writer.WriteLine("#\tTotal\t\tProduct");
for (int i = 0; i < currentOrder.size; i++)
{
```

- After inserting order in database, it creates a new text file in the form of a receipt that can be printed.

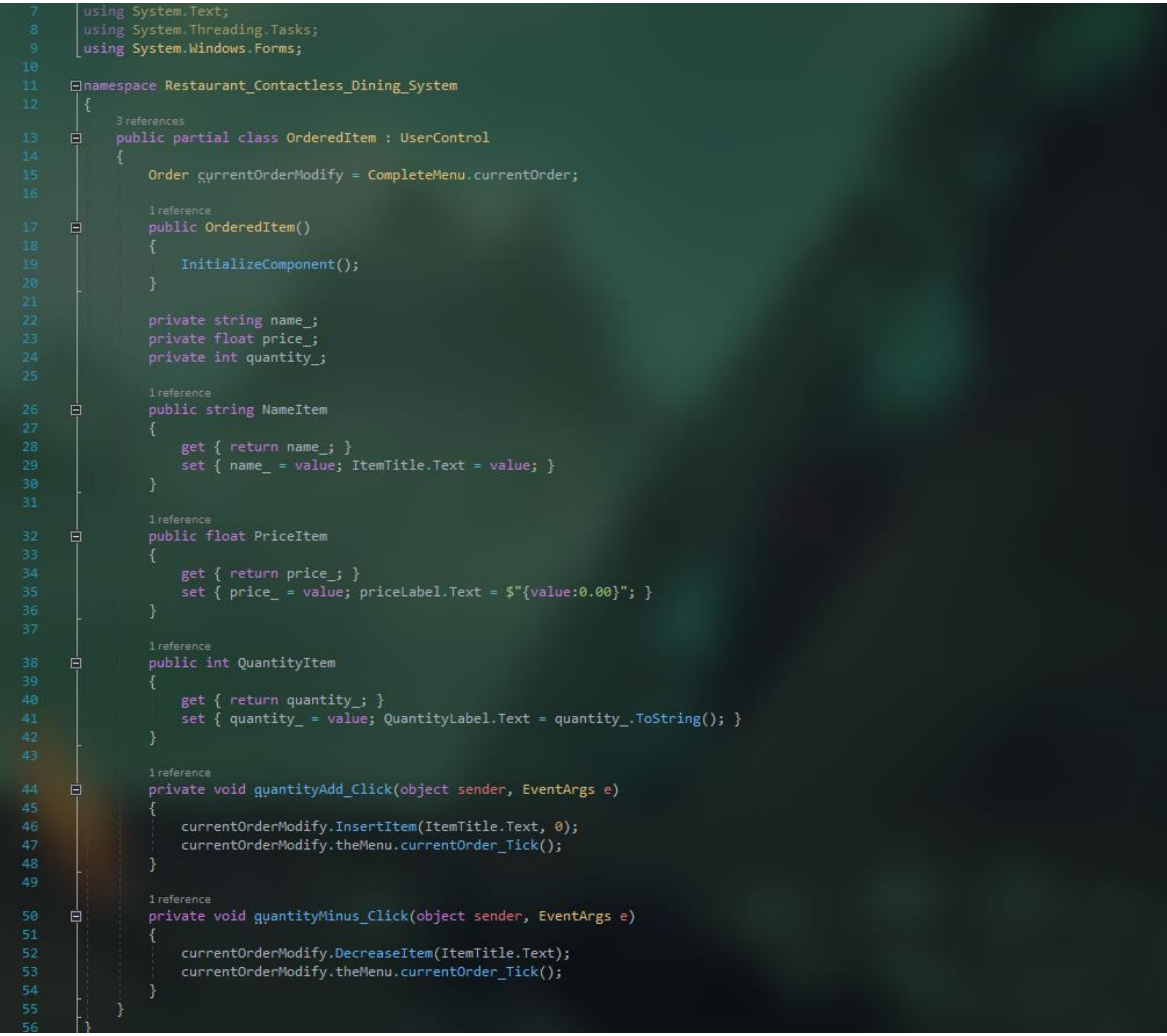
MenuItem.cs



```
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 Restaurant_Contactless_Dining_System
{
    3 references
    public partial class MenuItem : UserControl
    {
        Order currentOrderModify = CompleteMenu.currentOrder;

