

Contactless Dining System Report

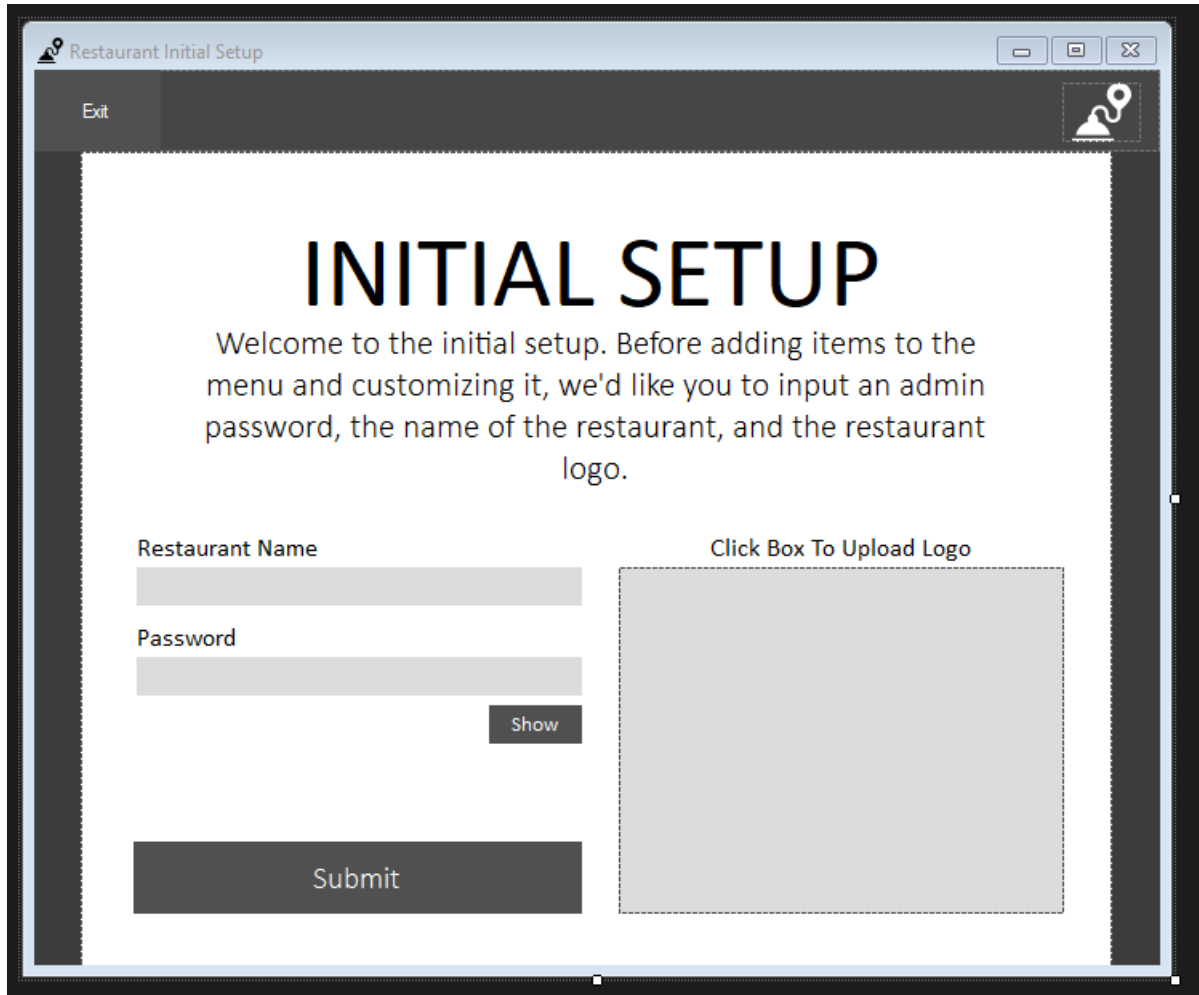
Zaid Issa 20190148

Contents

GUI Documentation	2
Initial Setup	2
Main Menu.....	3
Orders being prepared display.....	3
Customer Ordering Menu Splash Screen.....	4
Management Menu	4
Cashier Side.....	5
Kitchen Side.....	6
Manager Retrieve All Orders	7
Restaurant Info Manager.....	7
Manage Menu Items.....	8
Customer Menu	8
Extra Windows Forms	10
Code Implementation	12
Form1.cs.....	12
Form2.cs.....	13
Form3.cs.....	14
Form4.cs.....	15
Form5.cs.....	16
CompleteMenu.cs.....	33
MenuItem.cs	37
OrderedItem.cs	38
Order.cs.....	39
PasswordChecker.cs.....	41
Program.cs	42

GUI Documentation

Initial Setup



The screenshot shows a window titled "Restaurant Initial Setup" with standard OS window controls (minimize, maximize, close) in the top right. A dark header bar contains an "Exit" button on the left and a logo on the right. The main content area has a large "INITIAL SETUP" heading, followed by a welcome message: "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." Below this, there are three input fields: "Restaurant Name", "Password", and a large dashed box labeled "Click Box To Upload Logo". The "Password" field has a "Show" button next to it. At the bottom left is a large "Submit" button.

Restaurant Initial Setup

Exit

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.

Restaurant Name

Password

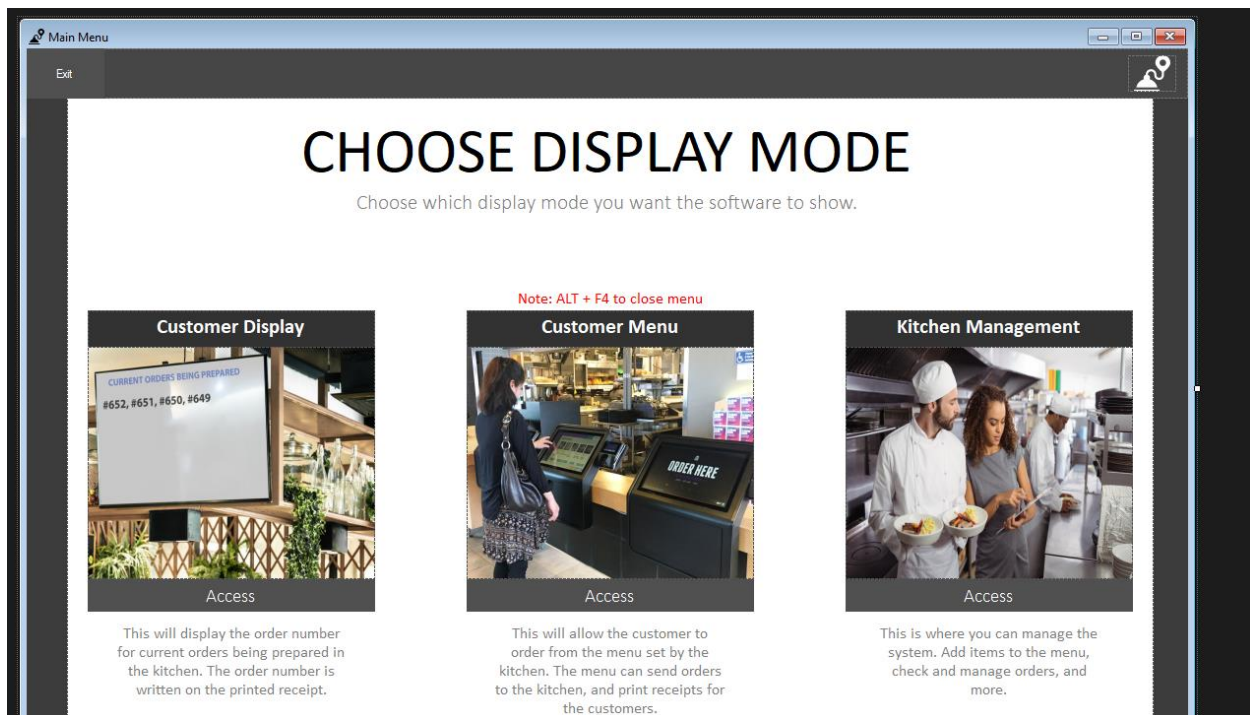
Show

Click Box To Upload Logo

Submit

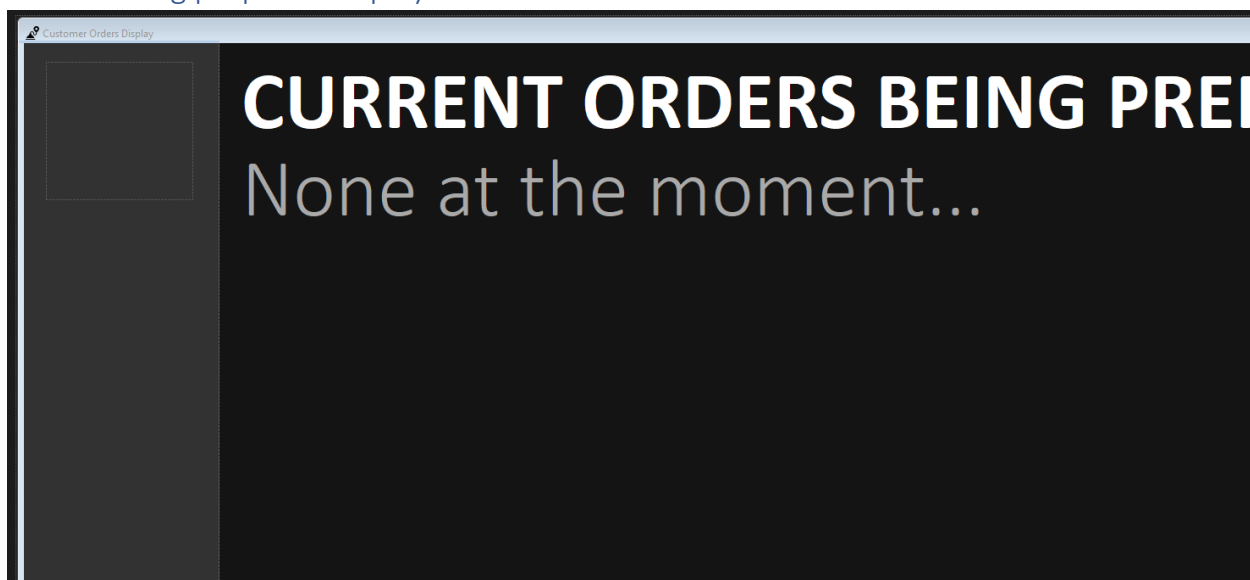
In the initial setup, the restaurant can add their restaurant logo, name, and manager password. After submitting the proper details, they will be taken to the main menu.

