Fall 2021

Guitar Store Inventory

Alexander DiDonato
ad212@uakron.edu

Guitar Inventory System

Alex DiDonato

Senior Programming Projects

Spring 2021

What is the problem

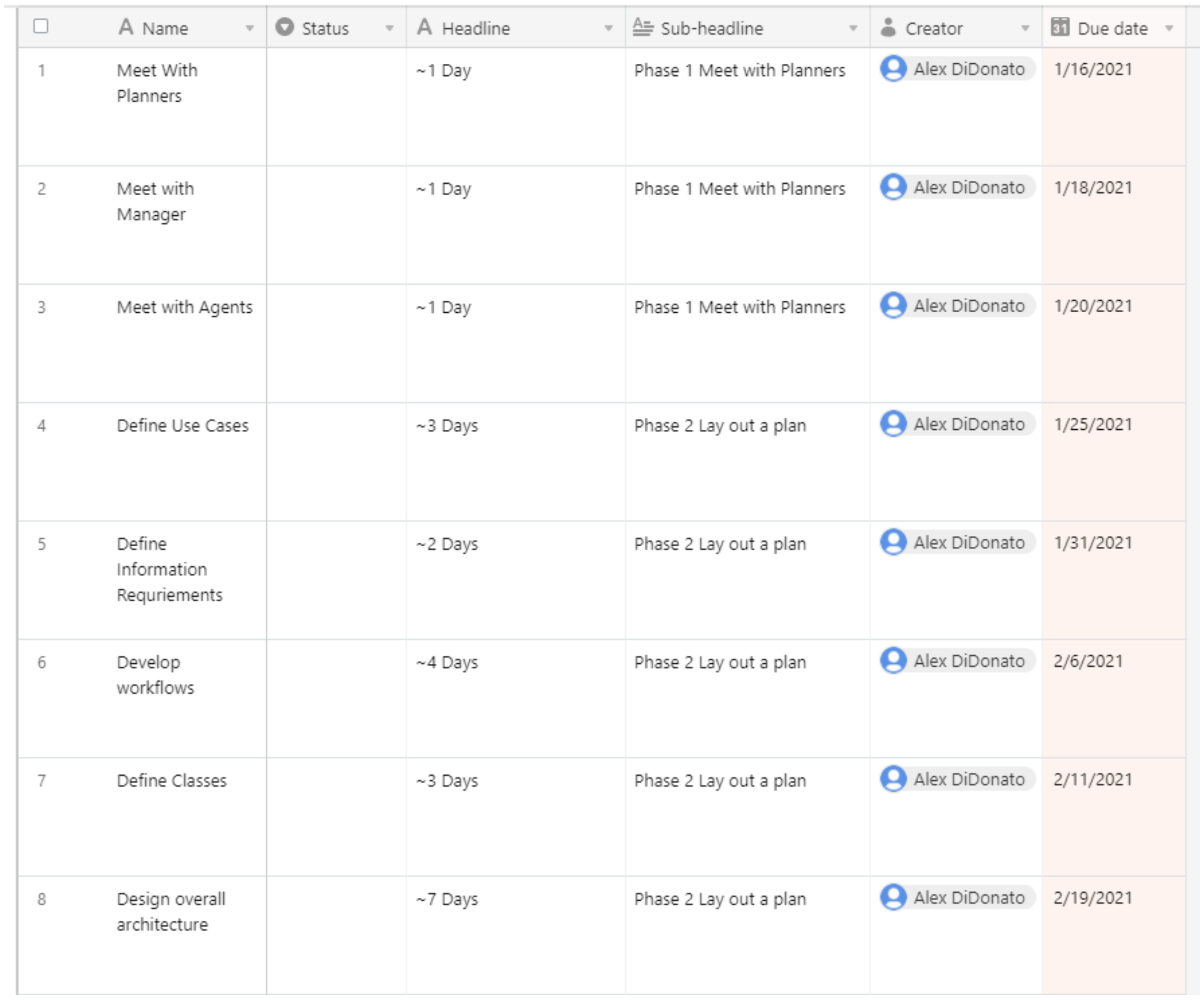
The problem I want to solve is that I want to make a better program to track our inventory and give our customers a chance to be order even if it is out of stock. I also wanted a priority list so people can still order an item even out of stock. I am trying to find a way to integrate a database into a website so customers can see when an item would go out of stock and how many are left. I chose this project because I have a profound interest in guitars and doing inventory as I work at Drug Mart. I also might want to open up a guitar business in the future and this will give me an idea on how to do my website. I intend to solve it by making a website that implemented a database to track the amount of stock left.

Databases

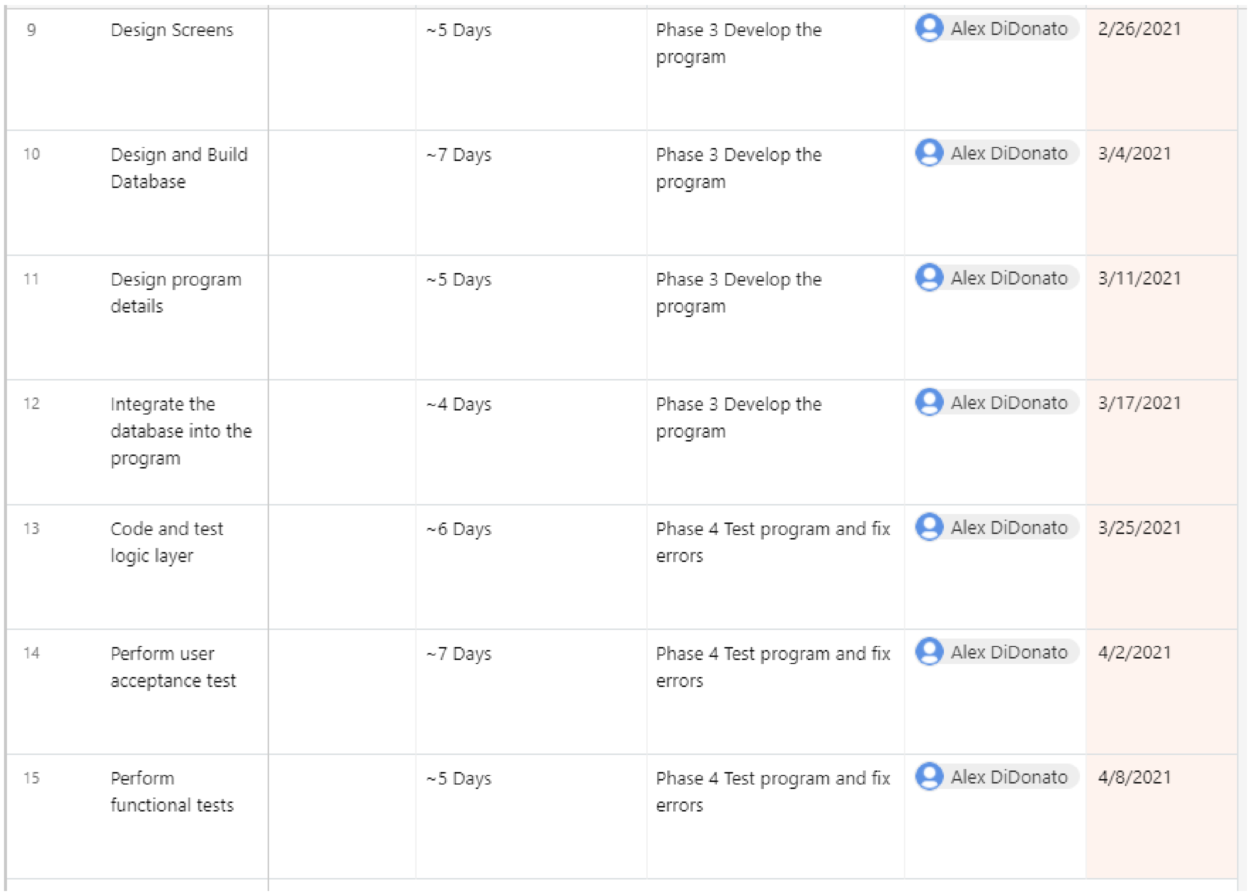
Integration

Employee side

Website



	Name	Status	Headline	Sub-headline	Creator	Due date
1	Meet With Planners		~1 Day	Phase 1 Meet with Planners	Alex DiDonato	1/16/2021
2	Meet with Manager		~1 Day	Phase 1 Meet with Planners	Alex DiDonato	1/18/2021
3	Meet with Agents		~1 Day	Phase 1 Meet with Planners	Alex DiDonato	1/20/2021
4	Define Use Cases		~3 Days	Phase 2 Lay out a plan	Alex DiDonato	1/25/2021
5	Define Information Requriements		~2 Days	Phase 2 Lay out a plan	Alex DiDonato	1/31/2021
6	Develop workflows		~4 Days	Phase 2 Lay out a plan	Alex DiDonato	2/6/2021
7	Define Classes		~3 Days	Phase 2 Lay out a plan	Alex DiDonato	2/11/2021
8	Design overall architecture		~7 Days	Phase 2 Lay out a plan	Alex DiDonato	2/19/2021



9	Design Screens		~5 Days	Phase 3 Develop the program	Alex DiDonato	2/26/2021
10	Design and Build Database		~7 Days	Phase 3 Develop the program	Alex DiDonato	3/4/2021
11	Design program details		~5 Days	Phase 3 Develop the program	Alex DiDonato	3/11/2021
12	Integrate the database into the program		~4 Days	Phase 3 Develop the program	Alex DiDonato	3/17/2021
13	Code and test logic layer		~6 Days	Phase 4 Test program and fix errors	Alex DiDonato	3/25/2021
14	Perform user acceptance test		~7 Days	Phase 4 Test program and fix errors	Alex DiDonato	4/2/2021
15	Perform functional tests		~5 Days	Phase 4 Test program and fix errors	Alex DiDonato	4/8/2021

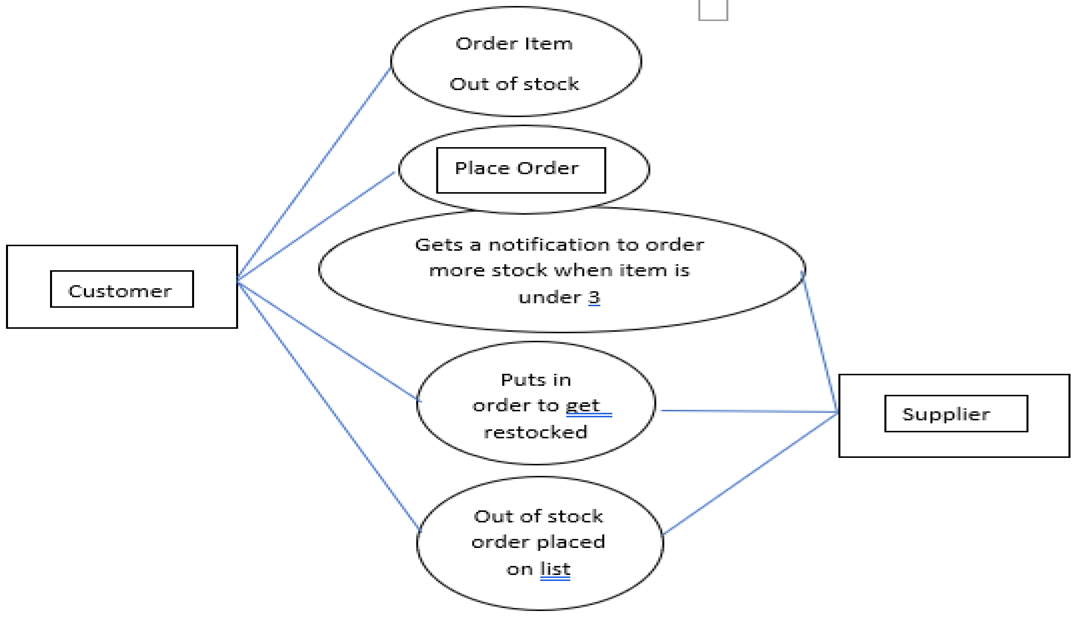
Analysis

A

Use Cases

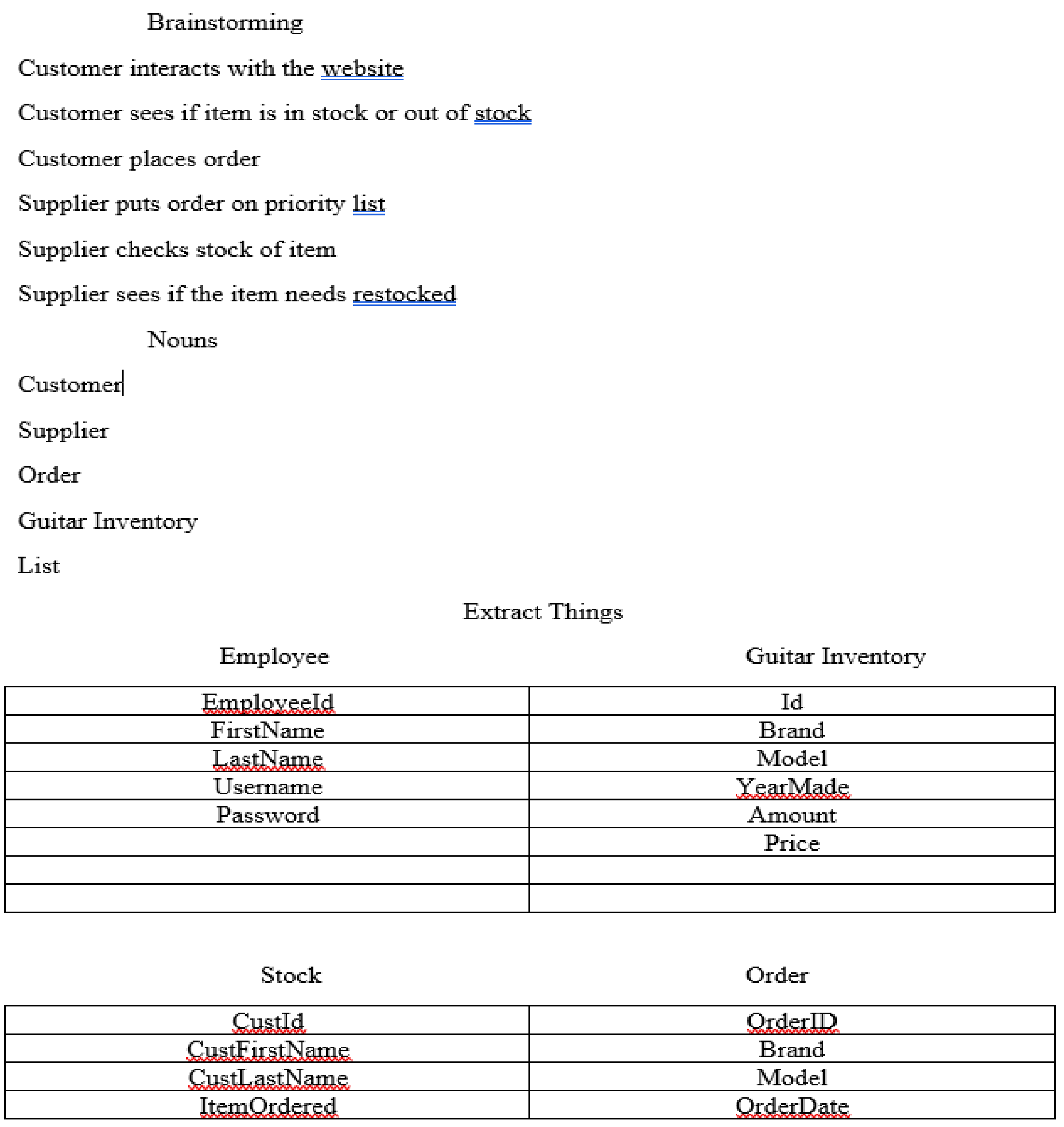
Using the event decomposition technique, a use case will be when an item gets to 3 left, the system will trigger an alert to send to the supplier to get more of the stock in as soon as possible. Another possible use case is when the item gets down to 10 because that is when the information system will put out a message saying there is 10 left. The second use case will be if the amount left gets under 3, that will be when the program puts a notice out to have the employees put out an order for said item. The final use case would be if a user put in an order on an out of stock item, the system will put the order on a priority list that says that person will get it as soon as it arrives.

Customer	Order an item Place an order in the PriorityList
Supplier	Puts in order to get item restocked Gets notified if an item is 10 or less Gets notice to place order if stock drops below 3


Order Item
Out of stock
Place Order
Gets a notification to order more stock when item is under 3
Customer
Puts in order to get restocked
Supplier
Out of stock order placed on list

I used this to analyze how users and suppliers would act in some scenarios and how events would play out. The customer would buy an item and the supplier would note that the amount of a certain guitar would decrease. I put in things that both would do and create possible scenarios that this application would be used for.

B



Brainstorming

Customer interacts with the website

Customer sees if item is in stock or out of stock

Customer places order

Supplier puts order on priority list

Supplier checks stock of item

Supplier sees if the item needs restocked

Nouns

Customer

Supplier

Order

Guitar Inventory

List

Extract Things

Employee	Guitar Inventory
EmployeeId	Id
FirstName	Brand
LastName	Model
Username	YearMade
Password	Amount
	Price

Stock	Order
CustId	OrderID
CustFirstName	Brand
CustLastName	Model
ItemOrdered	OrderDate

This was the brainstorm phase for all of my use cases. I extracted the common nouns like customer, order, and stock and created use cases. I also extracted things from each of the nouns to form database tables and the foundation of the project.

C

The Domain classes that I decided to use were to identify certain aspects of a guitar, like brand name, model, name, and year made. Other tables used different domain classes, such as employee and customer first and last name to make sure that the correct person is identified. The order status table used classes like itemOrdered and Order date to correctly identify the date of the order. All tables also use Id as the primary key to accurately keep track of all of the data. All of the domain classes will be listed below here.