        1 reference
        public MenuItem()
        {
            InitializeComponent();
        }

        private string name_;
        private string price_;
        private Image image_;

        1 reference
        public string NameItem
        {
            get { return name_; }
            set { name_ = value; ItemTitle.Text = value; }
        }

        1 reference
        public string PriceItem
        {
            get { return price_; }
            set { price_ = value; ItemPrice.Text = value; }
        }

        1 reference
        public Image ImageItem
        {
            get { return image_; }
            set { image_ = value; ItemPicture.Image = value; }
        }

        1 reference
        private void AddItem_Click(object sender, EventArgs e)
        {
            currentOrderModify.InsertItem(ItemTitle.Text, float.Parse(ItemPrice.Text));
            currentOrderModify.theMenu.currentOrder_Tick();
        }
    }
}
```

- Setters and getters that is used to change the menu item details in the complete menu user control.
- AddItem_Click inserts that selected menu items into the cart.

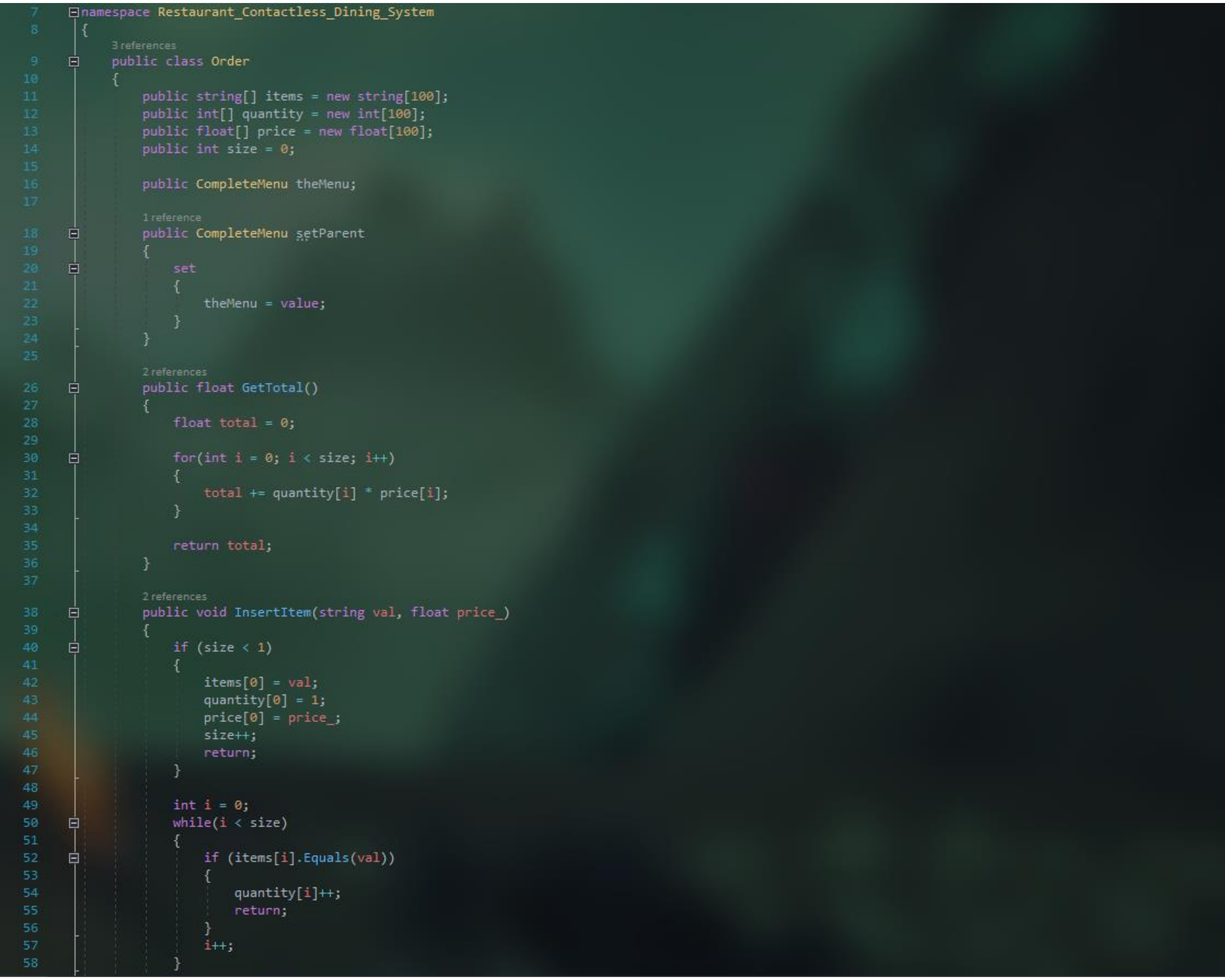
OrderedItem.cs



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

namespace Restaurant_Contactless_Dining_System
{
    3 references
    public partial class OrderedItem : UserControl
    {
        Order currentOrderModify = CompleteMenu.currentOrder;

        1 reference
        public OrderedItem()
        {
            InitializeComponent();
        }

        private string name_;
        private float price_;
        private int quantity_;

        1 reference
        public string NameItem
        {
            get { return name_; }
            set { name_ = value; ItemTitle.Text = value; }
        }

        1 reference
        public float PriceItem
        {
            get { return price_; }
            set { price_ = value; priceLabel.Text = $"{value:0.00}"; }
        }

        1 reference
        public int QuantityItem
        {
            get { return quantity_; }
            set { quantity_ = value; QuantityLabel.Text = quantity_.ToString(); }
        }

        1 reference
        private void quantityAdd_Click(object sender, EventArgs e)
        {
            currentOrderModify.InsertItem(ItemTitle.Text, 0);
            currentOrderModify.theMenu.currentOrder_Tick();
        }

        1 reference
        private void quantityMinus_Click(object sender, EventArgs e)
        {
            currentOrderModify.DecreaseItem(ItemTitle.Text);
            currentOrderModify.theMenu.currentOrder_Tick();
        }
    }
}
```