Main Menu



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



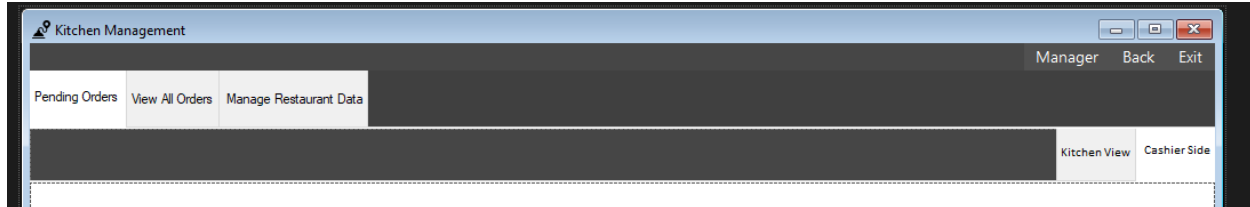
In the customer display, the logo of the restaurant is loaded on the left; while on the right, all the order IDs of orders being prepared will show up automatically, and refresh every 5 seconds.

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.

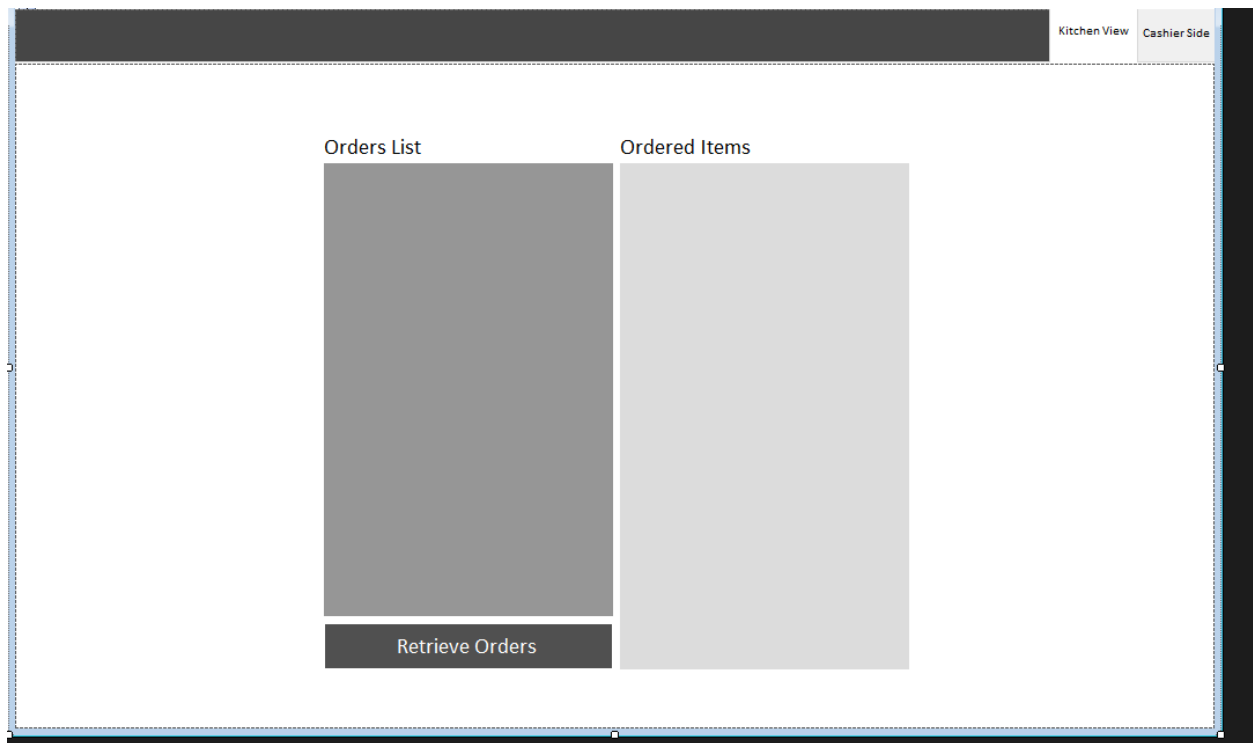
Cashier Side

The interface is divided into three main sections. On the left is the 'Orders List' section, which is currently empty and has a 'Retrieve Orders' button at the bottom. In the center is the 'Ordered Items' section, also empty. On the right is the 'Review' panel, which displays the 'Status:' and 'Total Price' (0.00 JD). Below the total price are three buttons: 'Confirm Order', 'Cancel Order', and 'Complete Order'.

Orders List	Ordered Items	Review
		Status: Total Price 0.00 JD Confirm Order Cancel Order Complete Order

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



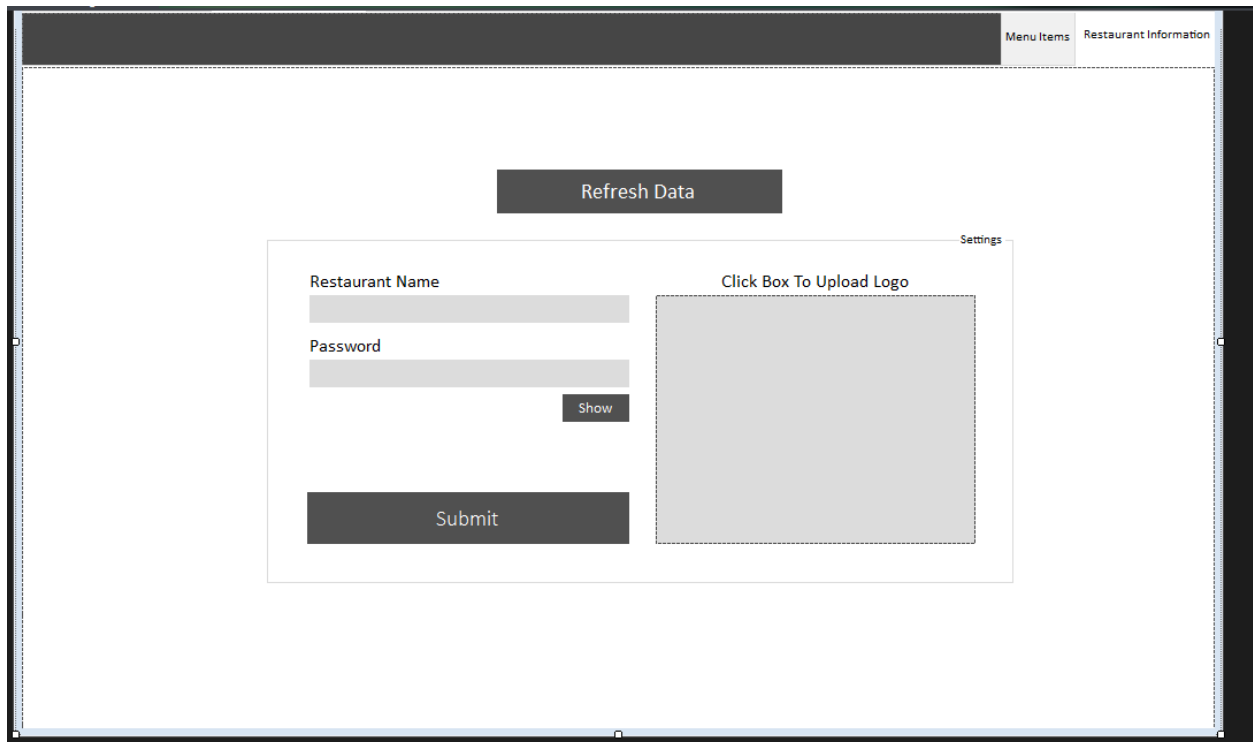
Kitchen tab shows the orders need to get prepared, while the ordered items show the item name, and the quantity.

Manager Retrieve All Orders



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

Restaurant Info Manager



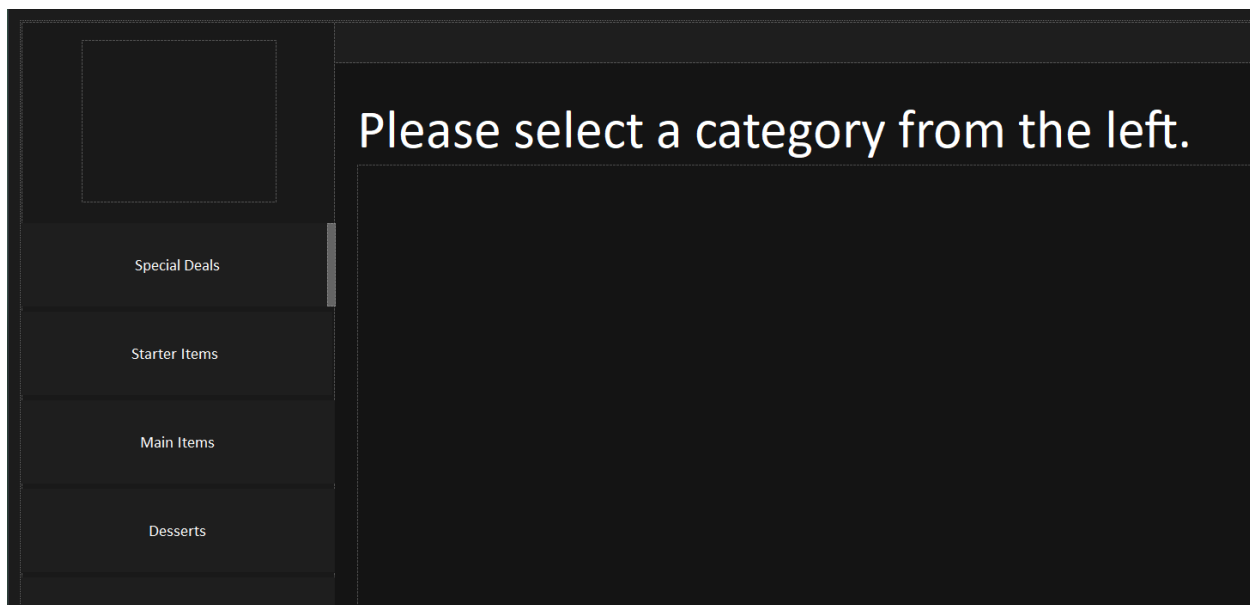
This tab allows the manager to change the restaurant name, password, or logo set before.