Employee

EmployeeId, FirstName, LastName, Username, Password

GuitarInventory

Id, Brand, Model, YearMade, Amount, Price

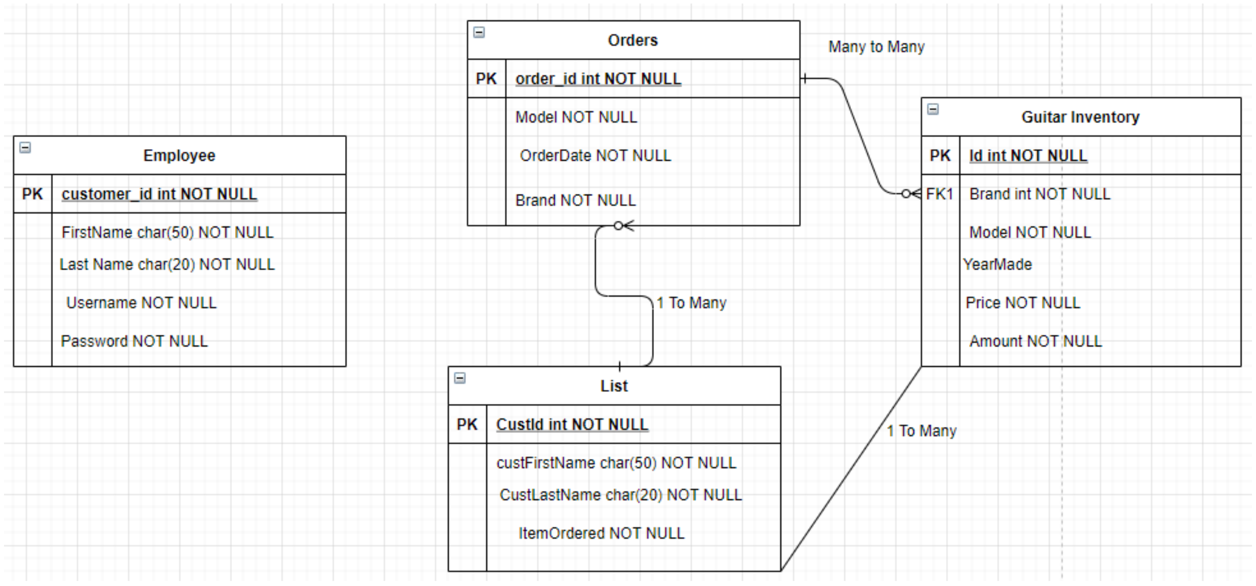
OrderStatus

OrderId, OrderDate, Brand, Model

PriorityList

CustId, CustFirstName, CustLastName, ItemOrdered

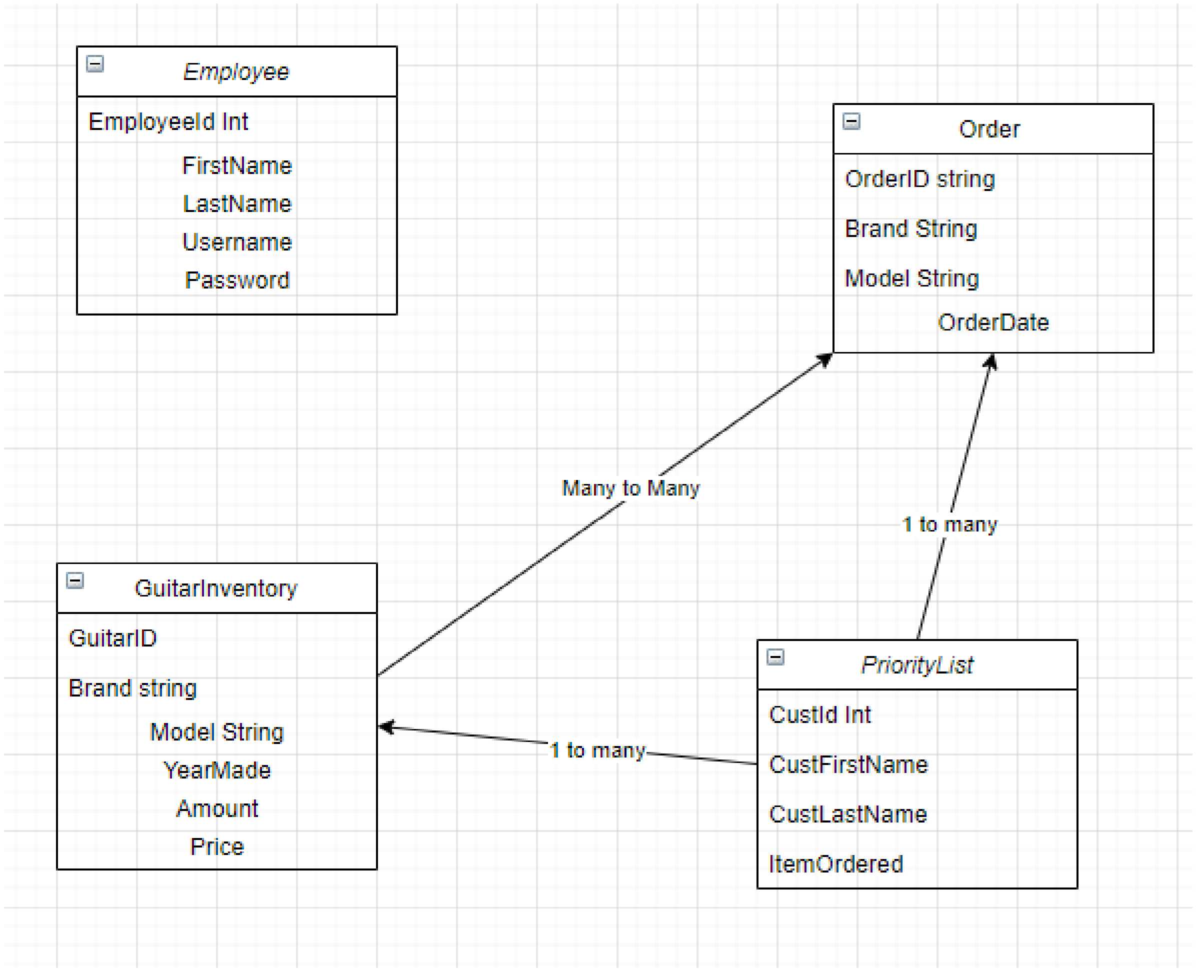
D



This ERD was a rough draft of how I was originally going to do this project. The shipments part of it was changed to Guitar Inventory because I had no way to look at guitar inventory. The List represents the priority list of people who ordered an item that was out of stock, which is related to the orders because that shows that a customer ordered an item out of stock. I also related orders to every entity because that is the centerpiece of keeping charge of stock.

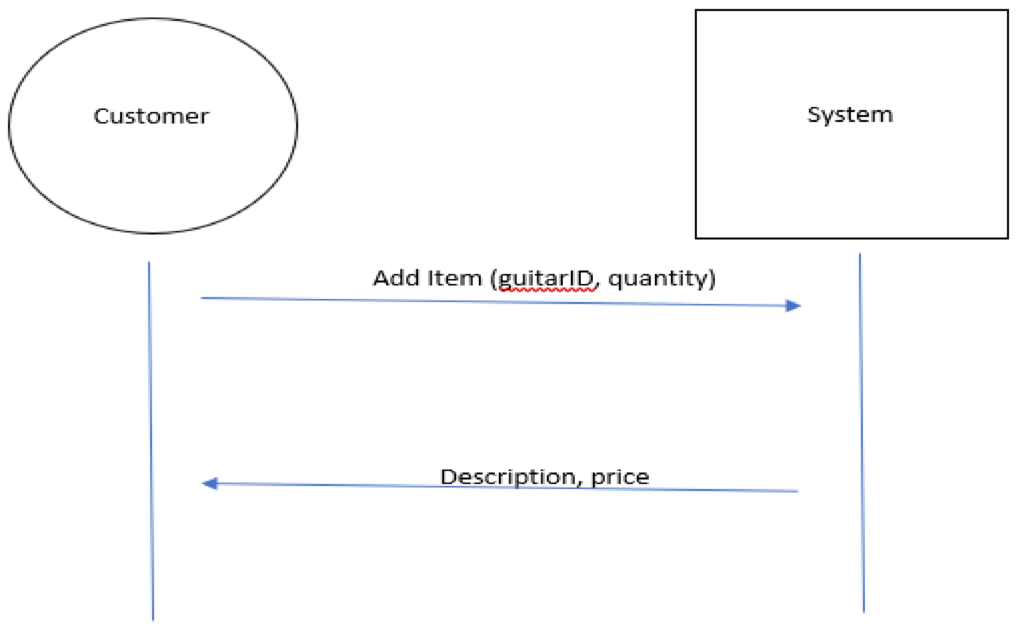
Design

A

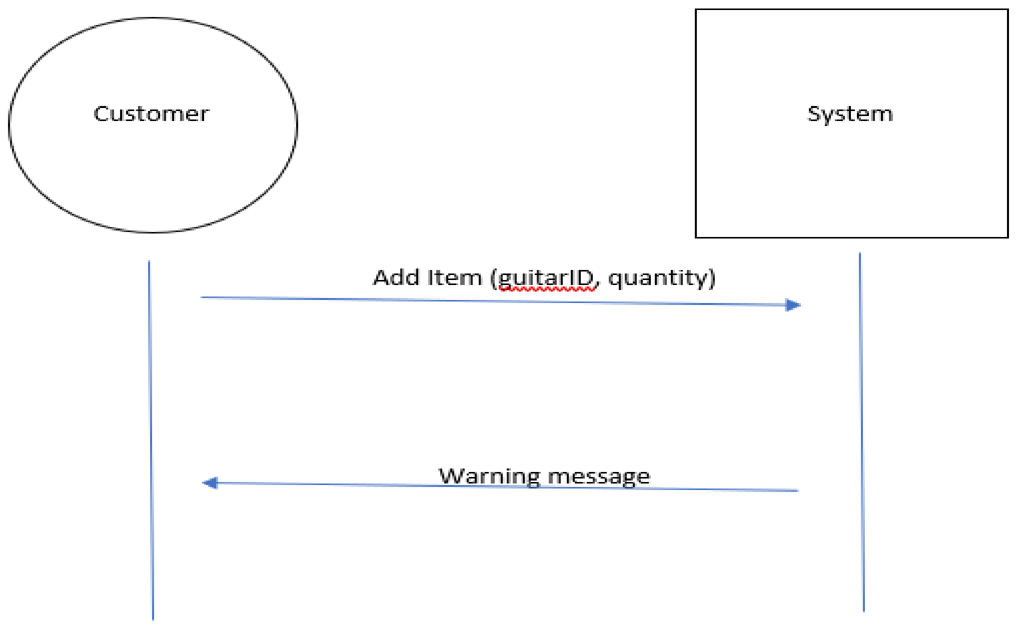


This is the Domain Model class diagram with the tables for the database. I have the Employee class separate because it has no relation to anything and its sole purpose is for security. The other three tables are for keeping track of inventory and who has ordered something out of stock. The order has many to many with the Guitar Inventory because many guitars can have many orders at once. Priority List has one to many because one person could have one guitar or many guitars.

5



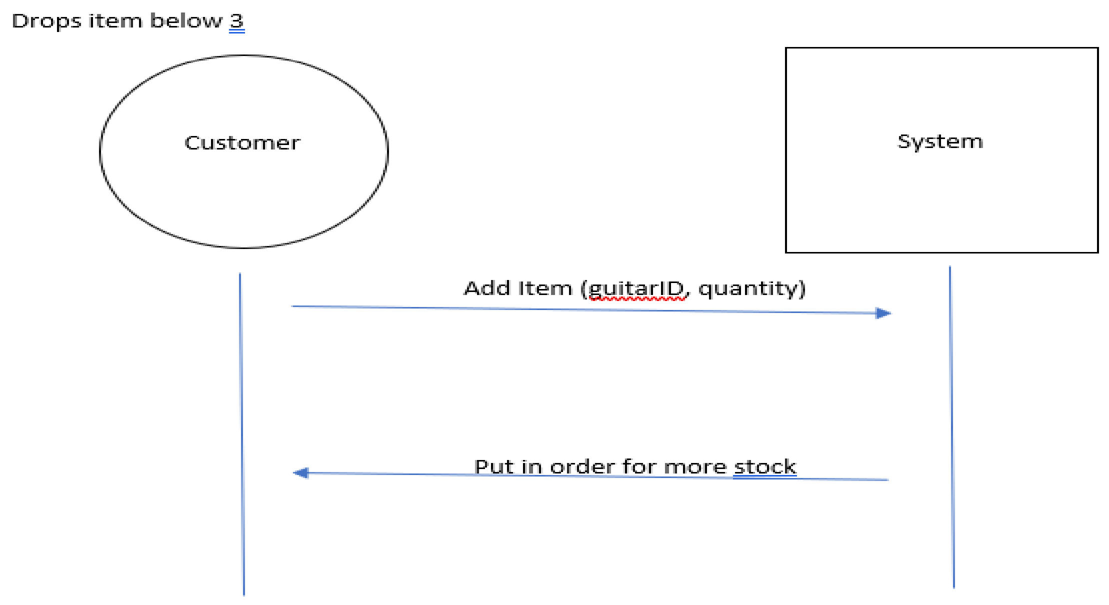
Drops item below 10

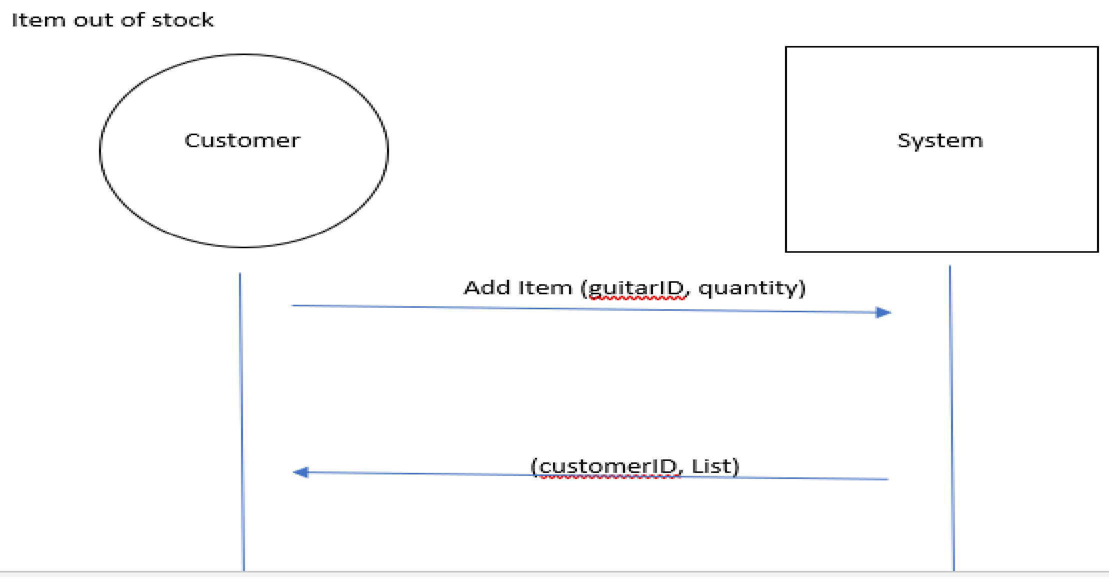


B

This is the state machine diagram that states that when an item goes to 10 or 3, it will do these actions. When the amount hits 10, a message should pop up saying, “only 10 left, hurry before it is too late.” When an Item hits 3, a message on the employee side should say, “only 3 left, place an order for this item.”

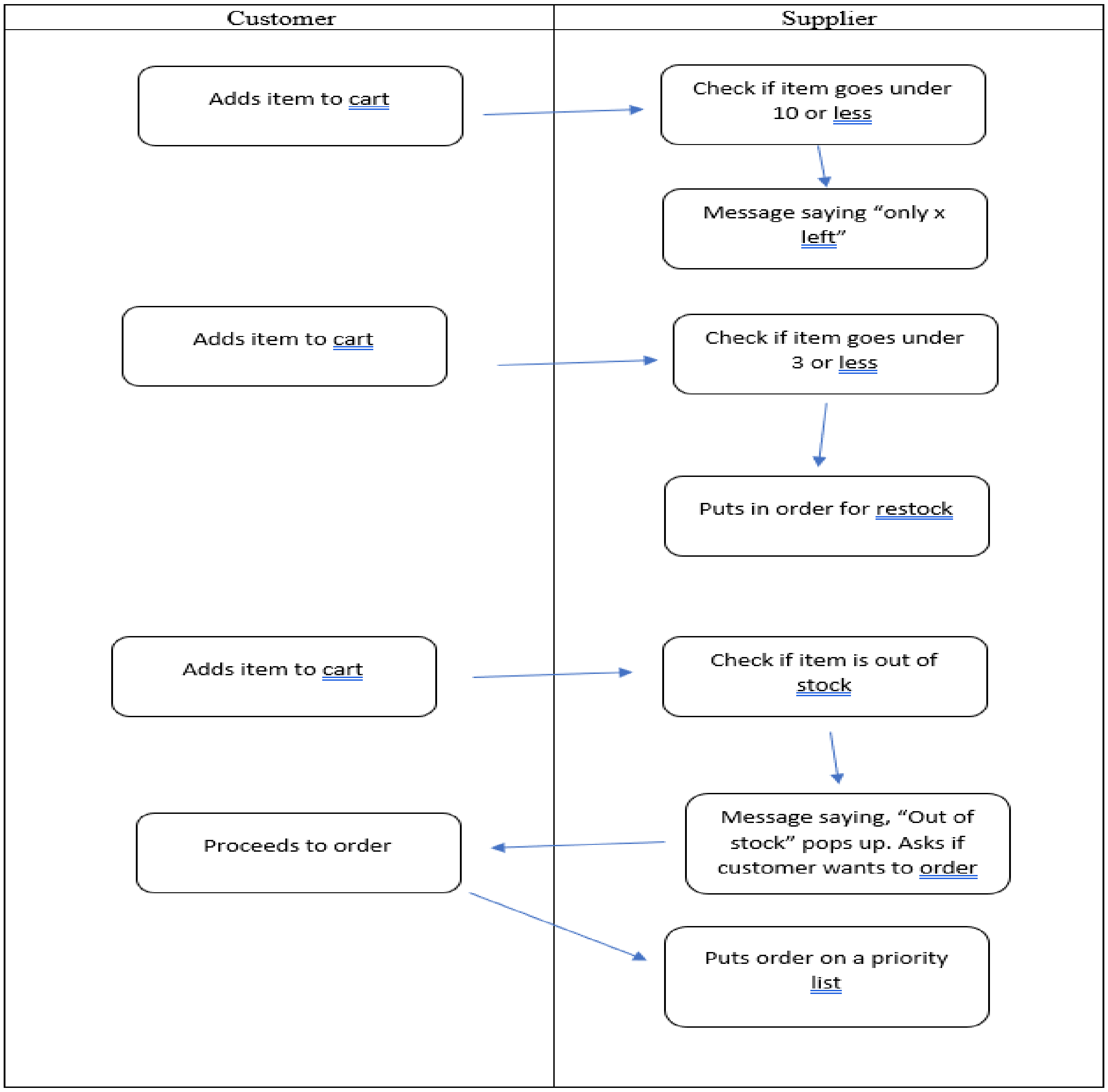
C





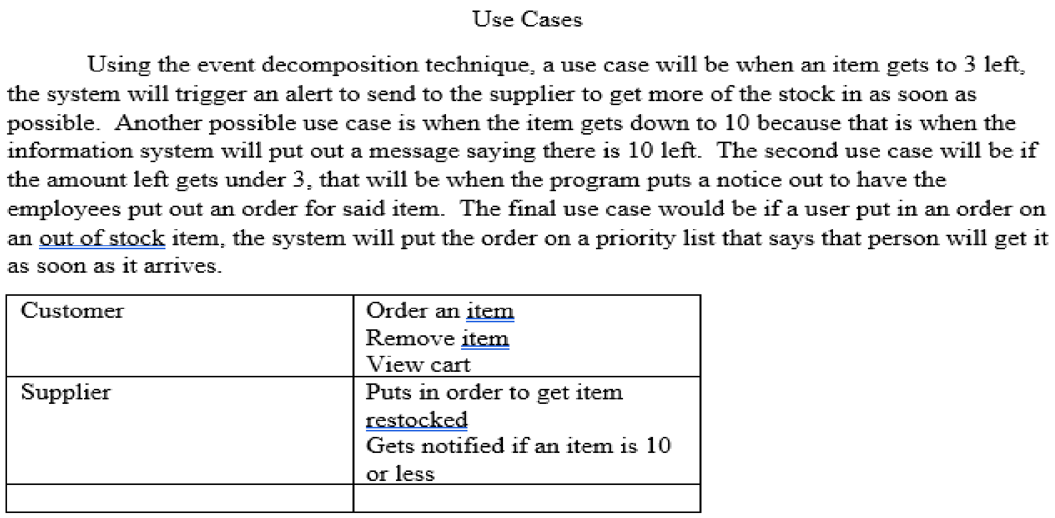
When an Item hits 3, a message on the employee side should say, “only 3 left, place an order for this item.” The bottom one is if the customer clicks buy it now and there is no more guitars. The system will pull up the priority list and the customer fills out his or her information.