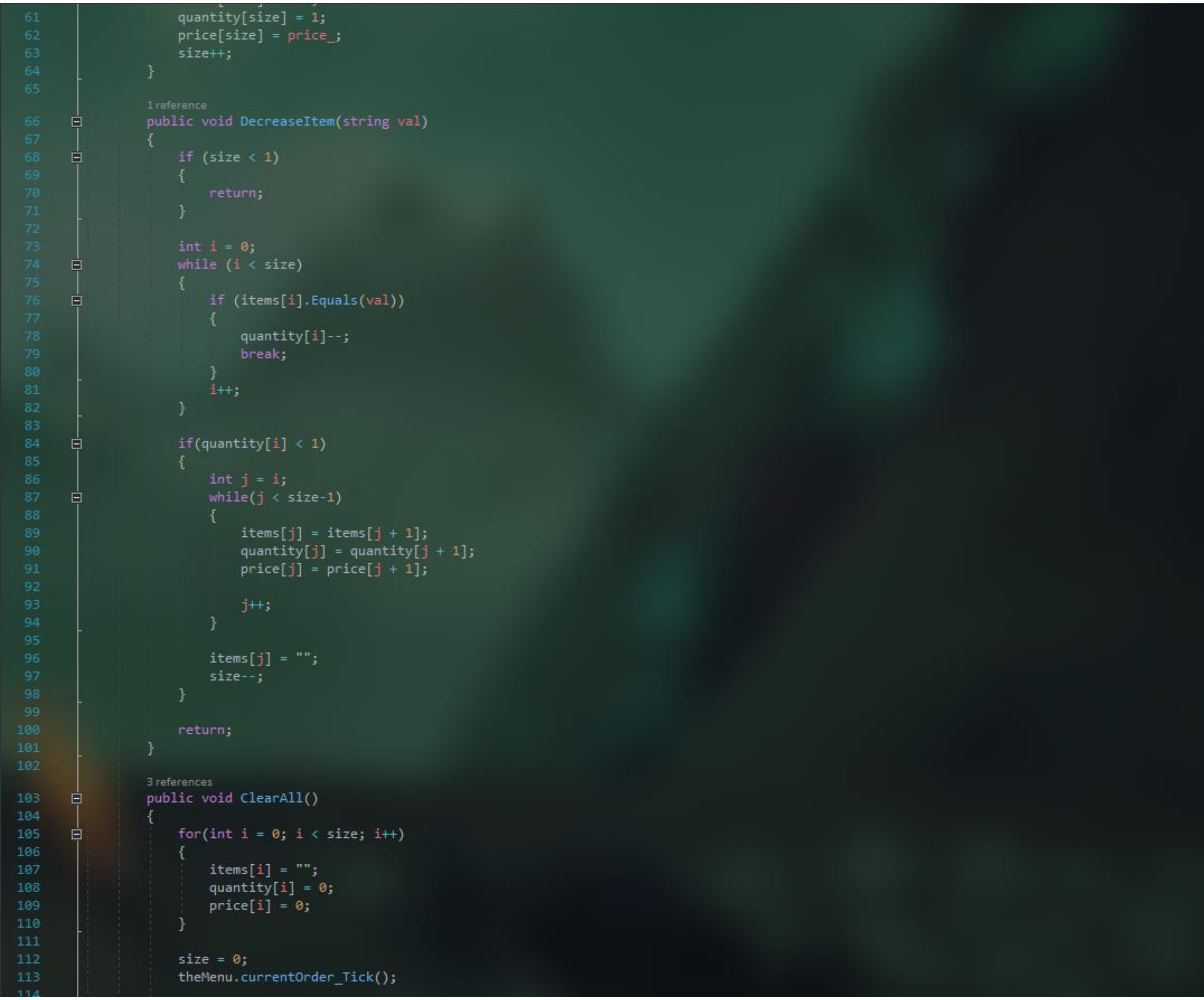
- Setters and getters that is used to change the cart item details in the checkout panel.
- quantityAdd_Click and quantityMinus_Click changes the specific cart item quantity.