Manage Menu Items

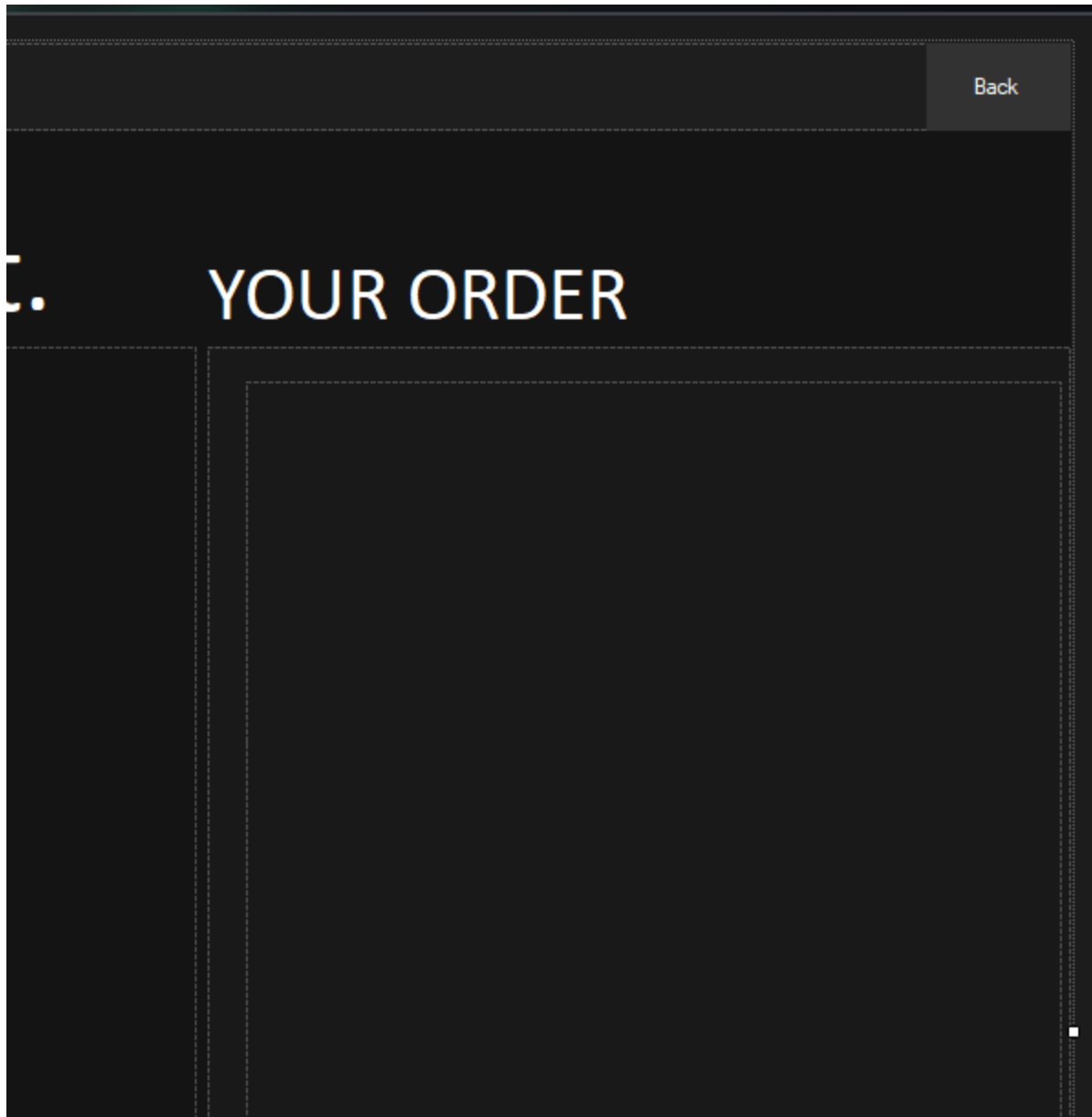
The screenshot shows a web application interface for managing menu items. At the top, there are two tabs: 'Menu Items' (active) and 'Restaurant Information'. Below the tabs, there are three buttons: 'Modify Items', 'Add New Item', and 'Clear Form Fields'. The main content area is divided into two sections. The left section, titled 'Choose which category items to view', contains five radio buttons: 'Special Deals', 'Starter Items', 'Main Items', 'Desserts', and 'Extra Items'. Below these is a large gray rectangular area representing the list of menu items, with a 'Delete' button at the bottom right. The right section, titled 'Click box to upload food item image', contains a large gray box for the image upload. Below this are three input fields: 'Item Name' (text), 'Item Price' (number, currently showing '0.00'), and 'Item Category' (dropdown menu). A 'Submit' button is located at the bottom right of the right section.

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

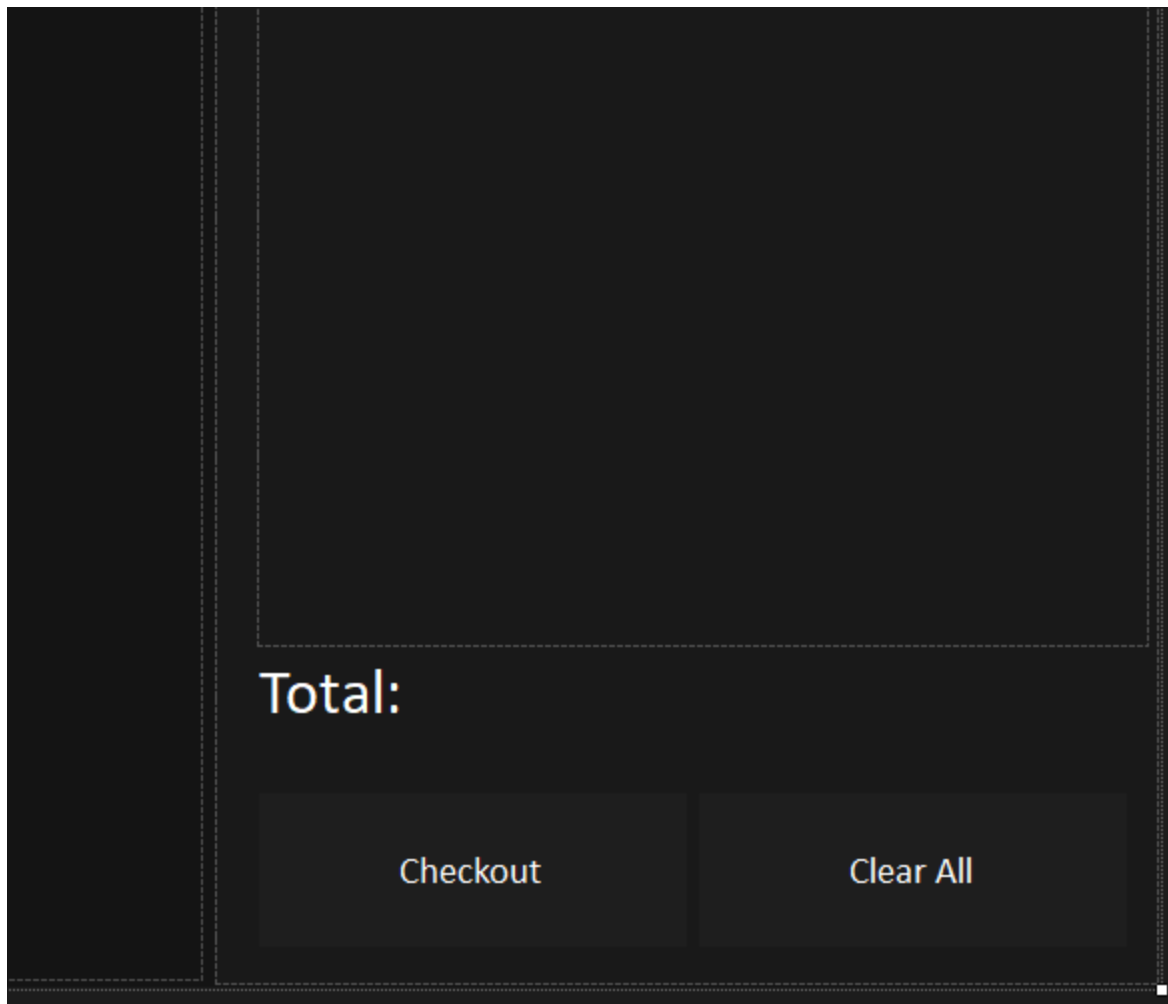
Customer Menu



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.