D


Customer
Supplier
Adds item to cart
Check if item goes under 10 or less
Message saying "only x left"
Adds item to cart
Check if item goes under 3 or less
Puts in order for restock
Adds item to cart
Check if item is out of stock
Message saying, "Out of stock" pops up. Asks if customer wants to order
Proceeds to order
Puts order on a priority list

This is the activity diagram stating all possibilities when a user clicks buy it now. The system will check if the item goes under 10 or less and act according to the programming. If it is 10 or less, pull up the only x left message. If it goes under 3, the program tells the employee side to order more stock. If there are none left, the Priority list page pulls up and the customer enters their information to be sent to the priority list table.

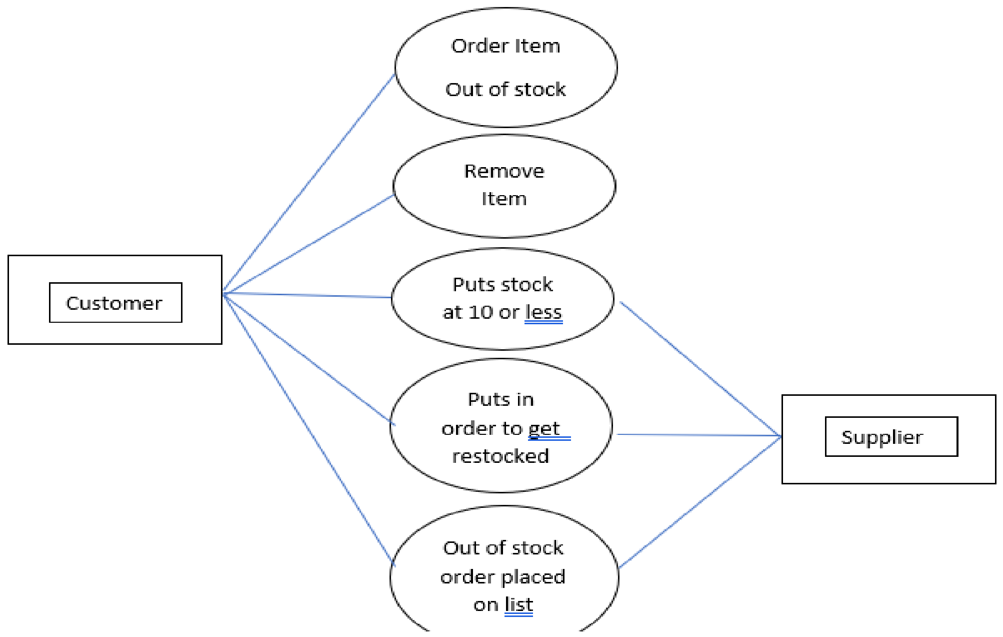
E



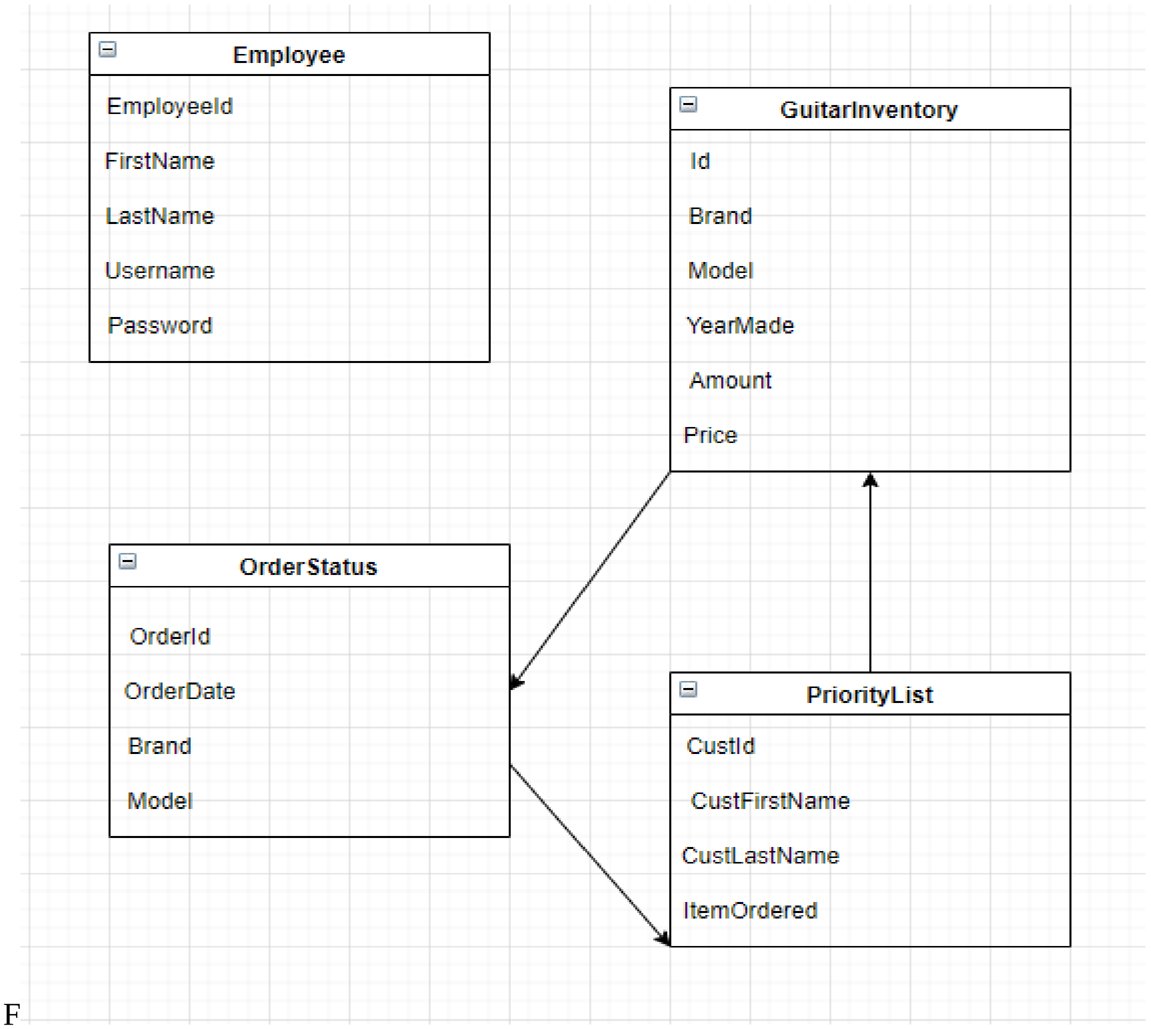
Use Cases

Using the event decomposition technique, a use case will be when an item gets to 3 left, the system will trigger an alert to send to the supplier to get more of the stock in as soon as possible. Another possible use case is when the item gets down to 10 because that is when the information system will put out a message saying there is 10 left. The second use case will be if the amount left gets under 3, that will be when the program puts a notice out to have the employees put out an order for said item. The final use case would be if a user put in an order on an out of stock item, the system will put the order on a priority list that says that person will get it as soon as it arrives.

Customer	Order an item Remove item View cart
Supplier	Puts in order to get item restocked Gets notified if an item is 10 or less



This is the use case table that puts together the use cases into one table. This tells us all of the possibilities and interactions between the customer and the supplier. All possibilities including what happens as the amount decreases and eventually hits 0. The supplier will interact on the last three cases as the customer directly influences the system by hitting buy it now.

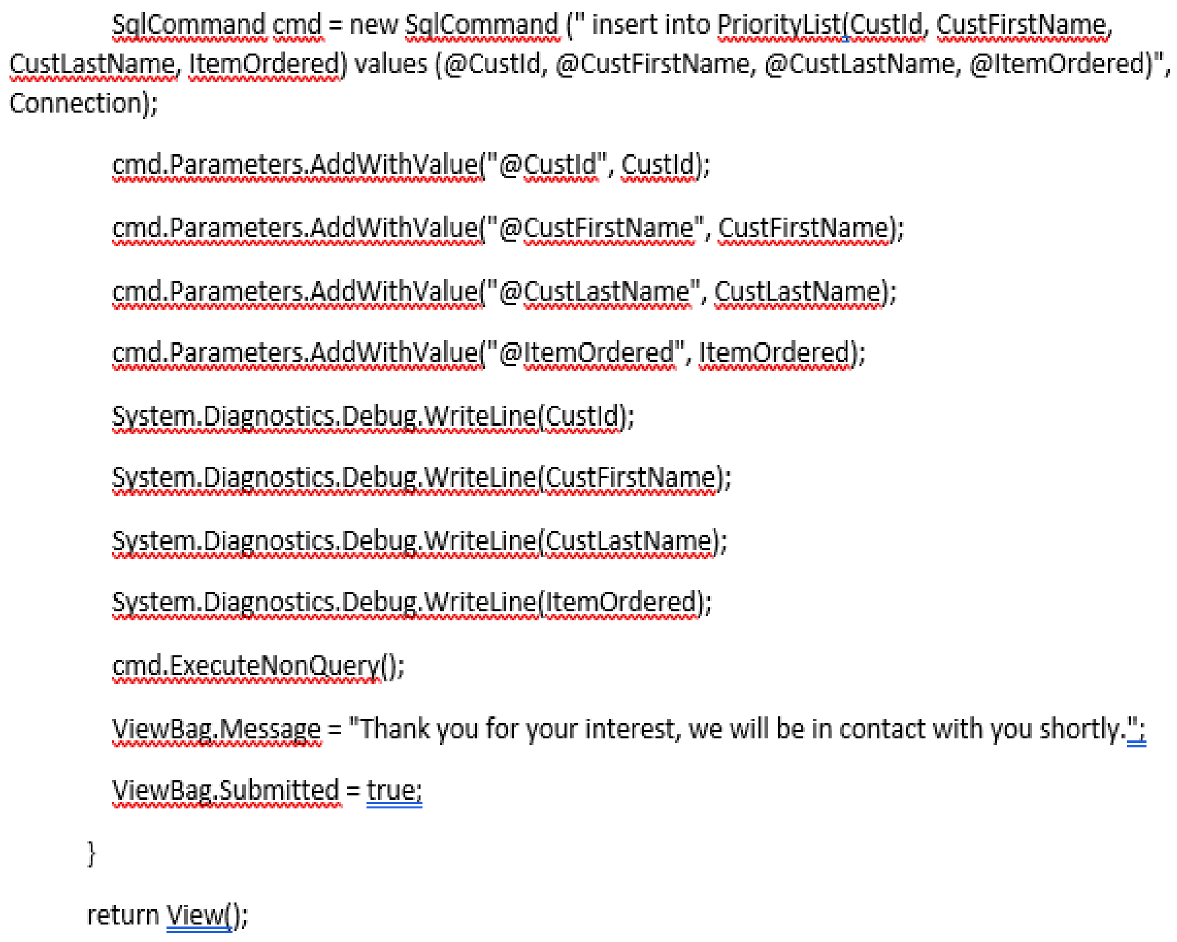


F

This is the normalized Database schema with the tables for the database. I have the Employee class separate because it has no relation to anything and its sole purpose is for security. The other three tables are for keeping track of inventory and who has ordered something out of stock. The order has many to many with the Guitar Inventory because many guitars can have many orders at once. Priority List has one to many because one person could have one guitar or many guitars.

G

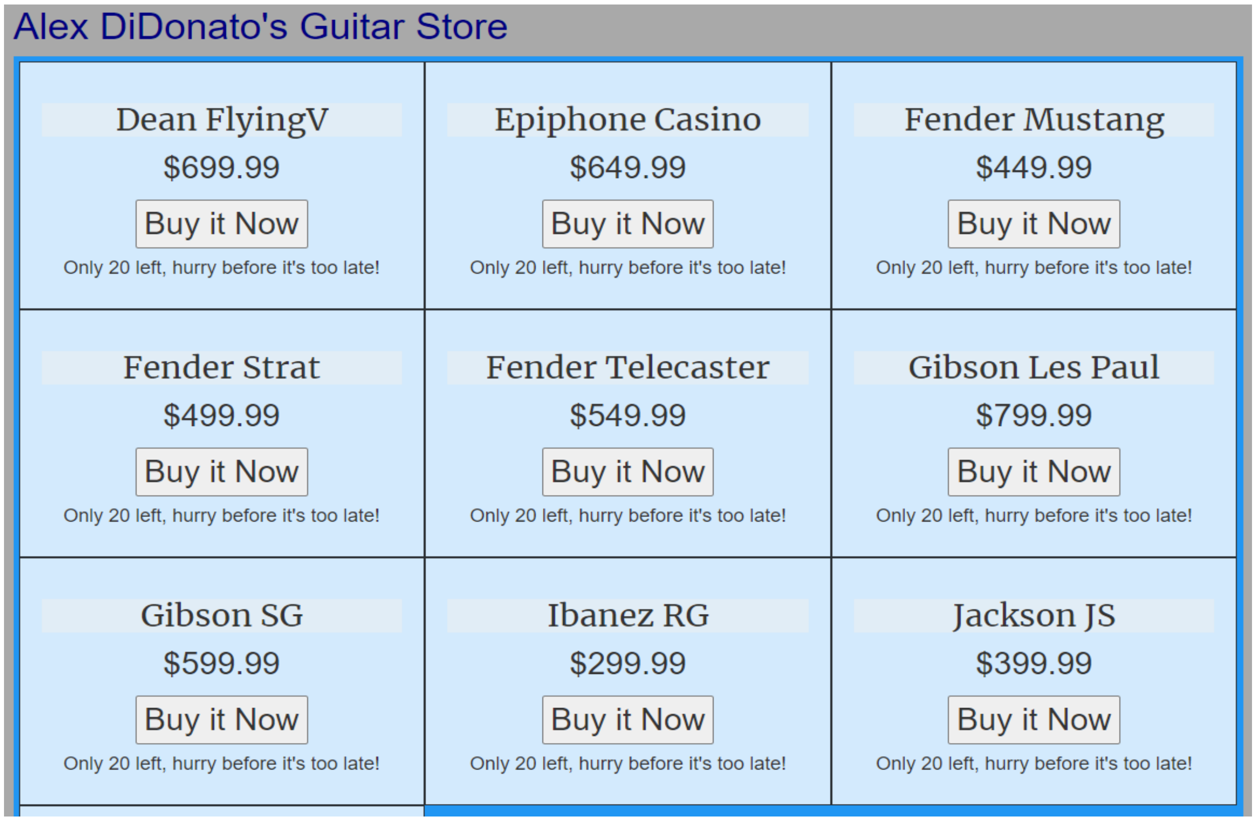
```
  public class HomeController : Controller
  {
    String ConnectionString;
    SqlConnection Connection;


    public HomeController()
    {
       ConnectionString = (@"Data
Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated
Security=True");
       Connection = new SqlConnection(ConnectionString);
    }
    public ActionResult Index()
    {
      SqlCommand cmd = new SqlCommand(" select * from GuitarInventory order by Brand, Model",
Connection);
      Connection.Open();
      SqlDataAdapter adpt = new SqlDataAdapter(cmd);
      adpt.SelectCommand = cmd;
      DataSet ds = new DataSet("GuitarInventory");
      adpt.Fill(ds);
      Connection.Close();
      List<Models.GuitarInventory> GuitarList = new List<Models.GuitarInventory>();
      foreach (DataRow dr in ds.Tables[0].Rows)
      {
        GuitarList.Add(new Models.GuitarInventory
        {
          Id = Convert.ToInt32(dr["Id"]),
          Brand = Convert.ToString(dr["Brand"]),
          Model = Convert.ToString(dr["Model"]),
          YearMade = Convert.ToInt32(dr["YearMade"]),
                Amount = Convert.ToInt32(dr["Amount"]),
                Price = Convert.ToString(dr["Price"])
            });
        }
        ViewData["GuitarList"] = GuitarList;
        return View();
    }

    public ActionResult Purchase()
    {
        try
        {
            int GuitarId = Convert.ToInt32(Request.Form["Id"]);
            SqlCommand command = new SqlCommand("select Amount from GuitarInventory where Id =
@id", Connection);
            Connection.Open();
            command.Parameters.AddWithValue("@id", GuitarId);
            int Amount = (Int32)command.ExecuteScalar();
           // System.Diagnostics.Debug.WriteLine(Amount);
            if (Amount > 0)
            {
                SqlCommand cmd = new SqlCommand(" update GuitarInventory set Amount = (Amount - 1)
where Id = @id", Connection);
                cmd.Parameters.AddWithValue("@id", GuitarId);
                cmd.ExecuteNonQuery();
                return RedirectToAction("Index");
            }
            else
            {
                return RedirectToAction("OrderOutOfStock");
    }
    catch(Exception ex)
    {
      return RedirectToAction("OrderOutOfStock");
    }
  }
    public ActionResult About()
  {
    ViewBag.Message = "Your application description page.";

    return View();
  }
  public ActionResult Contact()
  {
    ViewBag.Message = "Your contact page.";
    return View();
  }
  public ActionResult OrderOutOfStock()
  {
    ViewBag.Message = "Place your information here and you will be put on a Priority List.";
    ViewBag.Submitted = false;
    if (Request.Form.Count > 0)
    {
      String CustFirstName = Request.Form["CustFirstName"];
      String CustLastName = Request.Form["CustLastName"];
      String ItemOrdered = Request.Form["ItemOrdered"];
      SqlCommand command = new SqlCommand("select top 1 CustId from PriorityList Order By
CustId desc ", Connection);
      Connection.Open();
      int CustId = (Int32)command.ExecuteScalar() + 1;
```



```
            SqlCommand cmd = new SqlCommand (" insert into PriorityList(CustId, CustFirstName,
    CustLastName, ItemOrdered) values (@CustId, @CustFirstName, @CustLastName, @ItemOrdered)",
    Connection);
            cmd.Parameters.AddWithValue("@CustId", CustId);
            cmd.Parameters.AddWithValue("@CustFirstName", CustFirstName);
            cmd.Parameters.AddWithValue("@CustLastName", CustLastName);
            cmd.Parameters.AddWithValue("@ItemOrdered", ItemOrdered);
            System.Diagnostics.Debug.WriteLine(CustId);
            System.Diagnostics.Debug.WriteLine(CustFirstName);
            System.Diagnostics.Debug.WriteLine(CustLastName);
            System.Diagnostics.Debug.WriteLine(ItemOrdered);
            cmd.ExecuteNonQuery();
            ViewBag.Message = "Thank you for your interest, we will be in contact with you shortly.";
            ViewBag.Submitted = true;
        }
        return View();
```

This is the Psuedocode for the website and how it is supposed to interact with the database. This code is what sends data to the Employee portal and updates it there as well as on the website. This code also has queries to make sure that the right item is being modified when the buy it now button is clicked for a certain guitar. I also have a query to check if the amount for the guitar is at zero, the following If then statement determines that if the item is at zero, cue the OrderOutOfStock page, otherwise, the program continues as usual.