Order.cs



```
namespace Restaurant_Contactless_Dining_System
{
    3 references
    public class Order
    {
        public string[] items = new string[100];
        public int[] quantity = new int[100];
        public float[] price = new float[100];
        public int size = 0;

        public CompleteMenu theMenu;

        1 reference
        public CompleteMenu setParent
        {
            set
            {
                theMenu = value;
            }
        }

        2 references
        public float GetTotal()
        {
            float total = 0;

            for(int i = 0; i < size; i++)
            {
                total += quantity[i] * price[i];
            }

            return total;
        }

        2 references
        public void InsertItem(string val, float price_)
        {
            if (size < 1)
            {
                items[0] = val;
                quantity[0] = 1;
                price[0] = price_;
                size++;
                return;
            }

            int i = 0;
            while(i < size)
            {
                if (items[i].Equals(val))
                {
                    quantity[i]++;
                    return;
                }
                i++;
            }
```

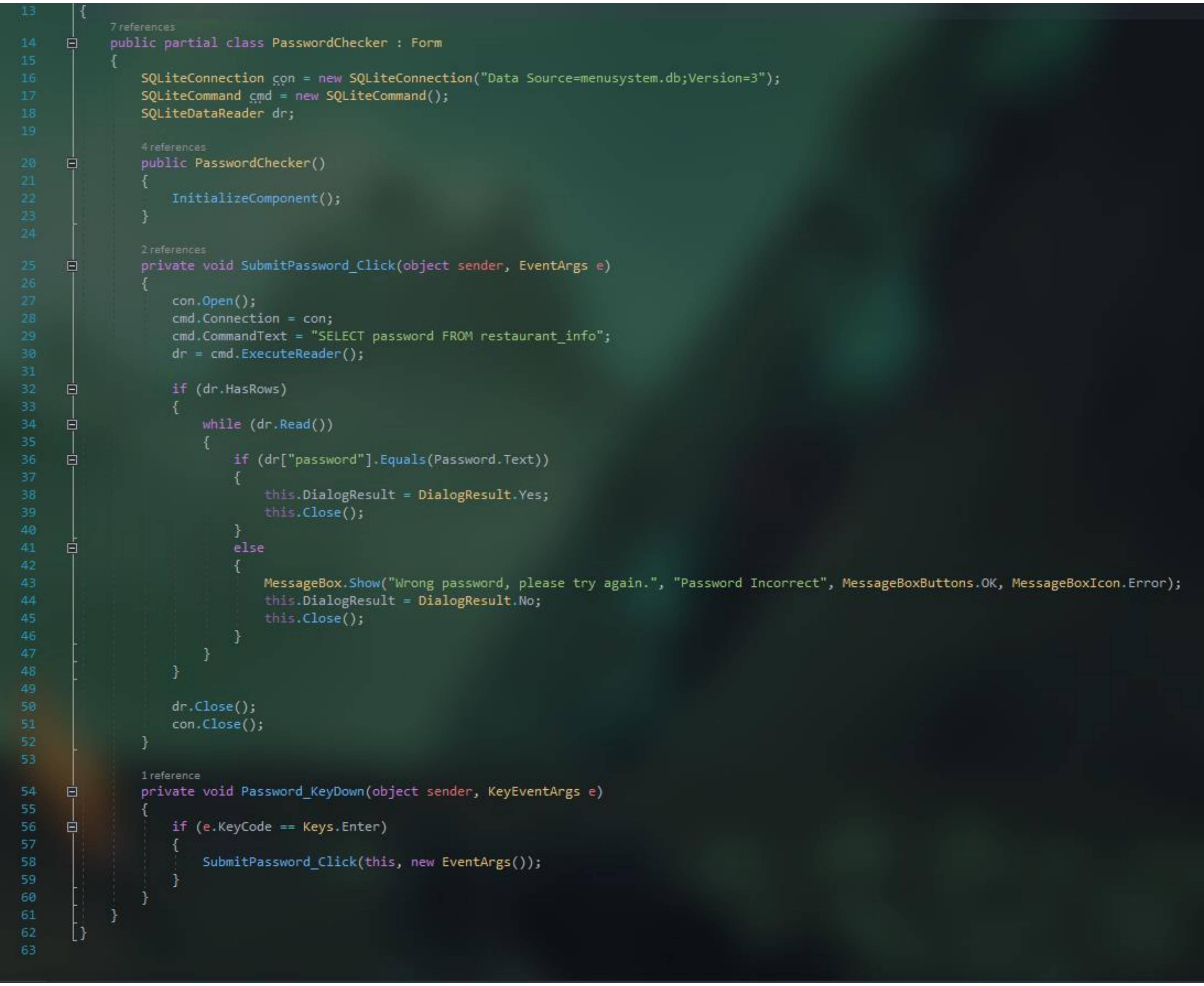
- Array that can contain up to 100 unique items in a single cart.
- GetTotal function gets the total of the cart and returns it.
- InsertItem checks if item exists, if not then adds it as a new item. If it exists then it changes quantity of item.



```
            quantity[size] = 1;
            price[size] = price_;
            size++;
        }

        1 reference
        public void DecreaseItem(string val)
        {
            if (size < 1)
            {
                return;
            }

            int i = 0;
            while (i < size)
            {
                if (items[i].Equals(val))
                {
                    quantity[i]--;
                    break;
                }
                i++;
            }

            if(quantity[i] < 1)
            {
                int j = i;
                while(j < size-1)
                {
                    items[j] = items[j + 1];
                    quantity[j] = quantity[j + 1];
                    price[j] = price[j + 1];

                    j++;
                }

                items[j] = "";
                size--;
            }

            return;
        }

        3 references
        public void ClearAll()
        {
            for(int i = 0; i < size; i++)
            {
                items[i] = "";
                quantity[i] = 0;
                price[i] = 0;
            }

            size = 0;
            theMenu.currentOrder_Tick();
```

- DecreaseItem function decreases the quantity of item, if the item reaches 0 quantity it is deleted from the array (cart).
- ClearAll function deletes all items from the cart.