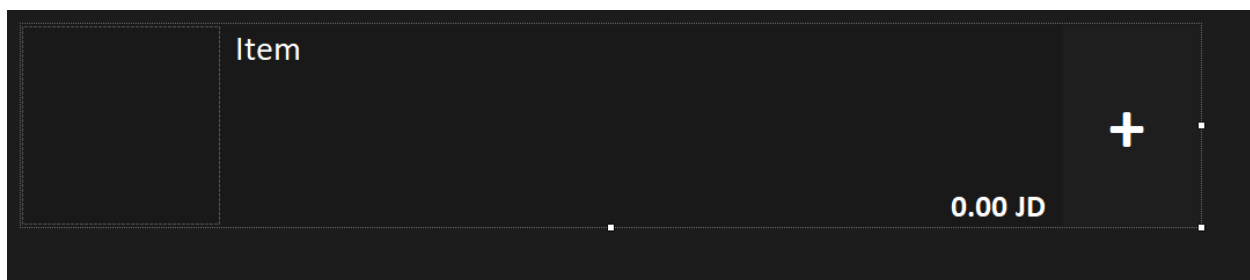


The back button allows the user to cancel their order, then the panel will go back to the splash screen. Under the “YOUR ORDER” label all the items in the cart would appear.



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.

A dark-themed user control for a shopping cart item. It features a dashed white border. Inside, the word "Item" is on the left. To its right are three dark gray buttons: a minus sign "-", the number "1", and a plus sign "+". Further right is the price "0.00". Small white square handles are visible at the corners and midpoints of the dashed border.

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

A custom dialog box titled "Admin Password Login" in a light blue header bar with a red close button (X) in the top right corner. The main area has a light gray background and contains the text "PLEASE ENTER ADMIN PASSWORD" in bold black letters. Below the text is a white rectangular input field. At the bottom center is a dark gray button labeled "Submit". Small white square handles are visible at the corners and midpoints of the dialog's border.

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

Code Implementation

Form1.cs

```
46 bool logoUploadOkay = false;
47 OpenFileDialog open = new OpenFileDialog();
48
49 1 reference
50 private void LogoUpload_Click(object sender, EventArgs e)
51 {
52     open.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
53     if (open.ShowDialog() == DialogResult.OK)
54     {
55         LogoUpload.Image = new Bitmap(open.FileName);
56         logoUploadOkay = true;
57     }
58 }
59
60 1 reference
61 private void ExitButton_Click(object sender, EventArgs e)
62 {
63     Application.Exit();
64 }
65
66 1 reference
67 private void SubmitPassword_Click(object sender, EventArgs e)
68 {
69     DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
70     if (res == DialogResult.No)
71         return;
72
73     if (RestaurantName.Text.Equals("") || Password.Text.Equals(""))
74     {
75         MessageBox.Show("Error: Restaurant Name or Password field is empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
76         return;
77     }
78     if (!logoUploadOkay)
79     {
80         MessageBox.Show("Error: Logo not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
81         return;
82     }
83
84     con.Open();
85     cmd.Connection = con;
86     cmd.CommandText = "INSERT INTO restaurant_info (name, logo, password)" + " values (@iname, @ilogo, @ipassword)";
87
88     byte[] image = File.ReadAllBytes(open.FileName);
89
90     cmd.Parameters.AddWithValue("iname", RestaurantName.Text);
91     cmd.Parameters.Add("ilogo", DbType.Binary, 20).Value = image;
92     cmd.Parameters.AddWithValue("ipassword", Password.Text);
93
94     int rowsAffected = cmd.ExecuteNonQuery();
95     if (rowsAffected == 0)
96         MessageBox.Show("Error... Record not added.");
97     else
98     {
99         Form changeForm = new MainMenu();
100         this.Hide();
101     }
102 }
```

- 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

```
17     InitializeComponent();
18 }
19
20 1 reference
21 private void ExitButton_Click(object sender, EventArgs e)
22 {
23     Application.Exit();
24 }
25
26 1 reference
27 private void Choice1Access_Click(object sender, EventArgs e)
28 {
29     Form changeForm = new CustomerOrdersDisplay();
30     this.Hide();
31     changeForm.ShowDialog();
32     this.Show();
33 }
34
35 1 reference
36 private void Choice2Access_Click(object sender, EventArgs e)
37 {
38     Form changeForm = new RestaurantMenu();
39     this.Hide();
40     changeForm.ShowDialog();
41     this.Show();
42 }
43
44 1 reference
45 private void Choice3Access_Click(object sender, EventArgs e)
46 {
47     PasswordChecker passCheck = new PasswordChecker();
48     passCheck.ShowDialog();
49
50     if (passCheck.DialogResult.Equals(DialogResult.Yes))
51     {
52         Form changeForm = new KitchenManagement();
53         this.Hide();
54         changeForm.ShowDialog();
55         try
56         {
57             this.Show();
58         }
59         catch(ObjectDisposedException)
60         {
61             Application.Exit();
62         }
63     }
64     else
65     {
66         return;
67     }
68 }
```

- 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

```

32 cmd.CommandText = "SELECT * FROM orders WHERE status='preparing'";
33
34 dr = cmd.ExecuteReader();
35
36 if (!dr.HasRows)
37     OrderNumbers.Text = "None at the moment...";
38 else
39 {
40     OrderNumbers.Text = "";
41     while (dr.Read())
42     {
43         OrderNumbers.Text += "#" + dr["id"].ToString() + " ";
44     }
45 }
46
47 dr.Close();
48 con.Close();
49 }
50
51 byte[] logoBytes = null;
52
53 1 reference
54 private void CustomerOrdersDisplay_Load(object sender, EventArgs e)
55 {
56     FormBorderStyle = FormBorderStyle.None;
57     WindowState = FormWindowState.Maximized;
58
59     con.Open();
60     cmd.Connection = con;
61     cmd.CommandText = "SELECT logo FROM restaurant_info";
62     dr = cmd.ExecuteReader();
63
64     if (dr.HasRows)
65     {
66         while (dr.Read())
67         {
68             if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
69             {
70                 logoBytes = (byte[])dr["logo"];
71             }
72         }
73     }
74
75     dr.Close();
76     con.Close();
77
78     MemoryStream ms = new MemoryStream(logoBytes);
79     Image finalImage = Image.FromStream(ms);
80
81     LogoFrame.Image = finalImage;
82
83     Timer ordersTimer = new Timer();
84     ordersTimer.Interval = 5000;
85     ordersTimer.Tick += new System.EventHandler(ordersTimer_Tick);
86     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

```

13 namespace Restaurant_Contactless_Dining_System
14 {
15     3references
16     public partial class RestaurantMenu : Form
17     {
18         SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
19         SQLiteCommand cmd = new SQLiteCommand();
20         SQLiteDataReader dr;
21
22         1reference
23         public RestaurantMenu()
24         {
25             InitializeComponent();
26
27             byte[] logoBytes = null;
28
29             1reference
30             private void RestaurantMenu_Load(object sender, EventArgs e)
31             {
32                 con.Open();
33                 cmd.Connection = con;
34                 cmd.CommandText = "SELECT * FROM restaurant_info";
35                 dr = cmd.ExecuteReader();
36
37                 if (dr.HasRows)
38                 {
39                     while (dr.Read())
40                     {
41                         Title.Text = "WELCOME TO " + (dr["name"].ToString()).ToUpper();
42                         if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
43                         {
44                             logoBytes = (byte[])dr["logo"];
45                         }
46                     }
47                 }
48                 dr.Close();
49                 con.Close();
50
51                 MemoryStream ms = new MemoryStream(logoBytes);
52                 Image finalImage = Image.FromStream(ms);
53                 LogoFrame.Image = finalImage;
54
55                 FormBorderStyle = FormBorderStyle.None;
56                 WindowState = FormWindowState.Maximized;
57             }
58
59             1reference
60             private void StartOrder_Click(object sender, EventArgs e)
61             {
62                 menuDisplay.Show();
63             }
64     }

```

- 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

```
29 private void KitchenManagement_Load(object sender, EventArgs e)
30 {
31     CompleteOrder.Enabled = false;
32     ConfirmOrder.Enabled = false;
33     CancelOrder.Enabled = false;
34
35     foreach (TabPage tab in Tabs.TabPages)
36     {
37         tab.Enabled = false;
38     }
39     (Tabs.TabPages[Tabs.SelectedIndex] as TabPage).Enabled = true;
40
41     foreach (TabPage tab in PendingOrdersTabs.TabPages)
42     {
43         tab.Enabled = false;
44     }
45     (PendingOrdersTabs.TabPages[PendingOrdersTabs.SelectedIndex] as TabPage).Enabled = true;
46
47     foreach (TabPage tab in ManageRestaurantTabs.TabPages)
48     {
49         tab.Enabled = false;
50     }
51     (ManageRestaurantTabs.TabPages[ManageRestaurantTabs.SelectedIndex] as TabPage).Enabled = true;
52 }
53
54 1 reference
55 private void exitToolStripMenuItem_Click(object sender, EventArgs e)
56 {
57     if (!isLock)
58         Application.Exit();
59     else
60     {
61         MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
62     }
63 }
64
65 1 reference
66 private void backToolStripMenuItem_Click(object sender, EventArgs e)
67 {
68     if (!isLock)
69         this.Close();
70     else
71     {
72         MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
73     }
74 }
75
76 1 reference
77 private void Tabs_SelectedIndexChanged(object sender, EventArgs e)
78 {
79     foreach (TabPage tab in Tabs.TabPages)
80     {
81         tab.Enabled = false;
82     }
83     (Tabs.TabPages[Tabs.SelectedIndex] as TabPage).Enabled = true;
84 }
```

- 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).