H


Alex DiDonato's Guitar Store

Dean FlyingV
$699.99

Only 20 left, hurry before it's too late!

Epiphone Casino
$649.99

Only 20 left, hurry before it's too late!

Fender Mustang
$449.99

Only 20 left, hurry before it's too late!

Fender Strat
$499.99

Only 20 left, hurry before it's too late!

Fender Telecaster
$549.99

Only 20 left, hurry before it's too late!

Gibson Les Paul
$799.99

Only 20 left, hurry before it's too late!

Gibson SG
$599.99

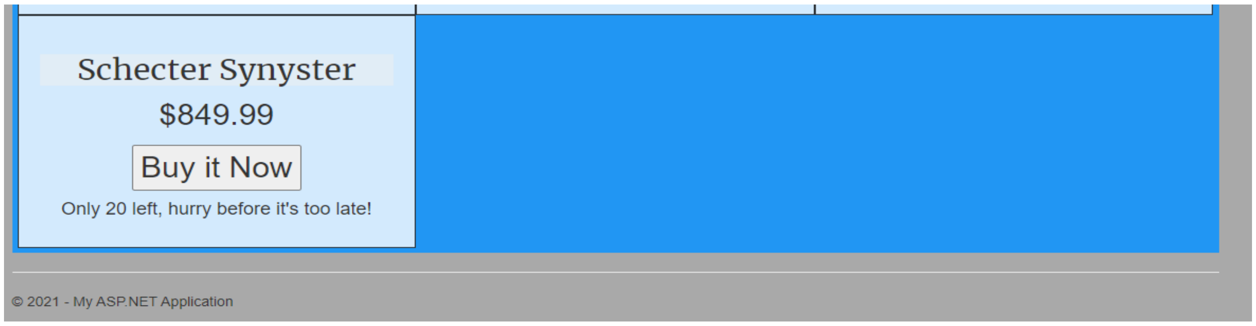
Only 20 left, hurry before it's too late!

Ibanez RG
$299.99

Only 20 left, hurry before it's too late!

Jackson JS
$399.99

Only 20 left, hurry before it's too late!

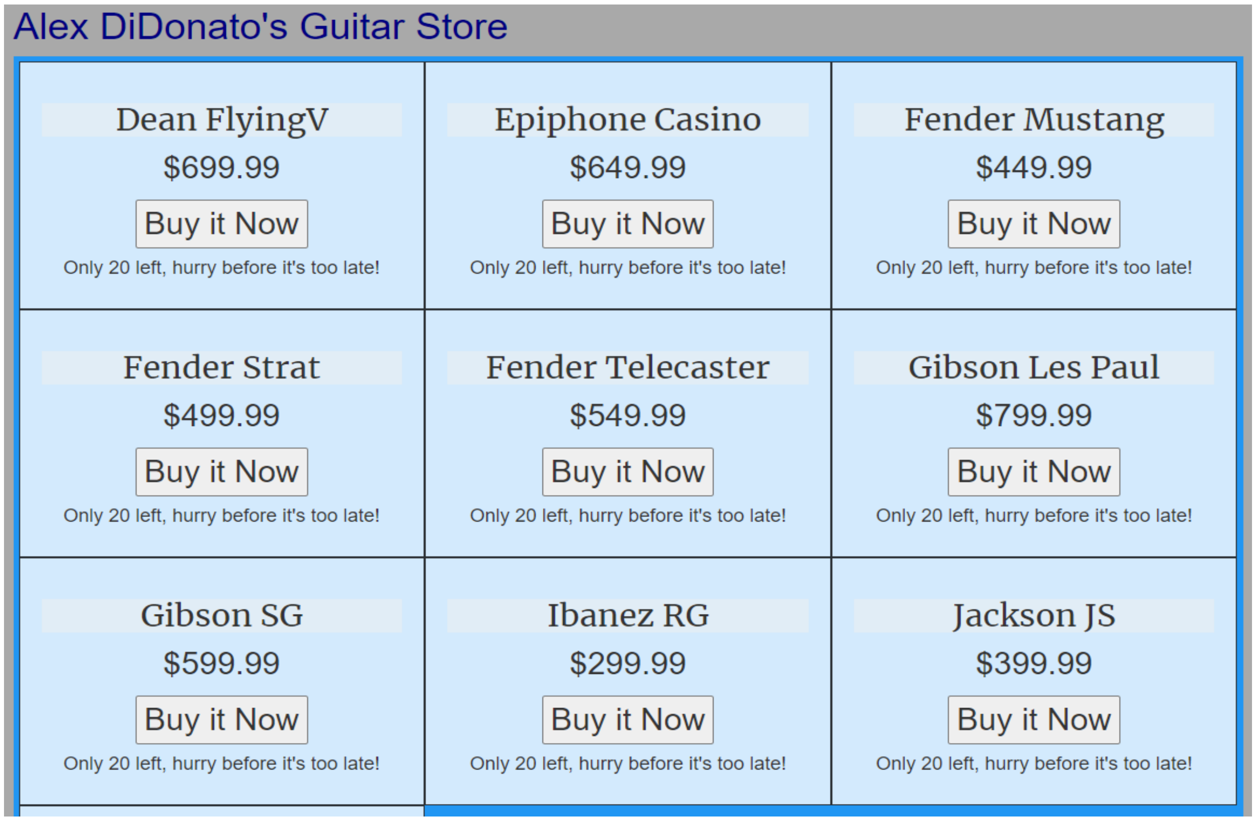

Schecter Synyster
$849.99

Only 20 left, hurry before it's too late!

© 2021 - My ASP.NET Application

This is the user interface for the customers online guitar website. I made it as simple as possible to make sure it interacted with the database. When the user clicks buy it now. The 20 on whatever product they clicked went down to 19, then 18, and so on so forth. This change was also reflected on the employee side in the guitar inventory page as it would show the change there as well.

I



Alex DiDonato's Guitar Store

Dean FlyingV
$699.99
Buy it Now
Only 20 left, hurry before it's too late!

Epiphone Casino
$649.99
Buy it Now
Only 20 left, hurry before it's too late!

Fender Mustang
$449.99
Buy it Now
Only 20 left, hurry before it's too late!

Fender Strat
$499.99
Buy it Now
Only 20 left, hurry before it's too late!

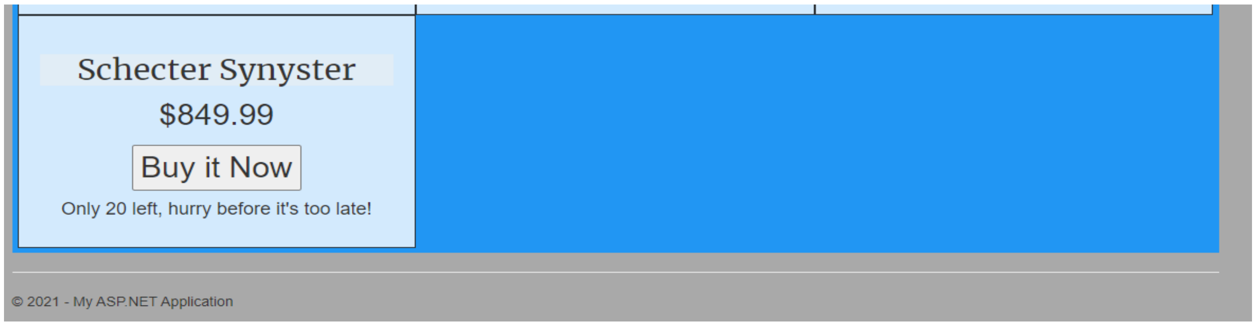
Fender Telecaster
$549.99
Buy it Now
Only 20 left, hurry before it's too late!

Gibson Les Paul
$799.99
Buy it Now
Only 20 left, hurry before it's too late!

Gibson SG
$599.99
Buy it Now
Only 20 left, hurry before it's too late!

Ibanez RG
$299.99
Buy it Now
Only 20 left, hurry before it's too late!

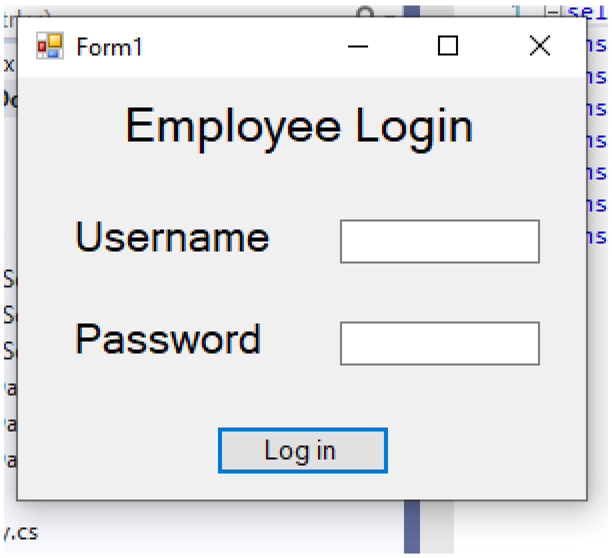
Jackson JS
$399.99
Buy it Now
Only 20 left, hurry before it's too late!



Schecter Synyster
$849.99
Buy it Now
Only 20 left, hurry before it's too late!

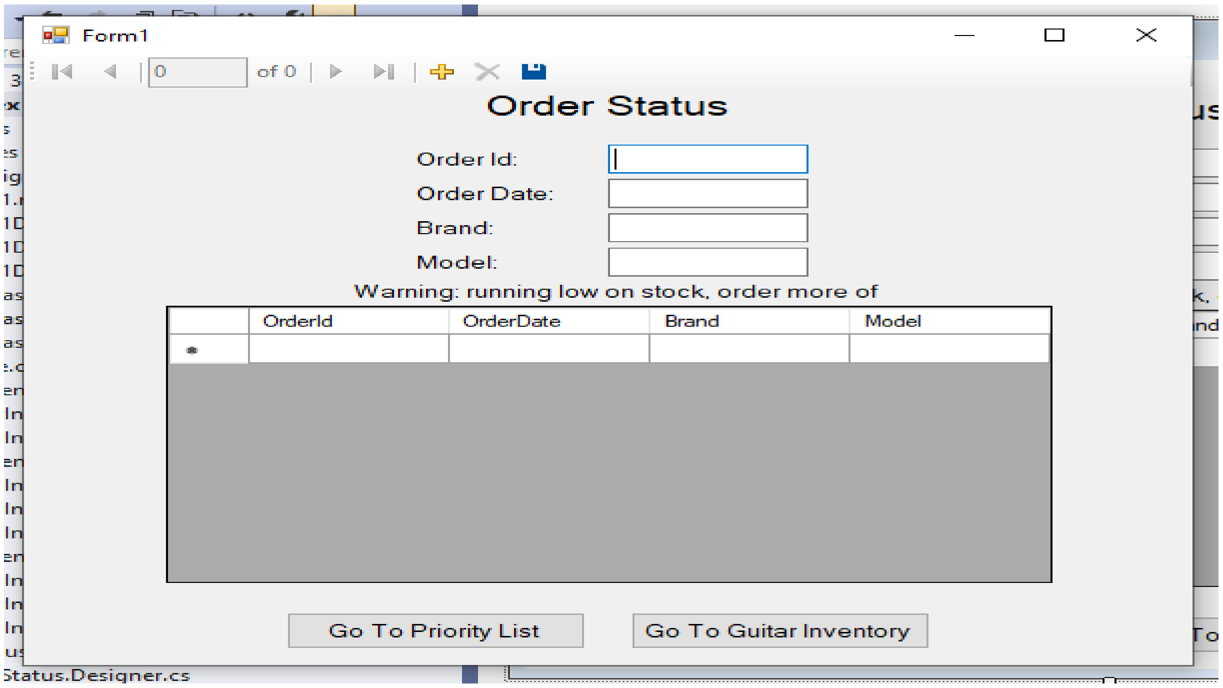
© 2021 - My ASP.NET Application

J

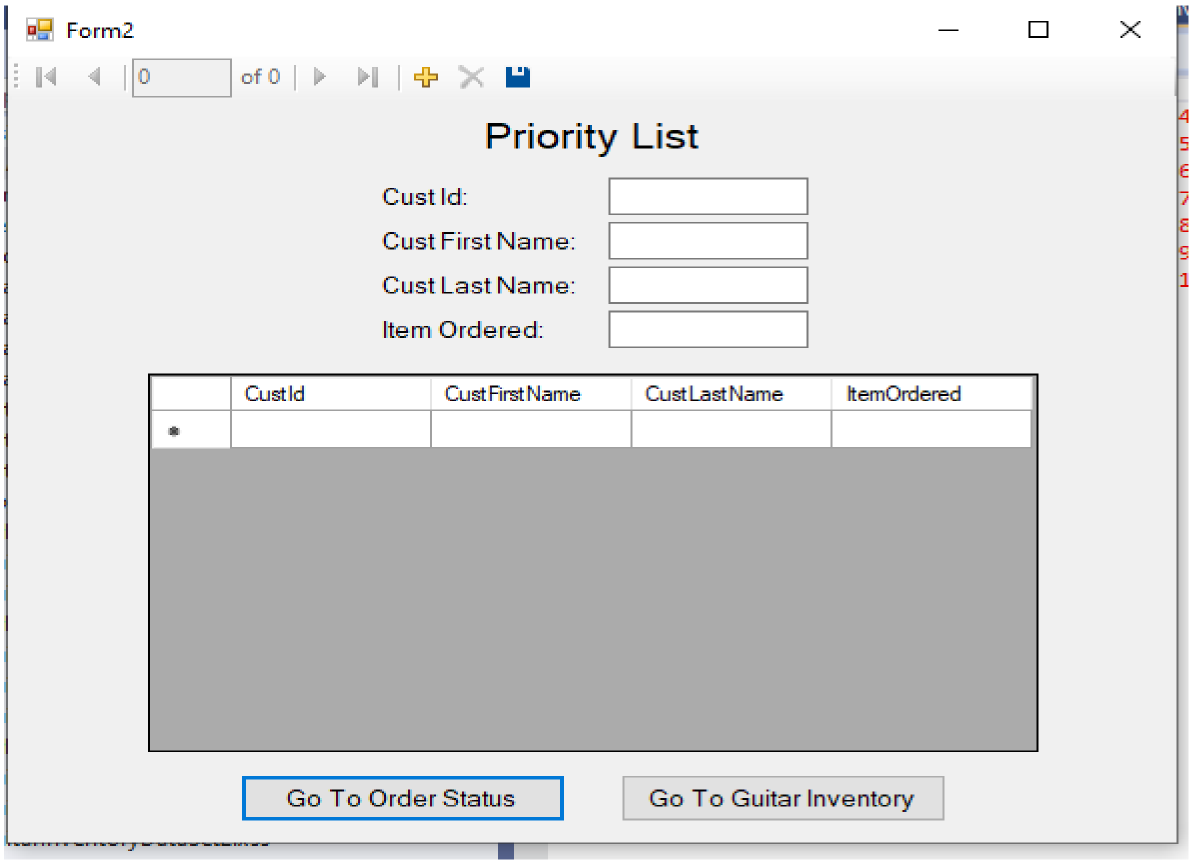


This is the security used for the employee side. This makes sure that the user logging in is in the database before continuing. If someone tries to enter in their username and password and it is not in the employee table, they will get denied access.