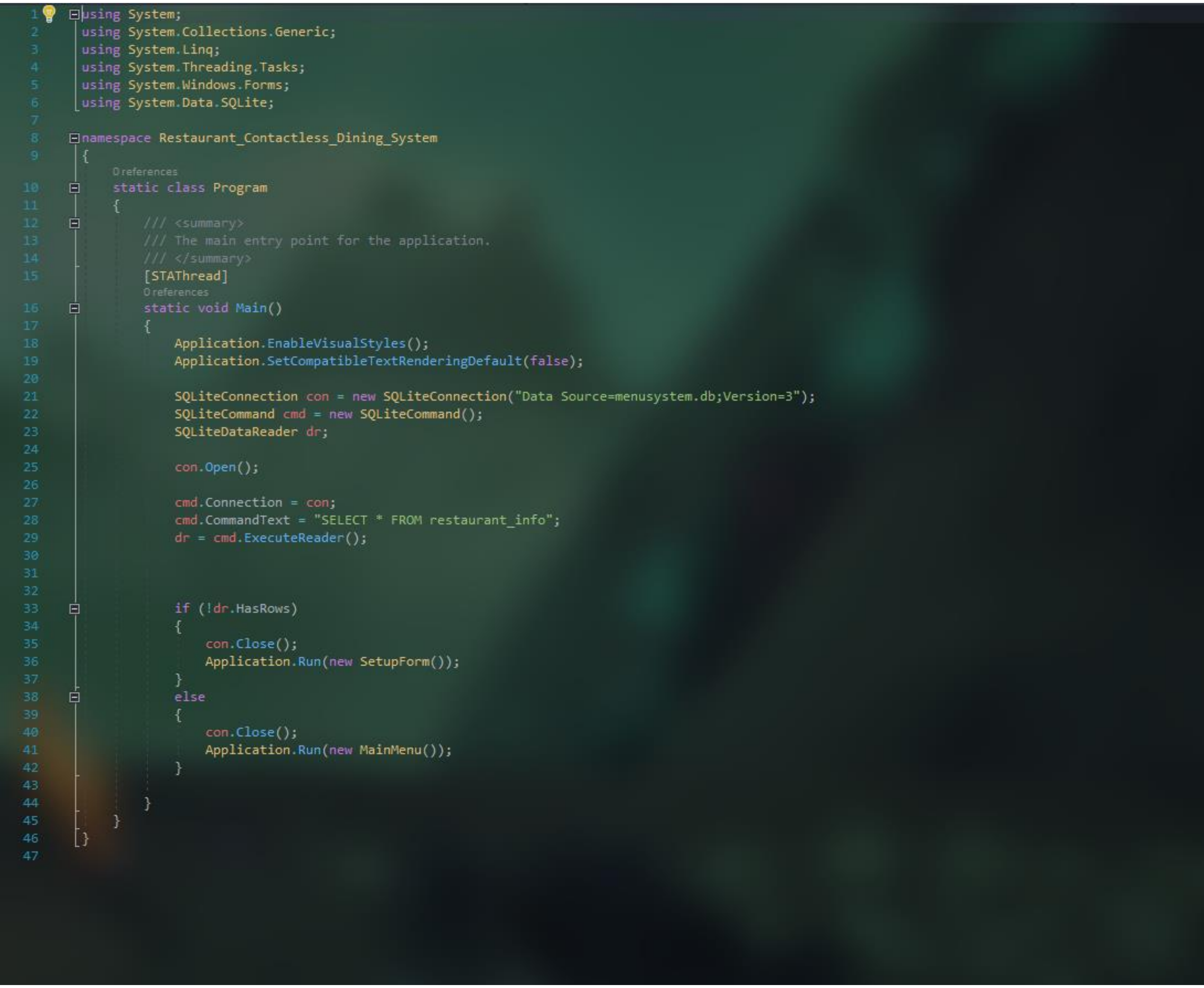
PasswordChecker.cs



```
{
    7 references
    public partial class PasswordChecker : Form
    {
        SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
        SQLiteCommand cmd = new SQLiteCommand();
        SQLiteDataReader dr;

        4 references
        public PasswordChecker()
        {
            InitializeComponent();
        }

        2 references
        private void SubmitPassword_Click(object sender, EventArgs e)
        {
            con.Open();
            cmd.Connection = con;
            cmd.CommandText = "SELECT password FROM restaurant_info";
            dr = cmd.ExecuteReader();

            if (dr.HasRows)
            {
                while (dr.Read())
                {
                    if (dr["password"].Equals(Password.Text))
                    {
                        this.DialogResult = DialogResult.Yes;
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("Wrong password, please try again.", "Password Incorrect", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        this.DialogResult = DialogResult.No;
                        this.Close();
                    }
                }
            }

            dr.Close();
            con.Close();
        }

        1 reference
        private void Password_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                SubmitPassword_Click(this, new EventArgs());
            }
        }
    }
}
```

- SubmitPassword_Click checks if password entered in the text field is the same as in the database.
- Password_KeyDown allows user to press enter to submit password instead of clicking button.

Program.cs



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SQLite;

namespace Restaurant_Contactless_Dining_System
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
            SQLiteCommand cmd = new SQLiteCommand();
            SQLiteDataReader dr;

            con.Open();

            cmd.Connection = con;
            cmd.CommandText = "SELECT * FROM restaurant_info";
            dr = cmd.ExecuteReader();



            if (!dr.HasRows)
            {
                con.Close();
                Application.Run(new SetupForm());
            }
            else
            {
                con.Close();
                Application.Run(new MainMenu());
            }

        }
    }
}
```

- At the start of the application, check if owner of restaurant submitted their details; if not then start the start up before the main menu.