```

83 1reference
84 private void adminToolStripMenuItem_Click(object sender, EventArgs e)
85 {
86     if (!isAdmin)
87     {
88         lockScreenToolStripMenuItem.Enabled = false;
89
90         PasswordChecker passCheck = new PasswordChecker();
91         passCheck.ShowDialog();
92
93         if (passCheck.DialogResult.Equals(DialogResult.Yes))
94         {
95             isAdmin = true;
96             MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
97         }
98     }
99     else
100     {
101         lockScreenToolStripMenuItem.Enabled = true;
102     }
103 }
104
105 1reference
106 private void Tabs_Selecting(object sender, TabControlCancelEventArgs e)
107 {
108     if (!e.TabPage.Enabled && isLock)
109     {
110         MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
111         e.Cancel = true;
112     }
113 }
114
115 1reference
116 private void offToolStripMenuItem_Click(object sender, EventArgs e)
117 {
118     isLock = false;
119 }
120
121 1reference
122 private void onToolStripMenuItem_Click(object sender, EventArgs e)
123 {
124     isLock = true;
125 }
126
127 1reference
128 private void PendingOrdersTabs_Selecting(object sender, TabControlCancelEventArgs e)
129 {
130     if (!e.TabPage.Enabled && isLock)
131     {
132         MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
133         e.Cancel = true;
134     }
135 }
136
137 1reference
138 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

```
168 private void OrdersList_SelectedIndexChanged(object sender, EventArgs e)
169 {
170     ReviewGroupBox.Enabled = true;
171     OrderedItemsList.Items.Clear();
172
173     string lines = "";
174
175     con.Open();
176     cmd.Connection = con;
177     cmd.CommandText = "SELECT * FROM orders where id=" + OrdersList.SelectedItem;
178     try
179     {
180         dr = cmd.ExecuteReader();
181     }
182     catch (SQLiteException)
183     {
184         MessageBox.Show("Connection with database was cut, reconnecting...", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
185     }
186
187     try
188     {
189         var check = dr.HasRows;
190     }
191     catch (InvalidOperationException)
192     {
193         dr.Close();
194         con.Close();
195         return;
196     }
197
198     if (dr.HasRows)
199     {
200         double total = 0;
201
202         while (dr.Read())
203         {
204             lines += dr["items"];
205
206             foreach (string line in lines.Split('\n'))
207             {
208                 var found = line.IndexOf(":");
209                 var found2 = line.LastIndexOf(":");
210
211                 try
212                 {
213                     if (found > 0 && found2 - 1 > 0)
214                     {
215                         int i = 0;
216                         if (found2 - found > 2)
217                         {
218                             i = 2;
219                         }
220                         else
221                         {
222                             i = 1;
223                         }
224                     }
225                 }
226             }
227         }
228     }
229 }
```

- 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).

```

203 {
204     lines += dr["items"];
205
206     foreach (string line in lines.Split('\n'))
207     {
208         var found = line.IndexOf(":");
209         var found2 = line.LastIndexOf(":");
210
211         try
212         {
213             if(found > 0 && found2 - 1 > 0)
214             {
215                 int i = 0;
216                 if(found2 - found > 2)
217                 {
218                     i = 2;
219                 }
220                 else
221                 {
222                     i = 1;
223                 }
224                 OrderedItemsList.Items.Add($"{line.Substring(found + 1, i)}x {line.Substring(0, found)}");
225
226                 total += (float.Parse(line.Substring(found2 + 1)) * int.Parse(line.Substring(found + 1, i)));
227             }
228         }
229         catch(ArgumentOutOfRangeException)
230         {
231             MessageBox.Show("Error: no items found.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
232         }
233         finally { }
234     }
235
236     PriceNumber.Text = $"{total:0.00} JD";
237
238     if(dr["status"].Equals("preparing"))
239     {
240         StatusLabel.Text = "Status: Preparing";
241         ConfirmOrder.Enabled = false;
242         CancelOrder.Enabled = true;
243         CompleteOrder.Enabled = true;
244     }
245     else
246     {
247         StatusLabel.Text = "Status: Pending";
248         CompleteOrder.Enabled = false;
249         ConfirmOrder.Enabled = true;
250         CancelOrder.Enabled = true;
251     }
252 }
253
254 dr.Close();
255 con.Close();
256
257 }
258

```

- 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.

```

257 }
258
259 1reference
260 private void ConfirmOrder_Click(object sender, EventArgs e)
261 {
262     var res = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
263     if(res == DialogResult.No)
264     {
265         return;
266     }
267
268     con.Open();
269     cmd.Connection = con;
270     cmd.CommandText = "UPDATE orders SET status='preparing' where id=" + OrdersList.SelectedItem;
271
272     int rowsAffected = cmd.ExecuteNonQuery();
273     if (rowsAffected == 0)
274     {
275         MessageBox.Show("Error... Record not modified.");
276     }
277     else
278     {
279         MessageBox.Show("Order confirmed successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
280     }
281
282     con.Close();
283
284     RetrieveOrdersList_Click(this, new EventArgs());
285     ReviewGroupBox.Enabled = false;
286 }
287
288 1reference
289 private void CancelOrder_Click(object sender, EventArgs e)
290 {
291     var res = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
292     if (res == DialogResult.No)
293     {
294         return;
295     }
296
297     con.Open();
298     cmd.Connection = con;
299     cmd.CommandText = "UPDATE orders SET status='cancelled' where id=" + OrdersList.SelectedItem;
300
301     int rowsAffected = cmd.ExecuteNonQuery();
302     if (rowsAffected == 0)
303     {
304         MessageBox.Show("Error... Record not modified.");
305     }
306     else
307     {
308         MessageBox.Show("Order confirmed successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
309     }
310
311     con.Close();
312
313     RetrieveOrdersList_Click(this, new EventArgs());
314     ReviewGroupBox.Enabled = false;
315 }

```

- 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 database 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.


```

336
337 1reference
338 private void KitRetrieveOrders_Click(object sender, EventArgs e)
339 {
340     KitOrdersList.Items.Clear();
341
342     con.Open();
343     cmd.Connection = con;
344     cmd.CommandText = "SELECT id FROM orders where status='pending' OR status='preparing'";
345     dr = cmd.ExecuteReader();
346
347     if (dr.HasRows)
348     {
349         while (dr.Read())
350         {
351             KitOrdersList.Items.Add(dr["id"]);
352         }
353     }
354     dr.Close();
355     con.Close();
356 }
357
358 1reference
359 private void KitOrdersList_SelectedIndexChanged(object sender, EventArgs e)
360 {
361     KitOrderedItemsList.Items.Clear();
362
363     string lines = "";
364
365     con.Open();
366     cmd.Connection = con;
367     cmd.CommandText = "SELECT * FROM orders where id=" + KitOrdersList.SelectedItem;
368     dr = cmd.ExecuteReader();
369
370     if (dr.HasRows)
371     {
372         double total = 0;
373
374         while (dr.Read())
375         {
376             lines += dr["items"];
377
378             foreach (string line in lines.Split('\n'))
379             {
380                 var found = line.IndexOf(":");
381                 var found2 = line.LastIndexOf(":");
382
383                 try
384                 {
385                     if (found > 0 && found2 - 1 > 0)
386                     {
387                         int i = 0;
388                         if (found2 - found > 2)
389                         {
390                             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.

```

412
413 private void RetrieveAllOrders_Click(object sender, EventArgs e)
414 {
415     con.Open();
416
417     string comm = "SELECT * FROM orders ORDER BY id";
418     cmd = new SQLiteCommand(comm, con);
419
420     var da = new SQLiteDataAdapter(comm, con);
421     var ds = new DataSet();
422
423     da.Fill(ds, "orders");
424
425     OrdersGridView.DataSource = ds.Tables["orders"].DefaultView;
426
427     con.Close();
428 }
429
430 1 reference
431 private void ManageRestaurantTabs_Selecting(object sender, TabControlCancelEventArgs e)
432 {
433     if (!e.TabPage.Enabled && isLock)
434     {
435         MessageBox.Show("Please disable lock screen from manager settings.", "Permission Denied", MessageBoxButtons.OK, MessageBoxIcon.Error);
436         e.Cancel = true;
437     }
438 }
439
440 1 reference
441 private void ManageRestaurantTabs_SelectedIndexChanged(object sender, EventArgs e)
442 {
443     foreach (TabPage tab in ManageRestaurantTabs.TabPages)
444     {
445         tab.Enabled = false;
446     }
447     (ManageRestaurantTabs.TabPages[ManageRestaurantTabs.SelectedIndex] as TabPage).Enabled = true;
448 }
449
450 bool logoUploadOkay = false;
451 byte[] logoBytes = null;
452
453 1 reference
454 private void RefreshData_Click(object sender, EventArgs e)
455 {
456     if (!isAdmin)
457     {
458         PasswordChecker passCheck = new PasswordChecker();
459         passCheck.ShowDialog();
460
461         if (passCheck.DialogResult.Equals(DialogResult.Yes))
462         {
463             isAdmin = true;
464             MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
465         }
466         else
467         {
468

```

- 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.

```

454 {
455     PasswordChecker passCheck = new PasswordChecker();
456     passCheck.ShowDialog();
457
458     if (passCheck.DialogResult.Equals(DialogResult.Yes))
459     {
460         isAdmin = true;
461         MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
462     }
463     else
464     {
465         RestaurantInfoGroup.Enabled = false;
466         return;
467     }
468 }
469
470 RestaurantInfoGroup.Enabled = true;
471
472 con.Open();
473
474 cmd.Connection = con;
475 cmd.CommandText = "SELECT * FROM restaurant_info";
476
477 dr = cmd.ExecuteReader();
478
479 if (dr.HasRows)
480 {
481     while (dr.Read())
482     {
483         RestaurantName.Text = dr["name"].ToString();
484         Password.Text = dr["password"].ToString();
485         if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
486         {
487             logoBytes = (byte[])dr["logo"];
488         }
489     }
490 }
491
492 dr.Close();
493 con.Close();
494
495 MemoryStream ms = new MemoryStream(logoBytes);
496 Image finalImage = Image.FromStream(ms);
497
498 LogoUpload.Image = finalImage;
499 logoUploadOkay = true;
500 }
501
502 bool showPWD = false;
503 1 reference
504 private void ShowPassword_Click(object sender, EventArgs e)
505 {
506     if (showPWD)
507     {
508         Password.UseSystemPasswordChar = true;
509         ShowPassword.Text = "Show";
510     }
511     else
512     {
513         Password.UseSystemPasswordChar = false;
514         ShowPassword.Text = "Hide";
515     }
516     showPWD = !showPWD;
517 }

```

- 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.