K



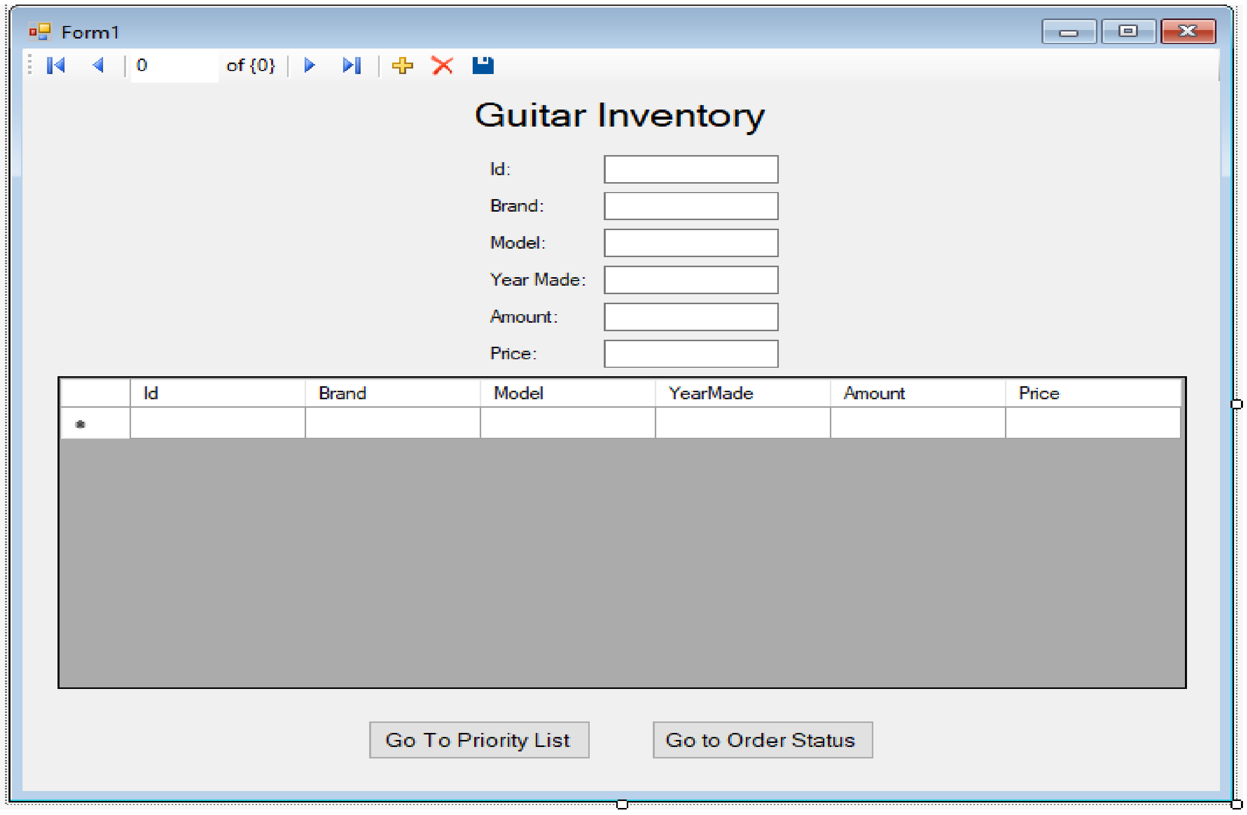
	OrderId	OrderDate	Brand	Model
*				



Priority List

Cust Id:	
Cust First Name:	
Cust Last Name:	
Item Ordered:	

	CustId	CustFirstName	CustLastName	ItemOrdered
*				



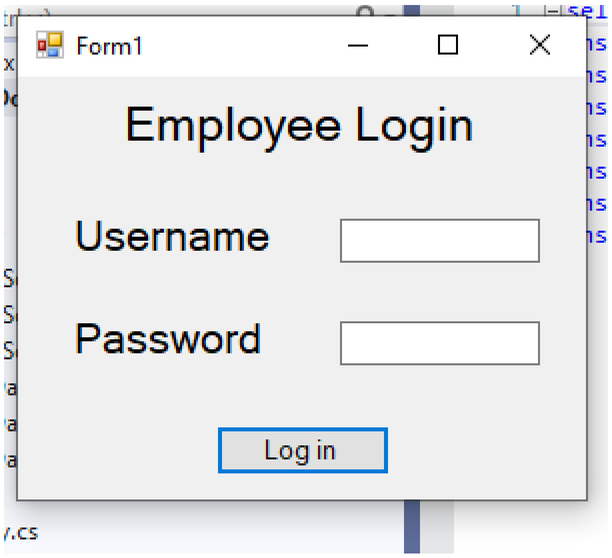
Guitar Inventory

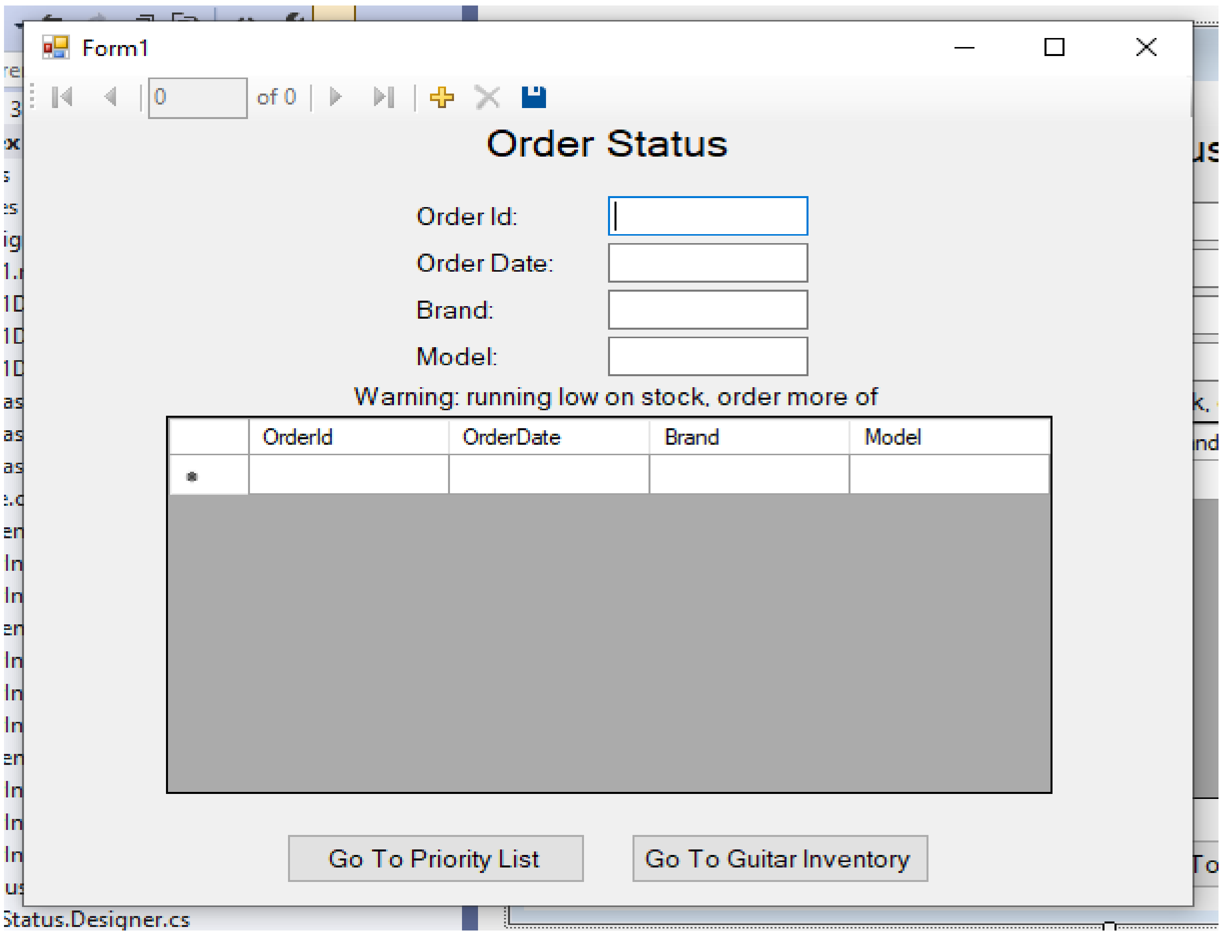
Id:	
Brand:	
Model:	
Year Made:	
Amount:	
Price:	

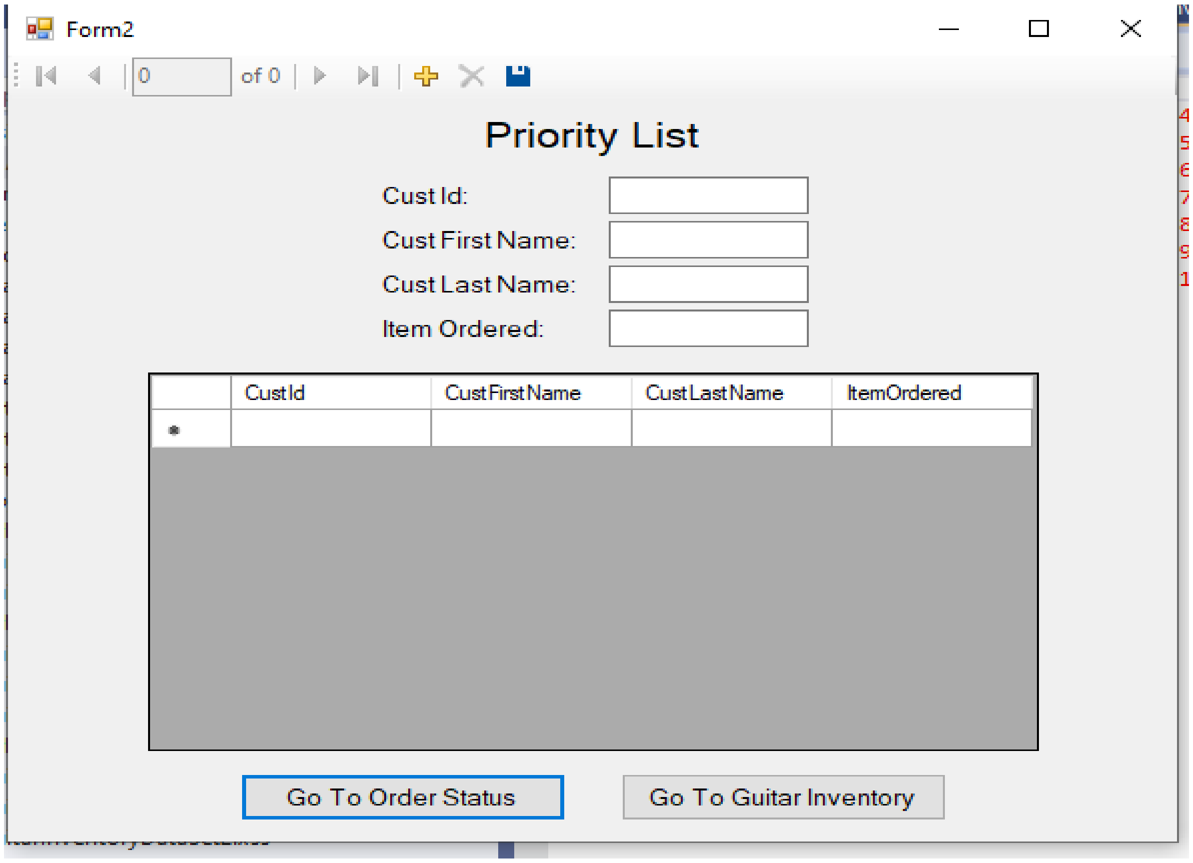
	Id	Brand	Model	YearMade	Amount	Price
*						

After you log in and get into the employee side. You will the Order status page pop up, and on the bottom of that are two buttons. From there, you can click on the two buttons down on the bottom that say “Priority List” and “Guitar Inventory”. The Guitar Inventory button takes you to the Guitar Inventory Page and the Priority List takes you to the Priority List. When you go to either of those pages, the format is the same and you can add things to the Guitar Inventory or the Priority List. At the bottom of the two pages, there are also two buttons at the bottom that say “Priority List” and “Order Status” for the Guitar Inventory Page and “Order Status” and “Guitar Inventory” on the Priority List page. The Order Status, Guitar Inventory, and Priority List pages are functionally identical. You can move around to each different page and edit the tables on all of these pages.

Interfaces at design time



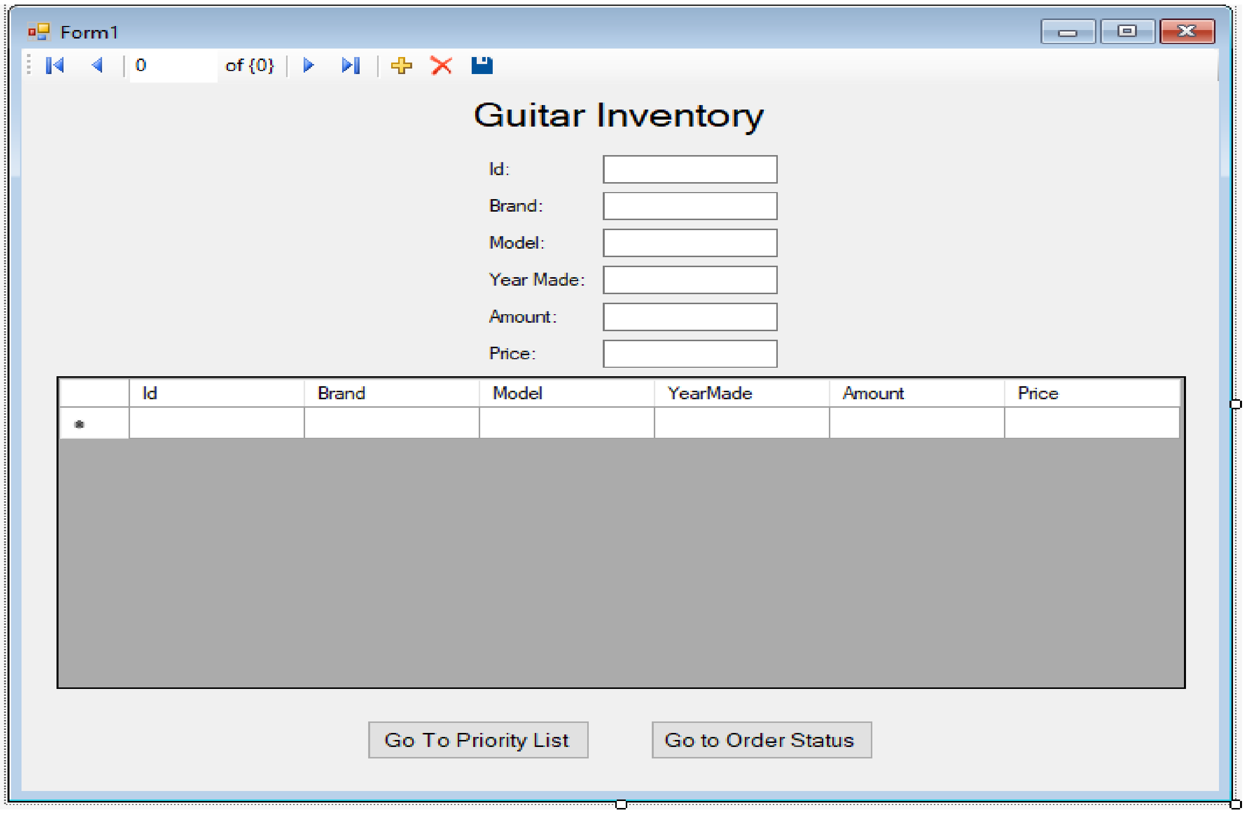




Priority List

Cust Id:	
Cust First Name:	
Cust Last Name:	
Item Ordered:	

	CustId	CustFirstName	CustLastName	ItemOrdered
*				

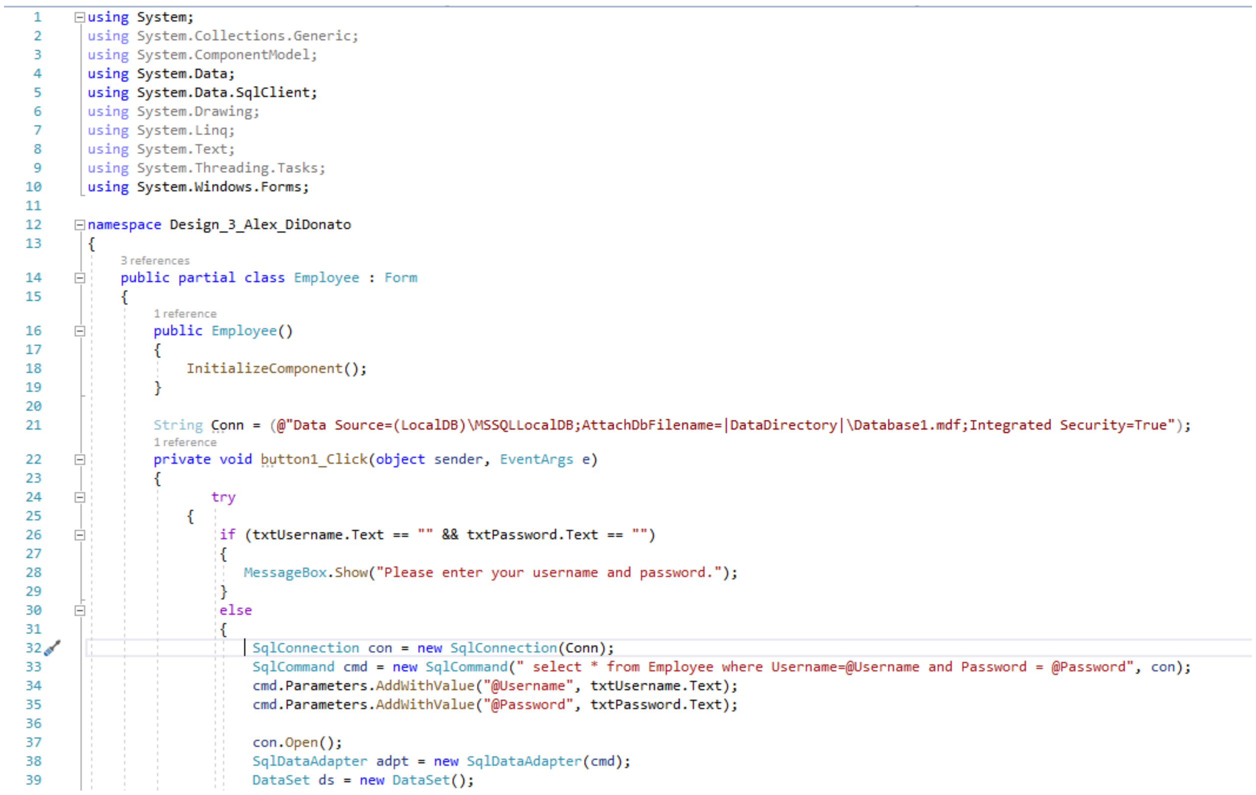


Guitar Inventory

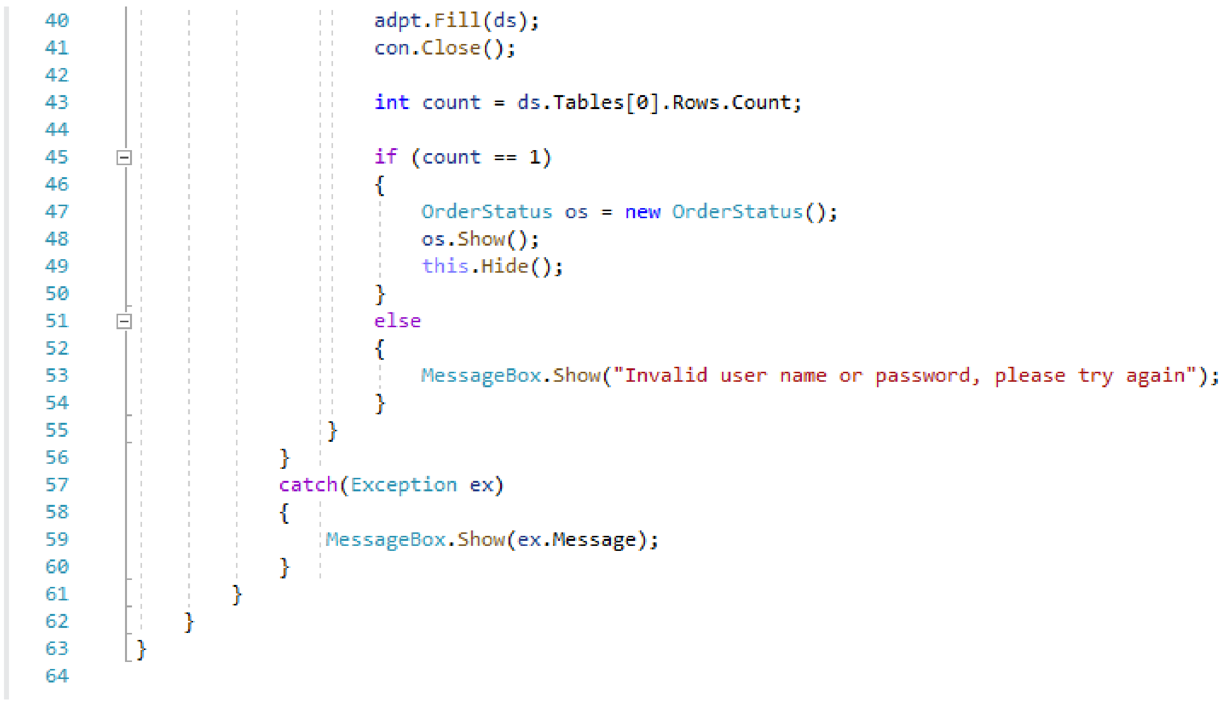
Id:	
Brand:	
Model:	
Year Made:	
Amount:	
Price:	