```

507 Password.UseSystemPasswordChar = true;
508 ShowPassword.Text = "Show";
509 showPWD = false;
510 }
511 else
512 {
513 Password.UseSystemPasswordChar = false;
514 ShowPassword.Text = "Hide";
515 showPWD = true;
516 }
517 }
518
519 OpenFileDialog open = new OpenFileDialog();
520
521 private void LogoUpload_Click(object sender, EventArgs e)
522 {
523 open.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
524 if (open.ShowDialog() == DialogResult.OK)
525 {
526 LogoUpload.Image = new Bitmap(open.FileName);
527
528 logoUploadOkay = true;
529 }
530 else
531 {
532 logoUploadOkay = false;
533 }
534 }
535
536 private void SubmitPassword_Click(object sender, EventArgs e)
537 {
538 if (!isAdmin)
539 {
540 PasswordChecker passCheck = new PasswordChecker();
541 passCheck.ShowDialog();
542
543 if (passCheck.DialogResult.Equals(DialogResult.Yes))
544 {
545 isAdmin = true;
546 MessageBox.Show("Logged in successfully.", "User Accepted", MessageBoxButtons.OK, MessageBoxIcon.Information);
547 }
548 else
549 {
550 RestaurantInfoGroup.Enabled = false;
551 return;
552 }
553 }
554
555 DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
556 if (res == DialogResult.No)
557 return;
558
559 if (RestaurantName.Text.Equals("") || Password.Text.Equals(""))
560 {

```

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


```

563     }
564     if (!logoUploadOkay)
565     {
566         MessageBox.Show("Error: Logo not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
567         return;
568     }
569
570     con.Open();
571     cmd.Connection = con;
572
573     if(open.FileName != "")
574     {
575         cmd.CommandText = "UPDATE restaurant_info SET name=@iname, password=@ipassword, logo=@ilogo";
576
577         byte[] image = File.ReadAllBytes(open.FileName);
578
579         cmd.Parameters.AddWithValue("iname", RestaurantName.Text);
580         cmd.Parameters.Add("ilogo", DbType.Binary, 20).Value = image;
581         cmd.Parameters.AddWithValue("ipassword", Password.Text);
582     }
583     else
584     {
585         cmd.CommandText = "UPDATE restaurant_info SET name=@iname, password=@ipassword";
586
587         cmd.Parameters.AddWithValue("iname", RestaurantName.Text);
588         cmd.Parameters.AddWithValue("ipassword", Password.Text);
589     }
590
591     int rowsAffected = cmd.ExecuteNonQuery();
592     if (rowsAffected == 0)
593         MessageBox.Show("Error... Record not updated.");
594     else
595     {
596         MessageBox.Show("Restaurant Information Updated Successfully.", "Confirmation", MessageBoxButtons.OK, MessageBoxIcon.Information);
597     }
598
599     con.Close();
600 }
601
602 bool newItem = false;
603
604 1reference
605 private void ModifyItems_Click(object sender, EventArgs e)
606 {
607     newItem = false;
608     ModifyItemsPanel.Enabled = true;
609     ItemDetailsPanel.Enabled = false;
610 }
611
612 1reference
613 private void AddNewItem_Click(object sender, EventArgs e)
614 {
615     newItem = true;
616     ModifyItemsPanel.Enabled = false;
617     ItemDetailsPanel.Enabled = true;
618 }

```

- 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
- AddNewItem_Click enables new item form fields to submit.

3 references

```
618 private void SpecialDealsRadio_CheckedChanged(object sender, EventArgs e)
619 {
620     MenuItemList.Enabled = true;
621     DeleteItem.Enabled = false;
622     ClearFormFields_Click(this, new EventArgs());
623     ItemDetailsPanel.Enabled = false;
624     MenuItemList.Items.Clear();
625
626     con.Open();
627     cmd.Connection = con;
628
629     cmd.CommandText = "SELECT * FROM menu WHERE category='special'";
630     dr = cmd.ExecuteReader();
631
632     if (dr.HasRows)
633     {
634         while (dr.Read())
635         {
636             MenuItemList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
637         }
638     }
639
640     dr.Close();
641     con.Close();
642 }
643
```

1 reference

```
644 private void StarterItemsRadio_CheckedChanged(object sender, EventArgs e)
645 {
646     MenuItemList.Enabled = true;
647     DeleteItem.Enabled = false;
648     ClearFormFields_Click(this, new EventArgs());
649     ItemDetailsPanel.Enabled = false;
650     MenuItemList.Items.Clear();
651
652     con.Open();
653     cmd.Connection = con;
654
655     cmd.CommandText = "SELECT * FROM menu WHERE category='starter'";
656     dr = cmd.ExecuteReader();
657
658     if (dr.HasRows)
659     {
660         while (dr.Read())
661         {
662             MenuItemList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
663         }
664     }
665
666     dr.Close();
667     con.Close();
668 }
669
```

1 reference

```
670 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.

```

670 private void MainItemsRadio_CheckedChanged(object sender, EventArgs e)
671 {
672     MenuItemList.Enabled = true;
673     DeleteItem.Enabled = false;
674     ClearFormFields_Click(this, new EventArgs());
675     ItemDetailsPanel.Enabled = false;
676     MenuItemList.Items.Clear();
677
678     con.Open();
679     cmd.Connection = con;
680
681     cmd.CommandText = "SELECT * FROM menu WHERE category='main'";
682     dr = cmd.ExecuteReader();
683
684     if (dr.HasRows)
685     {
686         while (dr.Read())
687         {
688             MenuItemList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
689         }
690     }
691
692     dr.Close();
693     con.Close();
694 }
695
696 1 reference
697 private void DessertsRadio_CheckedChanged(object sender, EventArgs e)
698 {
699     MenuItemList.Enabled = true;
700     DeleteItem.Enabled = false;
701     ClearFormFields_Click(this, new EventArgs());
702     ItemDetailsPanel.Enabled = false;
703     MenuItemList.Items.Clear();
704
705     con.Open();
706     cmd.Connection = con;
707
708     cmd.CommandText = "SELECT * FROM menu WHERE category='dessert'";
709     dr = cmd.ExecuteReader();
710
711     if (dr.HasRows)
712     {
713         while (dr.Read())
714         {
715             MenuItemList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
716         }
717     }
718
719     dr.Close();
720     con.Close();
721 }
722
723 1 reference
724 private void ExtraItemsRadio_CheckedChanged(object sender, EventArgs e)

```

- Continuation of the above.

```

722 private void ExtraItemsRadio_CheckedChanged(object sender, EventArgs e)
723 {
724     MenuItemList.Enabled = true;
725     DeleteItem.Enabled = false;
726     ClearFormFields_Click(this, new EventArgs());
727     ItemDetailsPanel.Enabled = false;
728     MenuItemList.Items.Clear();
729
730     con.Open();
731     cmd.Connection = con;
732
733     cmd.CommandText = "SELECT * FROM menu WHERE category='extra'";
734     dr = cmd.ExecuteReader();
735
736     if (dr.HasRows)
737     {
738         while (dr.Read())
739         {
740             MenuItemList.Items.Add(dr["id"] + ". " + dr["name"].ToString());
741         }
742     }
743
744     dr.Close();
745     con.Close();
746 }
747
748 bool picUploadOkay = false;
749
750 1 reference
751 private void MenuItemList_SelectedIndexChanged(object sender, EventArgs e)
752 {
753     DeleteItem.Enabled = true;
754
755     con.Open();
756     cmd.Connection = con;
757
758     string itemID = (MenuItemList.SelectedItem.ToString()).Substring(0, MenuItemList.SelectedItem.ToString().IndexOf('.'));
759
760     cmd.CommandText = "SELECT * FROM menu WHERE id=" + int.Parse(itemID);
761     dr = cmd.ExecuteReader();
762
763     byte[] imageBytes = null;
764
765     if (dr.HasRows)
766     {
767         ItemDetailsPanel.Enabled = true;
768         while (dr.Read())
769         {
770             ItemNameInput.Text = dr["name"].ToString();
771             ItemPriceInput.Value = decimal.Parse(dr["price"].ToString());
772             if (dr["category"].Equals("special"))
773             {
774                 ItemCategoryInput.SelectedIndex = 0;
775             }
776             else if (dr["category"].Equals("starter"))
777             {

```

- 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.


```

781         ItemCategoryInput.SelectedIndex = 2;
782     }
783     else if (dr["category"].Equals("dessert"))
784     {
785         ItemCategoryInput.SelectedIndex = 3;
786     }
787     else
788     {
789         ItemCategoryInput.SelectedIndex = 4;
790     }
791 }
792 if (dr["image"] != null && !Convert.IsDBNull(dr["image"]))
793 {
794     imageBytes = (byte[])dr["image"];
795 }
796 }
797 }
798
799 dr.Close();
800 con.Close();
801
802 MemoryStream ms = new MemoryStream(imageBytes);
803 Image finalImage = Image.FromStream(ms);
804
805 UploadItemPicture.Image = finalImage;
806 picUploadOkay = true;
807 }
808
809 OpenFileDialog open2 = new OpenFileDialog();
810 1 reference
811 private void UploadItemPicture_Click(object sender, EventArgs e)
812 {
813     open2.Filter = "Image Files(*.png; *.jpg; *.jpeg; *.gif)|*.png; *.jpg; *.jpeg; *.gif";
814     if (open2.ShowDialog() == DialogResult.OK)
815     {
816         UploadItemPicture.Image = new Bitmap(open2.FileName);
817         picUploadOkay = true;
818     }
819 }
820
821 1 reference
822 private void SubmitItemDetails_Click(object sender, EventArgs e)
823 {
824     if(newItem)
825     {
826         DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
827         if (res == DialogResult.No)
828             return;
829
830         if (ItemNameInput.Text.Equals("") || ItemCategoryInput.SelectedItem == null)
831         {
832             MessageBox.Show("Error: Some details might be missing, please check before submitting.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
833             return;
834         }
835         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.

```

836     MessageBox.Show("Error: Image not uploaded.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
837     return;
838 }
839
840 con.Open();
841
842 cmd.Connection = con;
843 cmd.CommandText = "INSERT INTO menu (name, price, category, image)" + " values (@iname, @iprice, @icategory, @iimage)";
844
845 byte[] image = File.ReadAllBytes(open2.FileName);
846 cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
847 cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
848 if (ItemCategoryInput.SelectedIndex == 0)
849 {
850     cmd.Parameters.AddWithValue("icategory", "special");
851 }
852 else if (ItemCategoryInput.SelectedIndex == 1)
853 {
854     cmd.Parameters.AddWithValue("icategory", "starter");
855 }
856 else if (ItemCategoryInput.SelectedIndex == 2)
857 {
858     cmd.Parameters.AddWithValue("icategory", "main");
859 }
860 else if (ItemCategoryInput.SelectedIndex == 3)
861 {
862     cmd.Parameters.AddWithValue("icategory", "dessert");
863 }
864 else
865 {
866     cmd.Parameters.AddWithValue("icategory", "extra");
867 }
868 cmd.Parameters.Add("iimage", DbType.Binary, 20).Value = image;
869
870 int rowsAffected = cmd.ExecuteNonQuery();
871 if (rowsAffected == 0)
872     MessageBox.Show("Error... item not added.");
873 else
874 {
875     MessageBox.Show("Item added successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
876 }
877
878 con.Close();
879
880 ItemDetailsPanel.Enabled = false;
881 UploadItemPicture.Image = null;
882 ClearFormFields_Click(this, new EventArgs());
883 }
884 else
885 {
886     DialogResult res = MessageBox.Show("Are you sure you want to submit details?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
887     if (res == DialogResult.No)
888         return;
889
890     if (ItemNameInput.Text.Equals("") || ItemCategoryInput.SelectedItem == null)
891     {

```

- 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.

```

891 {
892     MessageBox.Show("Error: Some details might be missing, please check before submitting.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
893     return;
894 }
895 if (!picUploadOkay)
896 {
897     MessageBox.Show("Error: Image not uploaded", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
898     return;
899 }
900
901 con.Open();
902
903 cmd.Connection = con;
904 cmd.CommandText = "INSERT INTO menu (name, price, category, image)" + " values (@iname, @iprice, @icategory, @iimage)";
905
906 if (open2.FileName != "")
907 {
908     cmd.CommandText = "UPDATE menu SET name=@iname, price=@iprice, category=@icategory, image=@iimage WHERE id=@iid";
909
910     byte[] image = File.ReadAllBytes(open2.FileName);
911     cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
912     cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
913     cmd.Parameters.AddWithValue("iid", (MenuItemList.SelectedItem.ToString()).Substring(0, MenuItemList.SelectedItem.ToString().IndexOf('.')));
914     if (ItemCategoryInput.SelectedIndex == 0)
915     {
916         cmd.Parameters.AddWithValue("icategory", "special");
917     }
918     else if (ItemCategoryInput.SelectedIndex == 1)
919     {
920         cmd.Parameters.AddWithValue("icategory", "starter");
921     }
922     else if (ItemCategoryInput.SelectedIndex == 2)
923     {
924         cmd.Parameters.AddWithValue("icategory", "main");
925     }
926     else if (ItemCategoryInput.SelectedIndex == 3)
927     {
928         cmd.Parameters.AddWithValue("icategory", "dessert");
929     }
930     else
931     {
932         cmd.Parameters.AddWithValue("icategory", "extra");
933     }
934     cmd.Parameters.Add("iimage", DbType.Binary, 20).Value = image;
935 }
936 else
937 {
938     cmd.CommandText = "UPDATE menu SET name=@iname, price=@iprice, category=@icategory WHERE id=@iid";
939
940     cmd.Parameters.AddWithValue("iname", ItemNameInput.Text);
941     cmd.Parameters.AddWithValue("iprice", ItemPriceInput.Value);
942     cmd.Parameters.AddWithValue("iid", (MenuItemList.SelectedItem.ToString()).Substring(0, MenuItemList.SelectedItem.ToString().IndexOf('.')));
943     if (ItemCategoryInput.SelectedIndex == 0)
944     {
945         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.

```

947         case 1: (ItemCategoryInput.SelectedIndex == 1)
948         {
949             cmd.Parameters.AddWithValue("icategory", "starter");
950         }
951         else if (ItemCategoryInput.SelectedIndex == 2)
952         {
953             cmd.Parameters.AddWithValue("icategory", "main");
954         }
955         else if (ItemCategoryInput.SelectedIndex == 3)
956         {
957             cmd.Parameters.AddWithValue("icategory", "dessert");
958         }
959         else
960         {
961             cmd.Parameters.AddWithValue("icategory", "extra");
962         }
963     }
964
965     int rowsAffected = cmd.ExecuteNonQuery();
966     if (rowsAffected == 0)
967         MessageBox.Show("Error... item not updated.");
968     else
969     {
970         MessageBox.Show("Item updated successfully.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Information);
971     }
972
973     con.Close();
974
975     ItemDetailsPanel.Enabled = false;
976     UploadItemPicture.Image = null;
977     ClearFormFields_Click(this, new EventArgs());
978 }
979
980 MenuItemList.Enabled = false;
981 SpecialDealsRadio_CheckedChanged(this, new EventArgs());
982 }
983
984 8 references
985 private void ClearFormFields_Click(object sender, EventArgs e)
986 {
987     UploadItemPicture.Image = null;
988     ItemNameInput.Clear();
989     ItemPriceInput.Value = 0;
990     ItemCategoryInput.Text = "";
991 }
992
993 1 reference
994 private void DeleteItem_Click(object sender, EventArgs e)
995 {
996     DialogResult res = MessageBox.Show("Are you sure you want to delete item?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
997     if (res == DialogResult.No)
998         return;
999
1000     con.Open();
1001     cmd.Connection = con;

```

- 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

```
29 private void ExitButton_Click(object sender, EventArgs e)
30 {
31     DialogResult confirmation = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);
32
33     if (confirmation == DialogResult.Yes)
34     {
35         currentOrder.ClearAll();
36         this.Hide();
37     }
38 }
39
40 4 references
41 public void currentOrder_Tick()
42 {
43     int i = 0;
44     OrderedItems.Controls.Clear();
45
46     while (i < currentOrder.size)
47     {
48         OrderedItem current = new OrderedItem();
49         current.NameItem = currentOrder.items[i];
50         current.PriceItem = currentOrder.price[i] * currentOrder.quantity[i];
51         current.QuantityItem = currentOrder.quantity[i];
52         OrderedItems.Controls.Add(current);
53         i++;
54     }
55
56     TotalPriceLabel.Text = $"Total: {currentOrder.GetTotal():0.00}";
57 }
58
59 byte[] logoBytes = null;
60
61 1 reference
62 private void CompleteMenu_Load(object sender, EventArgs e)
63 {
64     SelectedItem.Hide();
65
66     OrderedItems.HorizontalScroll.Visible = false;
67
68     con.Open();
69     cmd.Connection = con;
70     cmd.CommandText = "SELECT * FROM restaurant_info";
71     dr = cmd.ExecuteReader();
72
73     if (dr.HasRows)
74     {
75         while (dr.Read())
76         {
77             if (dr["logo"] != null && !Convert.IsDBNull(dr["logo"]))
78             {
79                 logoBytes = (byte[])dr["logo"];
80             }
81         }
82     }
```

- 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.

```

84         con.Close();
85
86         MemoryStream ms = new MemoryStream(logoBytes);
87         Image finalImage = Image.FromStream(ms);
88
89         LogoFrame.Image = finalImage;
90     }
91
92     5 references
93     public void retrieveItems(string catName)
94     {
95         con.Open();
96         cmd.Connection = con;
97         cmd.CommandText = "SELECT * FROM menu WHERE category='" + catName + "'";
98         dr = cmd.ExecuteReader();
99         DisplayItems.Controls.Clear();
100
101         if (dr.HasRows)
102         {
103             while (dr.Read())
104             {
105                 MenuItem current = new MenuItem();
106                 current.NameItem = dr["name"].ToString();
107                 current.PriceItem = $"{dr["price"]:0.00}";
108                 DisplayItems.Controls.Add(current);
109                 if (dr["image"] != null && !Convert.IsDBNull(dr["image"]))
110                 {
111                     logoBytes = (byte[])dr["image"];
112                     MemoryStream ms = new MemoryStream(logoBytes);
113                     Image finalImage = Image.FromStream(ms);
114                     current.ImageItem = finalImage;
115                 }
116             }
117
118             dr.Close();
119             con.Close();
120         }
121
122     1 reference
123     private void StarterItemsButton_Click(object sender, EventArgs e)
124     {
125         SelectedItem.Show();
126         SelectedItem.Top = StarterItemsButton.Top;
127         CategoryTitle.Text = "Starter Items";
128         retrieveItems("starter");
129
130     1 reference
131     private void SpecialDealsButton_Click(object sender, EventArgs e)
132     {
133         SelectedItem.Show();
134         SelectedItem.Top = SpecialDealsButton.Top;
135         CategoryTitle.Text = "Special Deals";
136         retrieveItems("special");
137     }

```

- 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.

```

138 {
139     SelectedItem.Show();
140     SelectedItem.Top = MainItemsButton.Top;
141     CategoryTitle.Text = "Main Items";
142     retrieveItems("main");
143 }
144
145 1 reference
146 private void DessertsButton_Click(object sender, EventArgs e)
147 {
148     SelectedItem.Show();
149     SelectedItem.Top = DessertsButton.Top;
150     CategoryTitle.Text = "Desserts";
151     retrieveItems("dessert");
152 }
153
154 1 reference
155 private void ExtraItemsButton_Click(object sender, EventArgs e)
156 {
157     SelectedItem.Show();
158     SelectedItem.Top = ExtraItemsButton.Top;
159     CategoryTitle.Text = "Extra Items";
160     retrieveItems("extra");
161 }
162
163 1 reference
164 private void CheckoutButton_Click(object sender, EventArgs e)
165 {
166     if (currentOrder.size < 1)
167     {
168         MessageBox.Show("Please select at least one item before checking out.", "Notification", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
169         return;
170     }
171
172     DialogResult confirmation = MessageBox.Show("Are you sure?", "Confirmation", MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);
173
174     if (confirmation == DialogResult.Yes)
175     {
176         string queryString = "";
177         for(int i = 0; i < currentOrder.size; i++)
178         {
179             queryString += currentOrder.items[i] + ":" + currentOrder.quantity[i] + ":" + currentOrder.price[i] + "\n";
180         }
181
182         con.Open();
183
184         cmd.Connection = con;
185         cmd.CommandText = "INSERT INTO orders (items)" + " values (@items_)";
186
187         cmd.Parameters.AddWithValue("items_", queryString);
188
189         int rowsAffected = cmd.ExecuteNonQuery();
190
191         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.