	Id	Brand	Model	YearMade	Amount	Price
*						

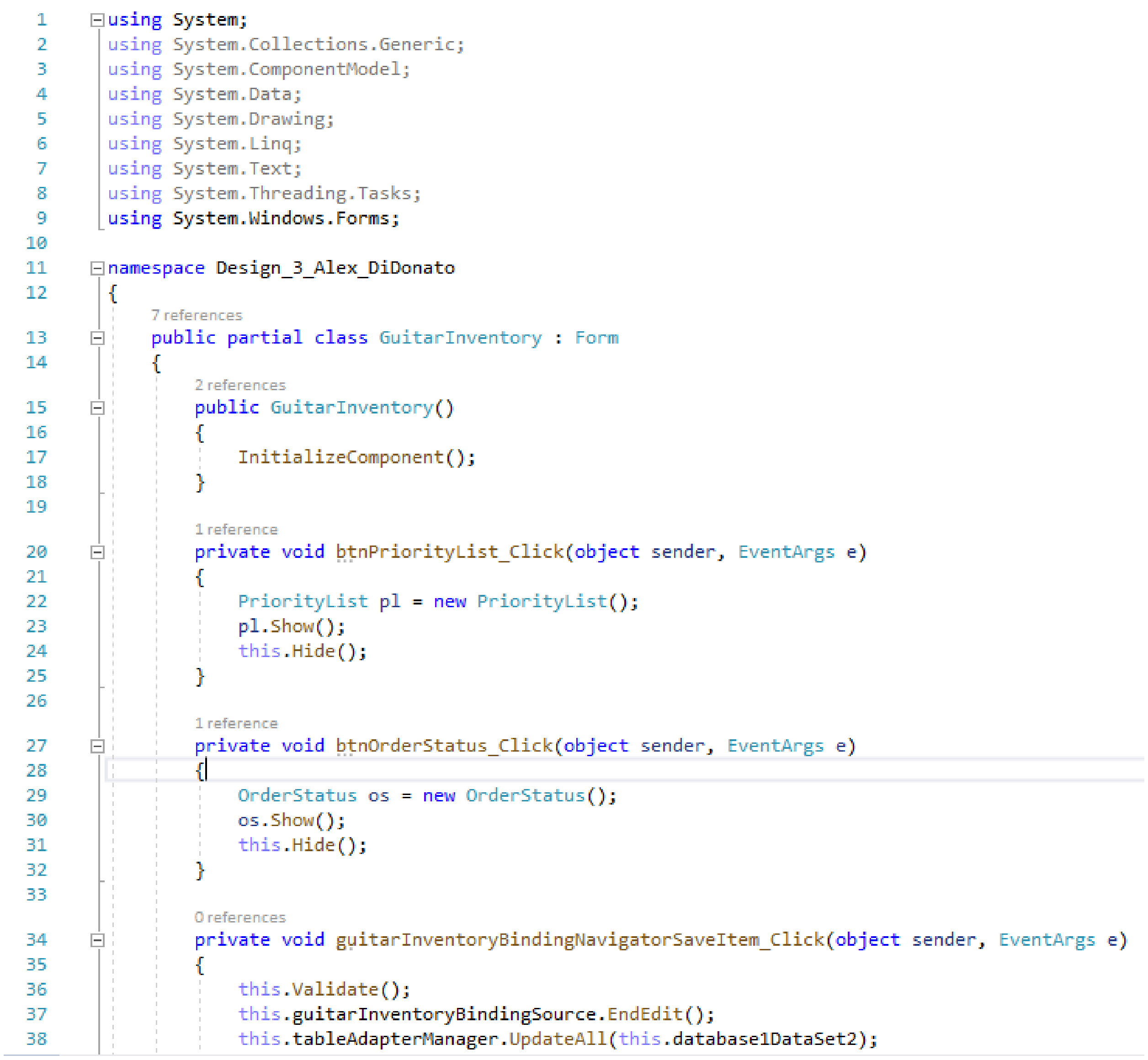
Source code



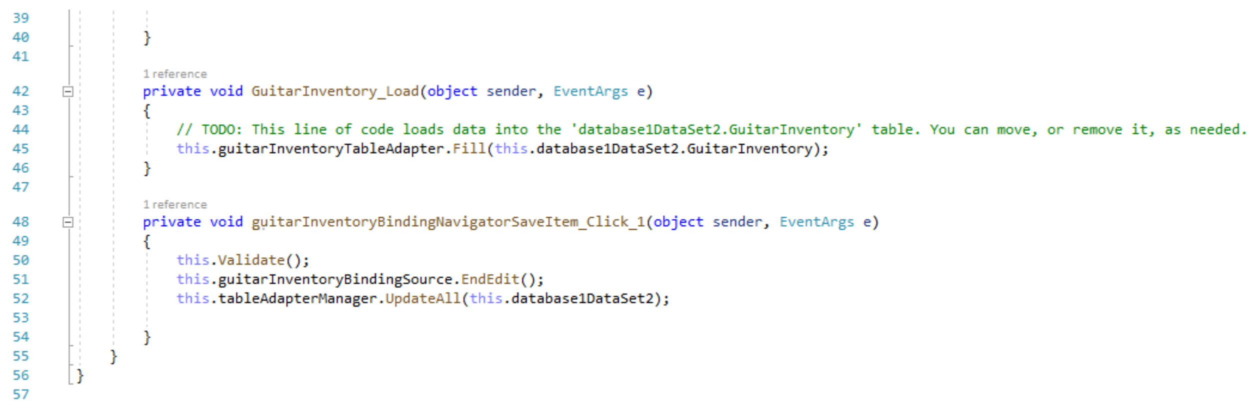
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Design_3_Alex_DiDonato
{
    public partial class Employee : Form
    {
        public Employee()
        {
            InitializeComponent();
        }

        String Conn = (@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\Database1.mdf;Integrated Security=True");
        private void button1_Click(object sender, EventArgs e)
        {
                try
            {
                if (txtUsername.Text == "" && txtPassword.Text == "")
                {
                    MessageBox.Show("Please enter your username and password.");
                }
                else
                {
                    SqlConnection con = new SqlConnection(Conn);
                    SqlCommand cmd = new SqlCommand(" select * from Employee where Username=@Username and Password = @Password", con);
                    cmd.Parameters.AddWithValue("@Username", txtUsername.Text);
                    cmd.Parameters.AddWithValue("@Password", txtPassword.Text);

                    con.Open();
                    SqlDataAdapter adpt = new SqlDataAdapter(cmd);
                    DataSet ds = new DataSet();
```



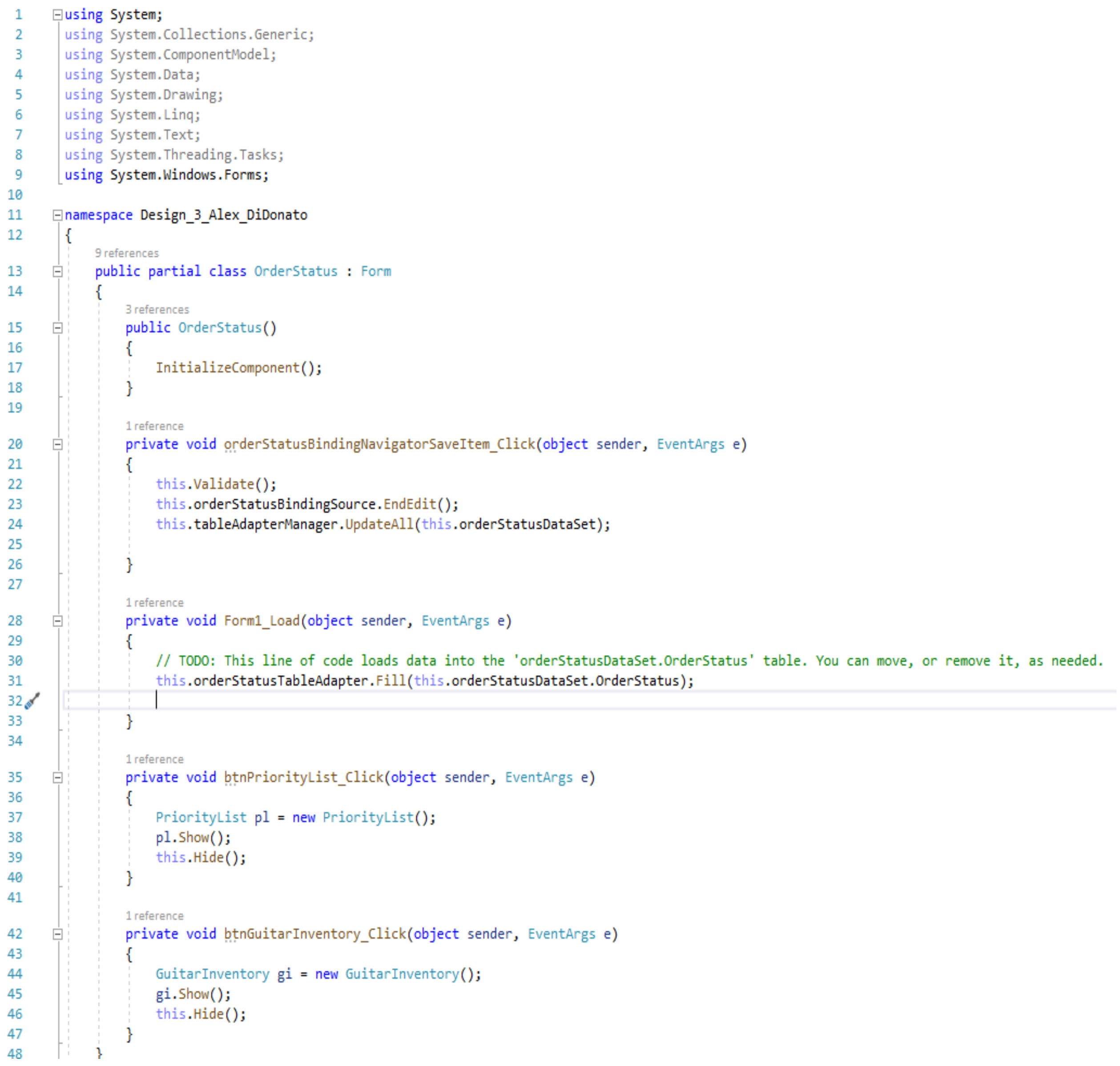
```
                    adpt.Fill(ds);
                    con.Close();

                    int count = ds.Tables[0].Rows.Count;

                    if (count == 1)
                    {
                        OrderStatus os = new OrderStatus();
                        os.Show();
                        this.Hide();
                    }
                    else
                    {
                        MessageBox.Show("Invalid user name or password, please try again");
                    }
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
```



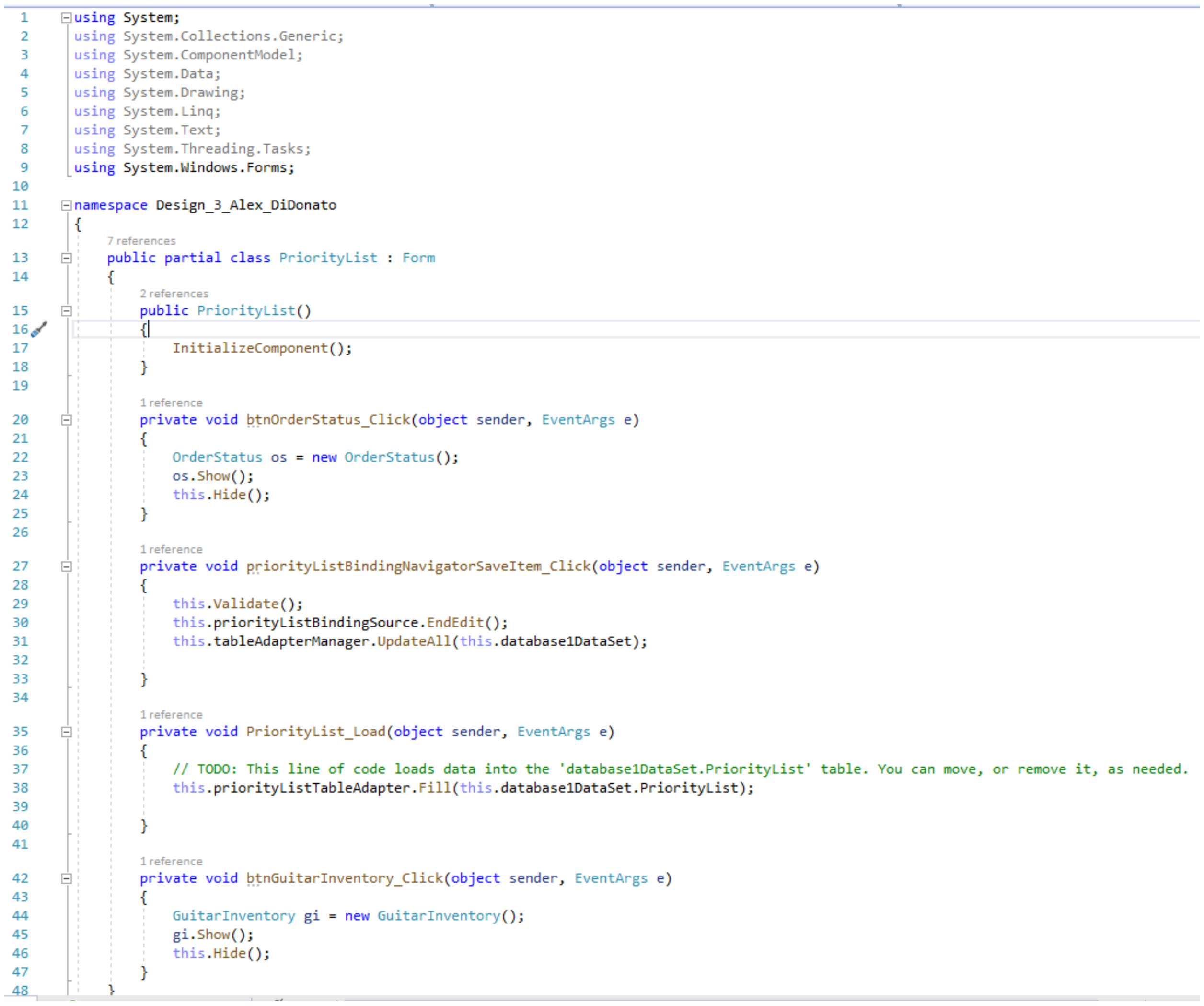
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Design_3_Alex_DiDonato
{
    7 references
    public partial class GuitarInventory : Form
    {
        2 references
        public GuitarInventory()
        {
            InitializeComponent();
        }

        1 reference
        private void btnPriorityList_Click(object sender, EventArgs e)
        {
            PriorityList pl = new PriorityList();
            pl.Show();
            this.Hide();
        }

        1 reference
        private void btnOrderStatus_Click(object sender, EventArgs e)
        {
            OrderStatus os = new OrderStatus();
            os.Show();
            this.Hide();
        }

        0 references
        private void guitarInventoryBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.guitarInventoryBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.database1DataSet2);
```

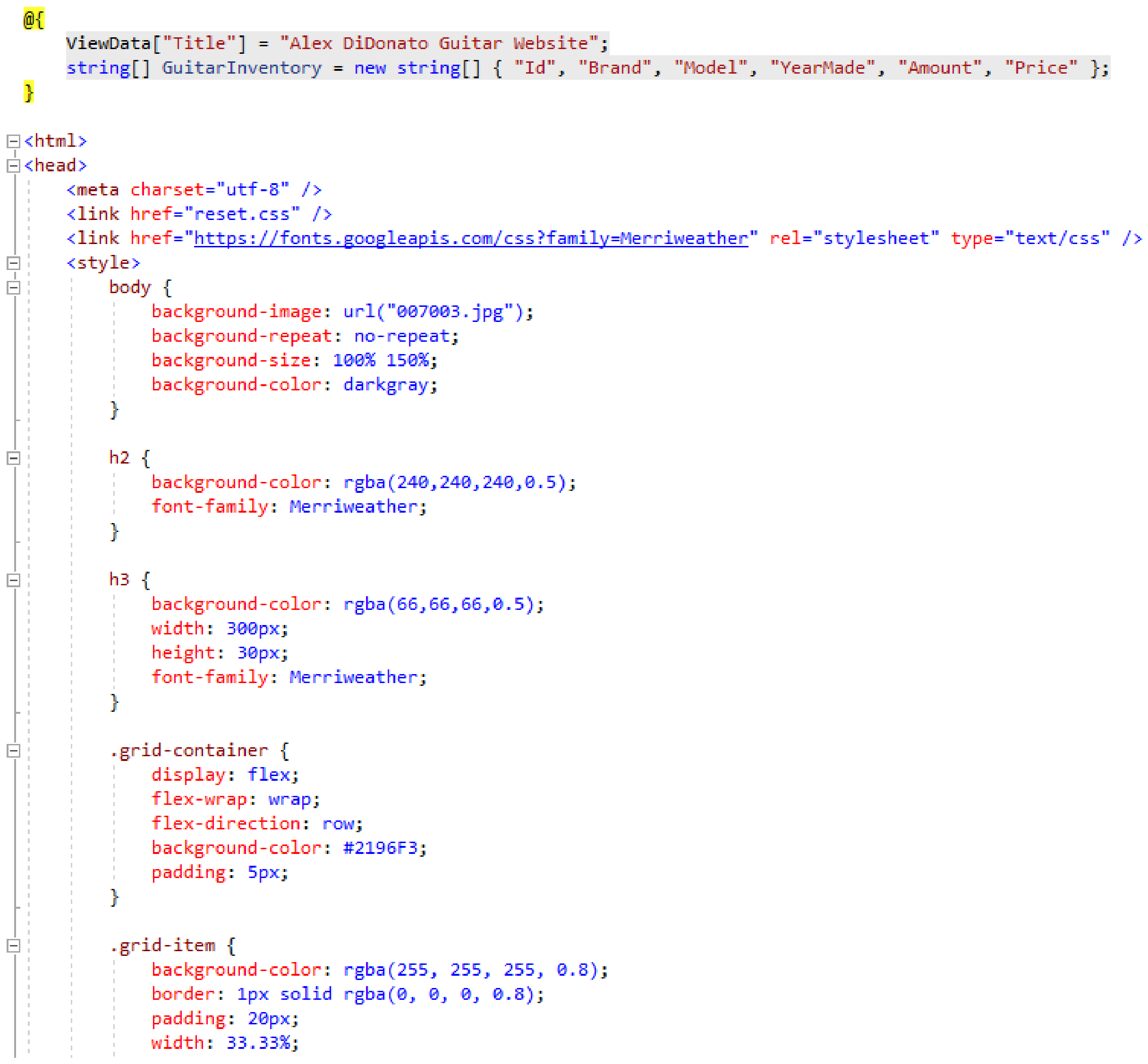


```

        }

        1 reference
        private void GuitarInventory_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'database1DataSet2.GuitarInventory' table. You can move, or remove it, as needed.
            this.guitarInventoryTableAdapter.Fill(this.database1DataSet2.GuitarInventory);
        }

        1 reference
        private void guitarInventoryBindingNavigatorSaveItem_Click_1(object sender, EventArgs e)
        {
            this.Validate();
            this.guitarInventoryBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.database1DataSet2);

        }
    }
}
```