```

189
190     if (rowsAffected == 0)
191         MessageBox.Show("Error... please contact technical maintenance service.");
192
193     con.Close();
194
195     int CurrentOrderId = 0;
196     string RestaurantName = "";
197
198     con.Open();
199     cmd.Connection = con;
200     cmd.CommandText = "SELECT id FROM orders ORDER BY id DESC LIMIT 1";
201     dr = cmd.ExecuteReader();
202
203     if (dr.HasRows)
204     {
205         while (dr.Read())
206         {
207             CurrentOrderId = int.Parse(dr["id"].ToString());
208         }
209     }
210
211     dr.Close();
212
213     cmd.CommandText = "SELECT * FROM restaurant_info";
214     dr = cmd.ExecuteReader();
215
216     if (dr.HasRows)
217     {
218         while (dr.Read())
219         {
220             RestaurantName += dr["name"].ToString();
221         }
222     }
223
224     dr.Close();
225     con.Close();
226
227     var writer = new StreamWriter("receipt.txt");
228     writer.WriteLine("THANK YOU");
229     writer.WriteLine("THANK YOU");
230     writer.WriteLine("");
231     writer.WriteLine("");
232     writer.WriteLine("");
233     writer.WriteLine("\t\t" + RestaurantName);
234     writer.WriteLine("");
235     writer.WriteLine("\tPlease go to the counter");
236     writer.WriteLine("\tto pay for your order.");
237     writer.WriteLine("");
238     writer.WriteLine("\t\tOrder ID: " + CurrentOrderId.ToString());
239     writer.WriteLine("");
240     writer.WriteLine("");
241     writer.WriteLine("#ORD " + CurrentOrderId.ToString() + " - " + DateTime.UtcNow);
242     writer.WriteLine("#\tTotal\t\tProduct");
243     for (int i = 0; i < currentOrder.size; i++)
244     {

```

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

MenuItem.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace Restaurant_Contactless_Dining_System
12 {
13     3 references
14     public partial class MenuItem : UserControl
15     {
16         Order currentOrderModify = CompleteMenu.currentOrder;
17
18         1 reference
19         public MenuItem()
20         {
21             InitializeComponent();
22         }
23
24         private string name_;
25         private string price_;
26         private Image image_;
27
28         1 reference
29         public string NameItem
30         {
31             get { return name_; }
32             set { name_ = value; ItemTitle.Text = value; }
33         }
34
35         1 reference
36         public string PriceItem
37         {
38             get { return price_; }
39             set { price_ = value; ItemPrice.Text = value; }
40         }
41
42         1 reference
43         public Image ImageItem
44         {
45             get { return image_; }
46             set { image_ = value; ItemPicture.Image = value; }
47         }
48
49         1 reference
50         private void AddItem_Click(object sender, EventArgs e)
51         {
52             currentOrderModify.InsertItem(ItemTitle.Text, float.Parse(ItemPrice.Text));
53             currentOrderModify.theMenu.currentOrder_Tick();
54         }
55     }
56 }
```

- 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

```
7 using System.Text;
8 using System.Threading.Tasks;
9 using System.Windows.Forms;
10
11 namespace Restaurant_Contactless_Dining_System
12 {
13     3 references
14     public partial class OrderedItem : UserControl
15     {
16         Order currentOrderModify = CompleteMenu.currentOrder;
17
18         1 reference
19         public OrderedItem()
20         {
21             InitializeComponent();
22
23             private string name_;
24             private float price_;
25             private int quantity_;
26
27         1 reference
28         public string NameItem
29         {
30             get { return name_; }
31             set { name_ = value; ItemTitle.Text = value; }
32         }
33
34         1 reference
35         public float PriceItem
36         {
37             get { return price_; }
38             set { price_ = value; priceLabel.Text = $"{value:0.00}"; }
39         }
40
41         1 reference
42         public int QuantityItem
43         {
44             get { return quantity_; }
45             set { quantity_ = value; QuantityLabel.Text = quantity_.ToString(); }
46         }
47
48         1 reference
49         private void quantityAdd_Click(object sender, EventArgs e)
50         {
51             currentOrderModify.InsertItem(ItemTitle.Text, 0);
52             currentOrderModify.theMenu.currentOrder_Tick();
53         }
54
55         1 reference
56         private void quantityMinus_Click(object sender, EventArgs e)
57         {
58             currentOrderModify.DecreaseItem(ItemTitle.Text);
59             currentOrderModify.theMenu.currentOrder_Tick();
60         }
61     }
62 }
```

- 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

```
7 namespace Restaurant_Contactless_Dining_System
8 {
9     3 references
10    public class Order
11    {
12        public string[] items = new string[100];
13        public int[] quantity = new int[100];
14        public float[] price = new float[100];
15        public int size = 0;
16
17        public CompleteMenu theMenu;
18
19        1 reference
20        public CompleteMenu setParent
21        {
22            set
23            {
24                theMenu = value;
25            }
26        }
27
28        2 references
29        public float GetTotal()
30        {
31            float total = 0;
32
33            for(int i = 0; i < size; i++)
34            {
35                total += quantity[i] * price[i];
36            }
37
38            return total;
39        }
40
41        2 references
42        public void InsertItem(string val, float price_)
43        {
44            if (size < 1)
45            {
46                items[0] = val;
47                quantity[0] = 1;
48                price[0] = price_;
49                size++;
50                return;
51            }
52
53            int i = 0;
54            while(i < size)
55            {
56                if (items[i].Equals(val))
57                {
58                    quantity[i]++;
59                    return;
60                }
61                i++;
62            }
63        }
64    }
65 }
```

- 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.

```

61     quantity[size] = 1;
62     price[size] = price_;
63     size++;
64 }
65
66 1 reference
67 public void DecreaseItem(string val)
68 {
69     if (size < 1)
70     {
71         return;
72     }
73     int i = 0;
74     while (i < size)
75     {
76         if (items[i].Equals(val))
77         {
78             quantity[i]--;
79             break;
80         }
81         i++;
82     }
83
84     if (quantity[i] < 1)
85     {
86         int j = i;
87         while (j < size-1)
88         {
89             items[j] = items[j + 1];
90             quantity[j] = quantity[j + 1];
91             price[j] = price[j + 1];
92
93             j++;
94         }
95
96         items[j] = "";
97         size--;
98     }
99
100     return;
101 }
102
103 3 references
104 public void ClearAll()
105 {
106     for (int i = 0; i < size; i++)
107     {
108         items[i] = "";
109         quantity[i] = 0;
110         price[i] = 0;
111     }
112
113     size = 0;
114     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

```
13 {
14     7 references
15     public partial class PasswordChecker : Form
16     {
17         SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
18         SQLiteCommand cmd = new SQLiteCommand();
19         SQLiteDataReader dr;
20
21         4 references
22         public PasswordChecker()
23         {
24             InitializeComponent();
25
26         2 references
27         private void SubmitPassword_Click(object sender, EventArgs e)
28         {
29             con.Open();
30             cmd.Connection = con;
31             cmd.CommandText = "SELECT password FROM restaurant_info";
32             dr = cmd.ExecuteReader();
33
34             if (dr.HasRows)
35             {
36                 while (dr.Read())
37                 {
38                     if (dr["password"].Equals(Password.Text))
39                     {
40                         this.DialogResult = DialogResult.Yes;
41                         this.Close();
42                     }
43                     else
44                     {
45                         MessageBox.Show("Wrong password, please try again.", "Password Incorrect", MessageBoxButtons.OK, MessageBoxIcon.Error);
46                         this.DialogResult = DialogResult.No;
47                         this.Close();
48                     }
49                 }
50             }
51             dr.Close();
52             con.Close();
53         }
54
55         1 reference
56         private void Password_KeyDown(object sender, KeyEventArgs e)
57         {
58             if (e.KeyCode == Keys.Enter)
59             {
60                 SubmitPassword_Click(this, new EventArgs());
61             }
62         }
63     }
64 }
```

- 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

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5 using System.Windows.Forms;
6 using System.Data.SQLite;
7
8 namespace Restaurant_Contactless_Dining_System
9 {
10     References
11     static class Program
12     {
13         /// <summary>
14         /// The main entry point for the application.
15         /// </summary>
16         [STAThread]
17         References
18         static void Main()
19         {
20             Application.EnableVisualStyles();
21             Application.SetCompatibleTextRenderingDefault(false);
22
23             SQLiteConnection con = new SQLiteConnection("Data Source=menusystem.db;Version=3");
24             SQLiteCommand cmd = new SQLiteCommand();
25             SQLiteDataReader dr;
26
27             con.Open();
28
29             cmd.Connection = con;
30             cmd.CommandText = "SELECT * FROM restaurant_info";
31             dr = cmd.ExecuteReader();
32
33             if (!dr.HasRows)
34             {
35                 con.Close();
36                 Application.Run(new SetupForm());
37             }
38             else
39             {
40                 con.Close();
41                 Application.Run(new MainMenu());
42             }
43         }
44     }
45 }
46
47
```

- 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.