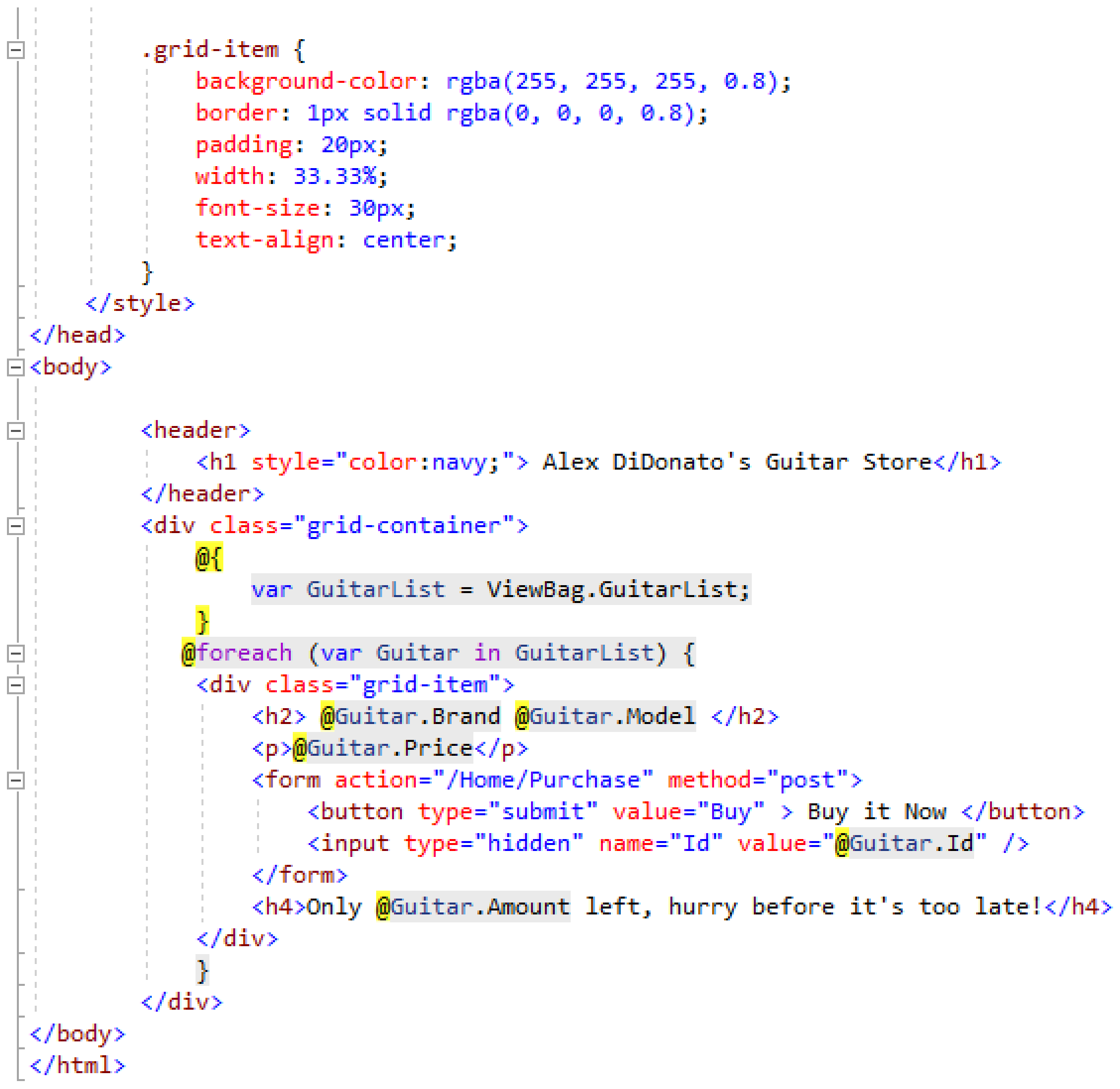


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Design_3_Alex_DiDonato
{
    9 references
    public partial class OrderStatus : Form
    {
        3 references
        public OrderStatus()
        {
            InitializeComponent();
        }

        1 reference
        private void orderStatusBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.orderStatusBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.orderStatusDataSet);

        }

        1 reference
        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'orderStatusDataSet.OrderStatus' table. You can move, or remove it, as needed.
            this.orderStatusTableAdapter.Fill(this.orderStatusDataSet.OrderStatus);

        }

        1 reference
        private void btnPriorityList_Click(object sender, EventArgs e)
        {
            PriorityList pl = new PriorityList();
            pl.Show();
            this.Hide();
        }

        1 reference
        private void btnGuitarInventory_Click(object sender, EventArgs e)
        {
            GuitarInventory gi = new GuitarInventory();
            gi.Show();
            this.Hide();
        }
    }
```


```
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 Design_3_Alex_DiDonato
{
    7 references
    public partial class PriorityList : Form
    {
        2 references
        public PriorityList()
        {
            InitializeComponent();
        }

        1 reference
        private void btnOrderStatus_Click(object sender, EventArgs e)
        {
            OrderStatus os = new OrderStatus();
            os.Show();
            this.Hide();
        }

        1 reference
        private void priorityListBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.priorityListBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.database1DataSet);

        }

        1 reference
        private void PriorityList_Load(object sender, EventArgs e)
        {
            // TODO: This line of code loads data into the 'database1DataSet.PriorityList' table. You can move, or remove it, as needed.
            this.priorityListTableAdapter.Fill(this.database1DataSet.PriorityList);

        }

        1 reference
        private void btnGuitarInventory_Click(object sender, EventArgs e)
        {
            GuitarInventory gi = new GuitarInventory();
            gi.Show();
            this.Hide();
        }
    }
```

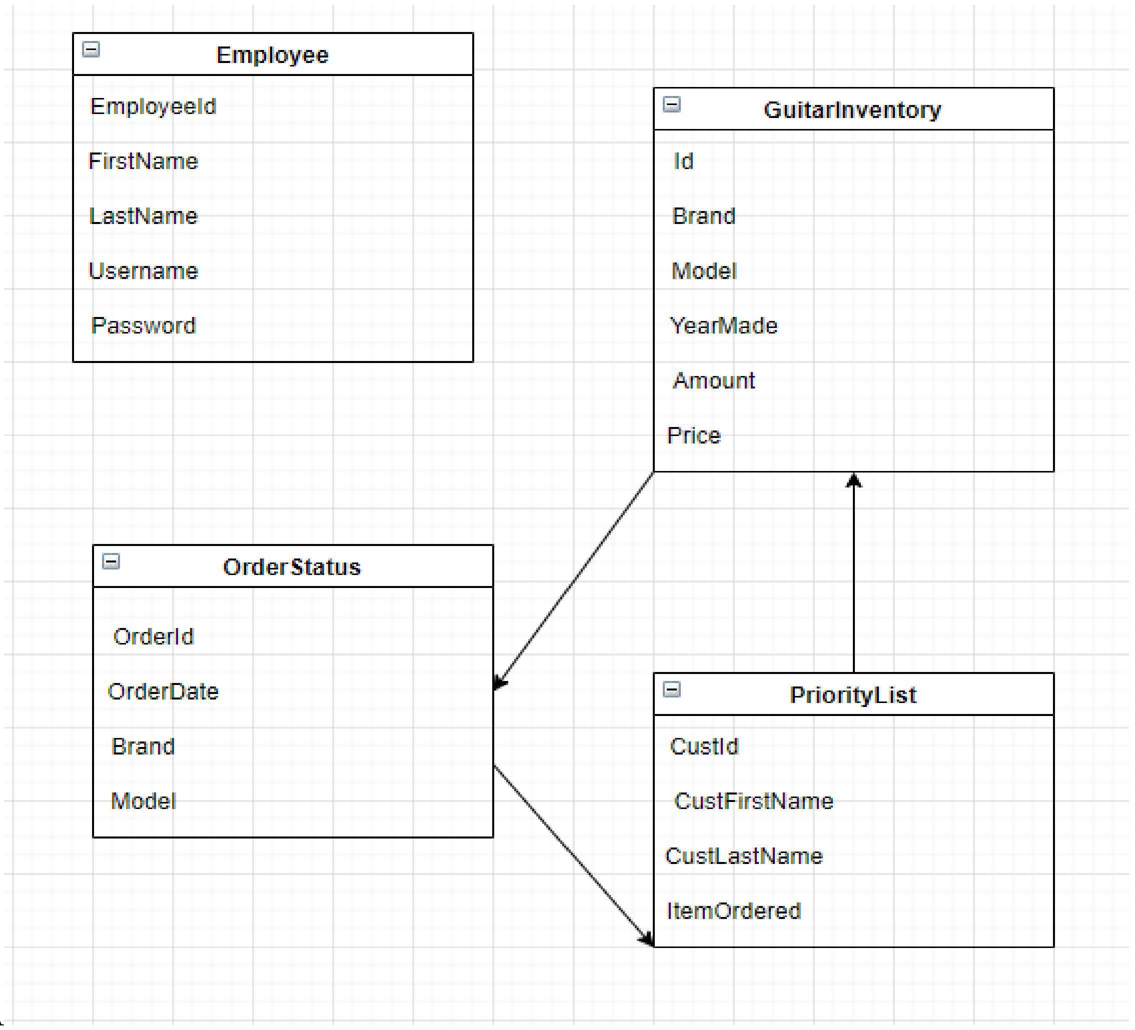


```
@{
    ViewData["Title"] = "Alex DiDonato Guitar Website";
    string[] GuitarInventory = new string[] { "Id", "Brand", "Model", "YearMade", "Amount", "Price" };
}

<html>
<head>
    <meta charset="utf-8" />
    <link href="reset.css" />
    <link href="https://fonts.googleapis.com/css?family=Merriweather" rel="stylesheet" type="text/css" />
    <style>
        body {
            background-image: url("007003.jpg");
            background-repeat: no-repeat;
            background-size: 100% 150%;
            background-color: darkgray;
        }

        h2 {
            background-color: rgba(240,240,240,0.5);
            font-family: Merriweather;
        }

        h3 {
            background-color: rgba(66,66,66,0.5);
            width: 300px;
            height: 30px;
            font-family: Merriweather;
        }

        .grid-container {
            display: flex;
            flex-wrap: wrap;
            flex-direction: row;
            background-color: #2196F3;
            padding: 5px;
        }

        .grid-item {
            background-color: rgba(255, 255, 255, 0.8);
            border: 1px solid rgba(0, 0, 0, 0.8);
            padding: 20px;
            width: 33.33%;
```

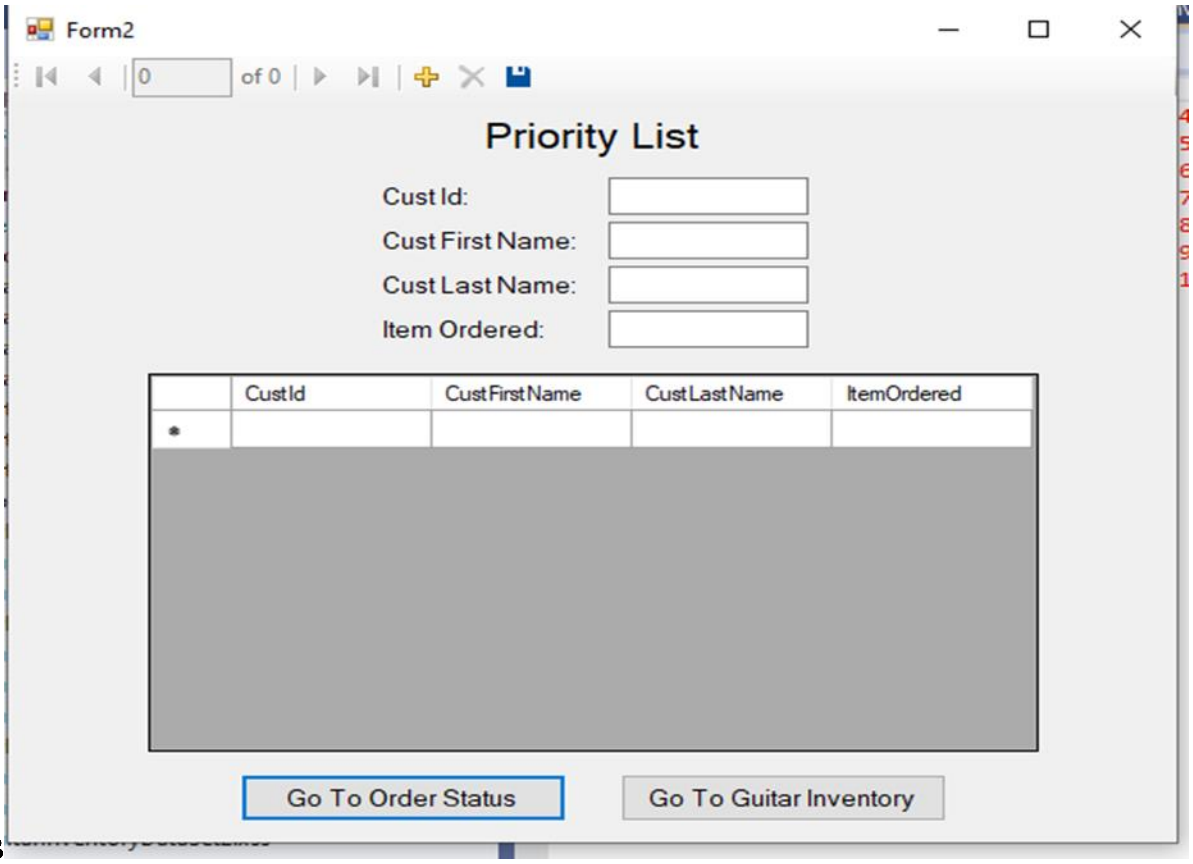


```
        .grid-item {
            background-color: rgba(255, 255, 255, 0.8);
            border: 1px solid rgba(0, 0, 0, 0.8);
            padding: 20px;
            width: 33.33%;
            font-size: 30px;
            text-align: center;
        }
    </style>
</head>
<body>

        <header>
            <h1 style="color:navy;"> Alex DiDonato's Guitar Store</h1>
        </header>
        <div class="grid-container">
            @{
                var GuitarList = ViewBag.GuitarList;
            }
           @foreach (var Guitar in GuitarList) {
            <div class="grid-item">
                <h2> @Guitar.Brand @Guitar.Model </h2>
                <p>@Guitar.Price</p>
                <form action="/Home/Purchase" method="post">
                    <button type="submit" value="Buy" > Buy it Now </button>
                    <input type="hidden" name="Id" value="@Guitar.Id" />
                </form>
                <h4>Only @Guitar.Amount left, hurry before it's too late!</h4>
            </div>
            }
        </div>
</body>
</html>
```

Database Documentation



A



Form2

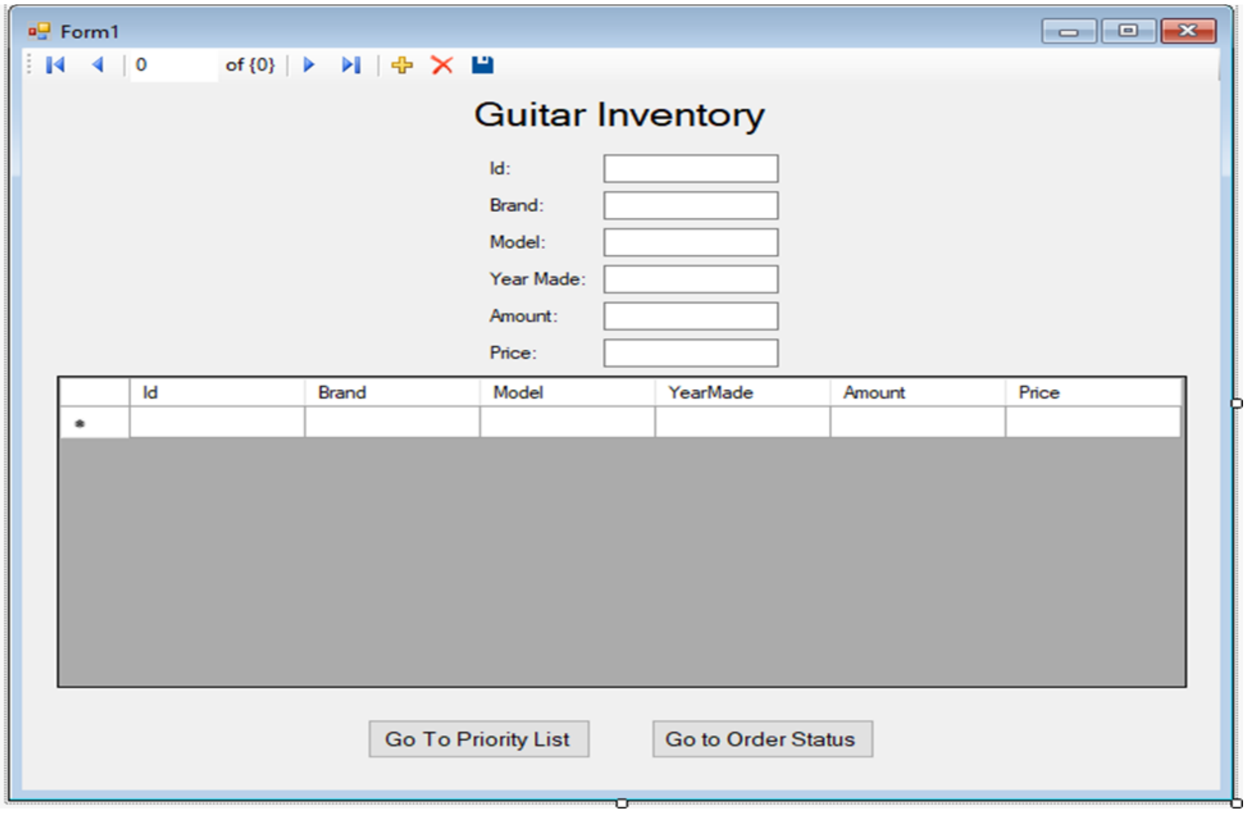
of 0

Priority List

Cust Id:	
Cust First Name:	
Cust Last Name:	
Item Ordered:	

	CustId	CustFirstName	CustLastName	ItemOrdered
*				

B



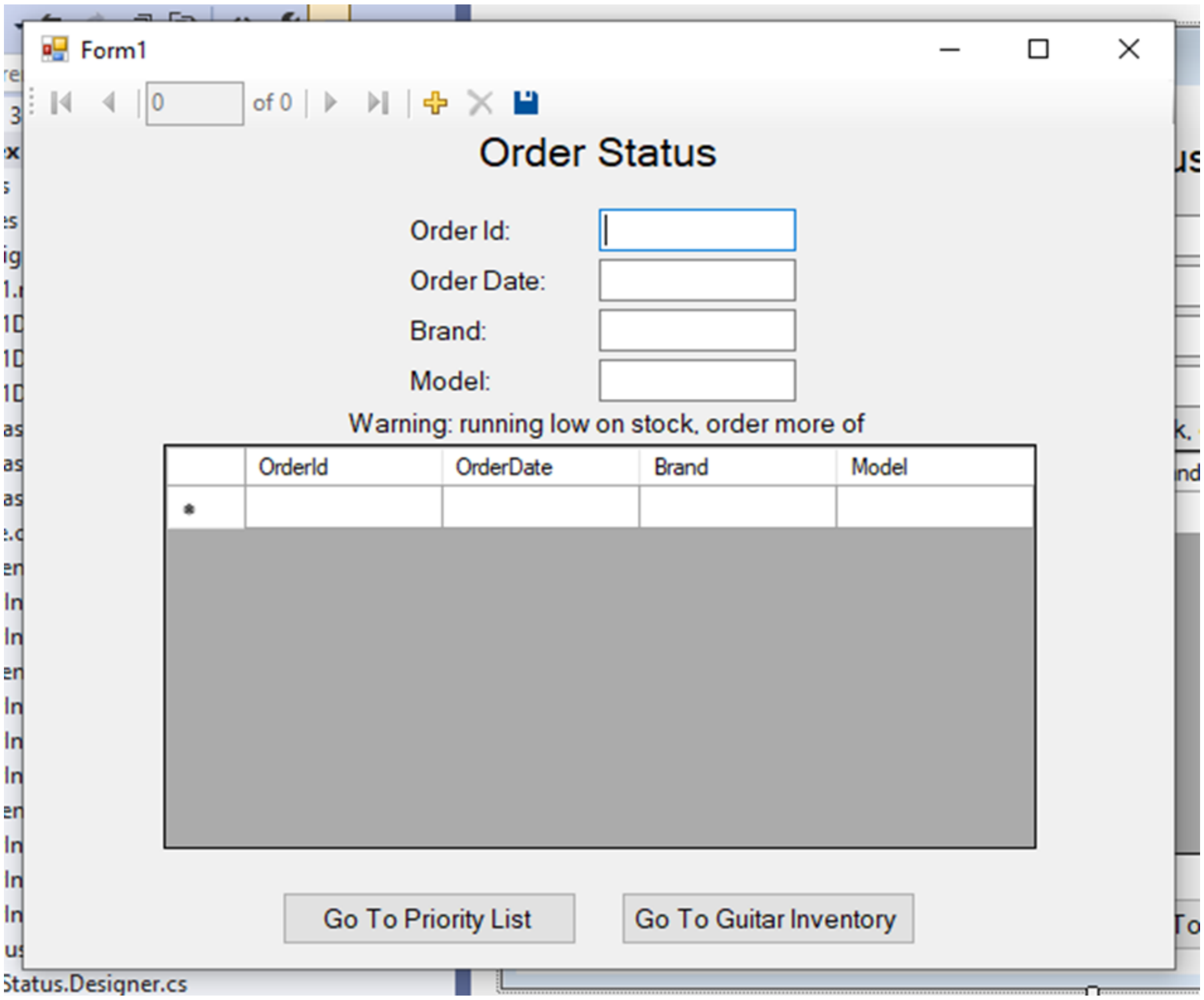
Form1

of {0}

Guitar Inventory

Id:	
Brand:	
Model:	
Year Made:	
Amount:	
Price:	

	Id	Brand	Model	YearMade	Amount	Price
*						

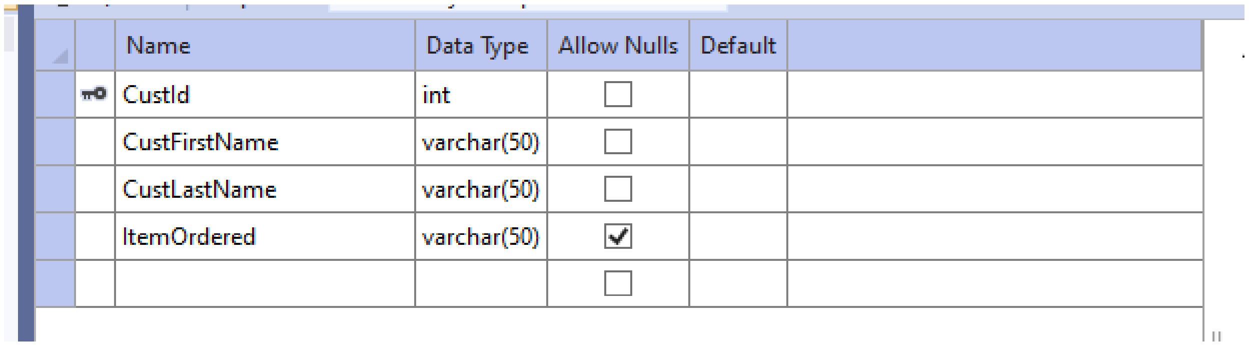


Order Status

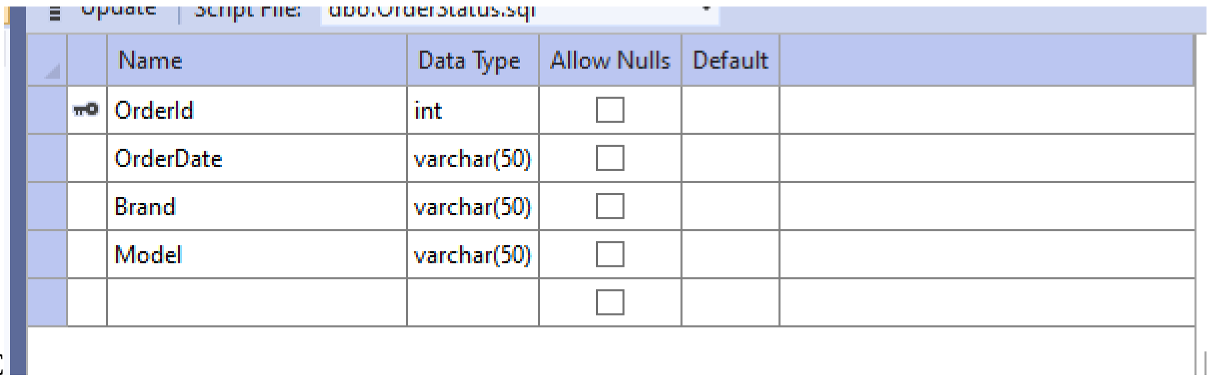
Order Id:	
Order Date:	
Brand:	
Model:	

Warning: running low on stock, order more of

	OrderId	OrderDate	Brand	Model
*				



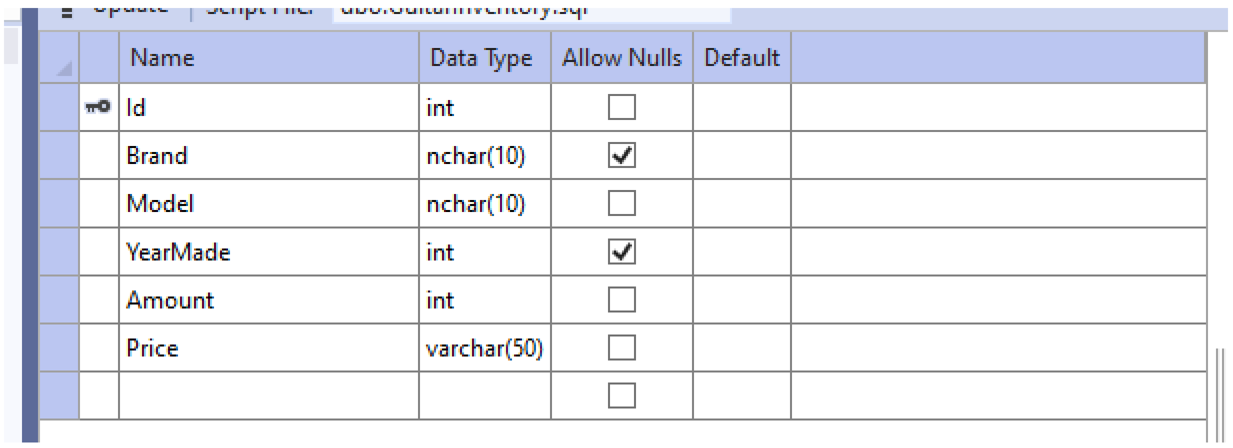
	Name	Data Type	Allow Nulls	Default
🔑	CustId	int	☐	
	CustFirstName	varchar(50)	☐	
	CustLastName	varchar(50)	☐	
	ItemOrdered	varchar(50)	✓	
			☐	



Update | Script File: dbo.OrderStatus.sql

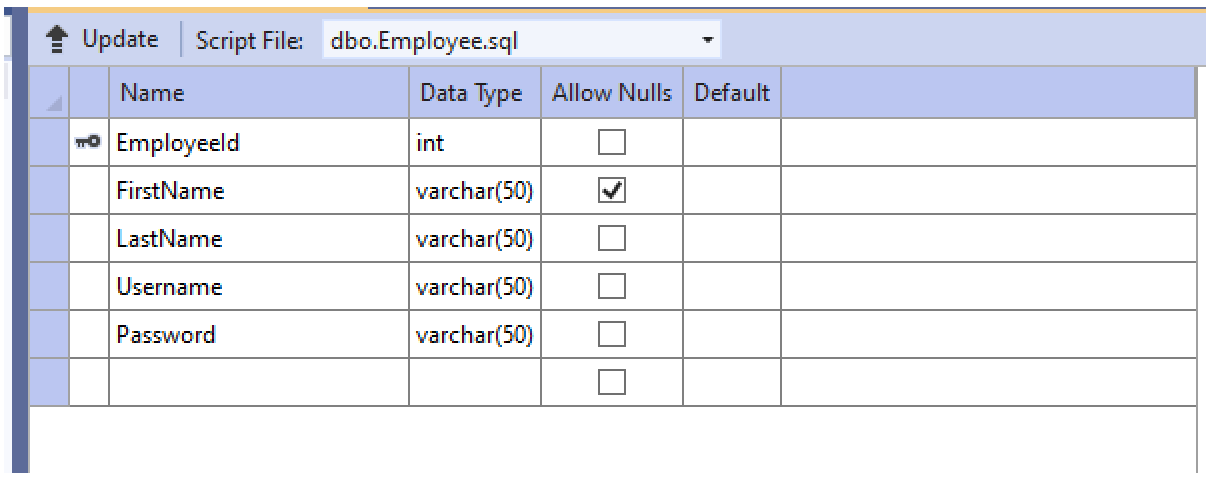
Name	Data Type	Allow Nulls	Default
OrderId	int	☐	
OrderDate	varchar(50)	☐	
Brand	varchar(50)	☐	
Model	varchar(50)	☐	
		☐	

C



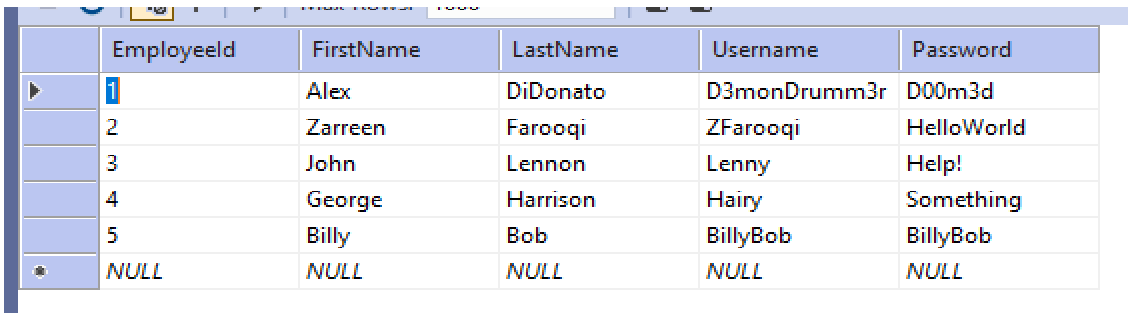
Update | Script File: dbo.GuitarInventory.sql

Name	Data Type	Allow Nulls	Default
Id	int	☐	
Brand	nchar(10)	✓	
Model	nchar(10)	☐	
YearMade	int	✓	
Amount	int	☐	
Price	varchar(50)	☐	
		☐	



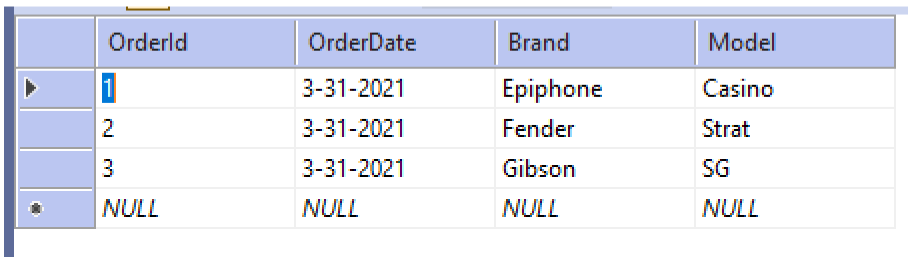
Update | Script File: dbo.Employee.sql

Name	Data Type	Allow Nulls	Default
EmployeeId	int	☐	
FirstName	varchar(50)	✓	
LastName	varchar(50)	☐	
Username	varchar(50)	☐	
Password	varchar(50)	☐	
		☐	

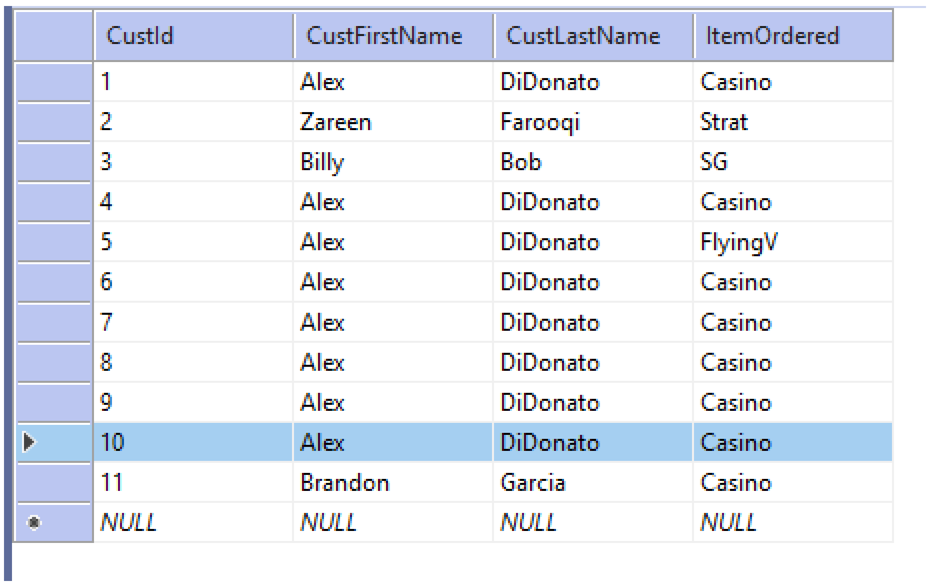


Max Rows: 1000

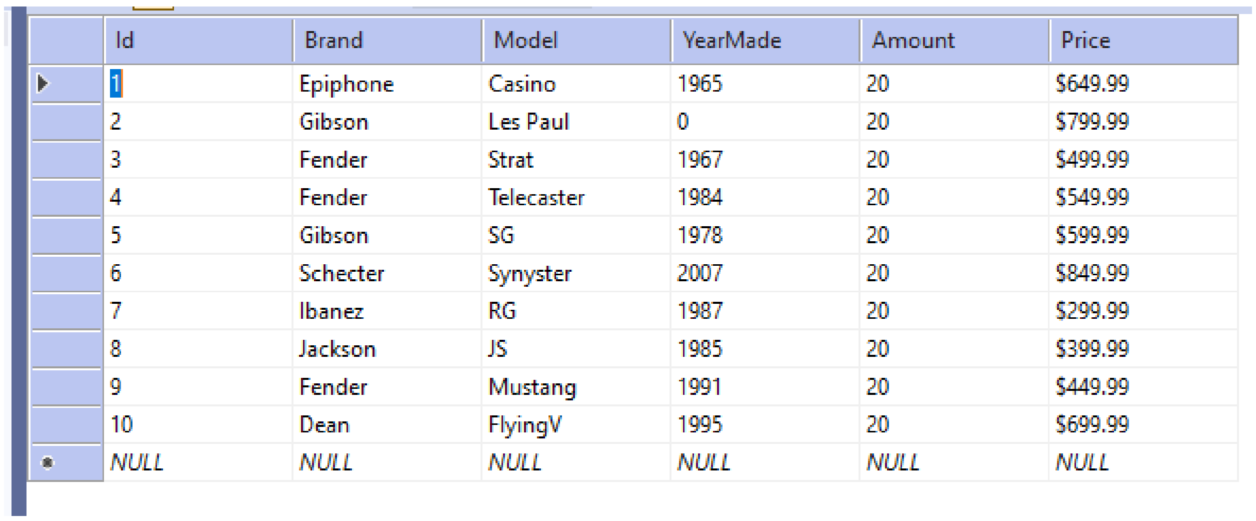
EmployeeId	FirstName	LastName	Username	Password
1	Alex	DiDonato	D3monDrumm3r	D00m3d
2	Zarreen	Farooqi	ZFarooqi	HelloWorld
3	John	Lennon	Lenny	Help!
4	George	Harrison	Hairy	Something
5	Billy	Bob	BillyBob	BillyBob
NULL	*NULL*	*NULL*	*NULL*	*NULL*



	OrderId	OrderDate	Brand	Model
▶	1	3-31-2021	Epiphone	Casino
	2	3-31-2021	Fender	Strat
	3	3-31-2021	Gibson	SG
*	*NULL*	*NULL*	*NULL*	*NULL*

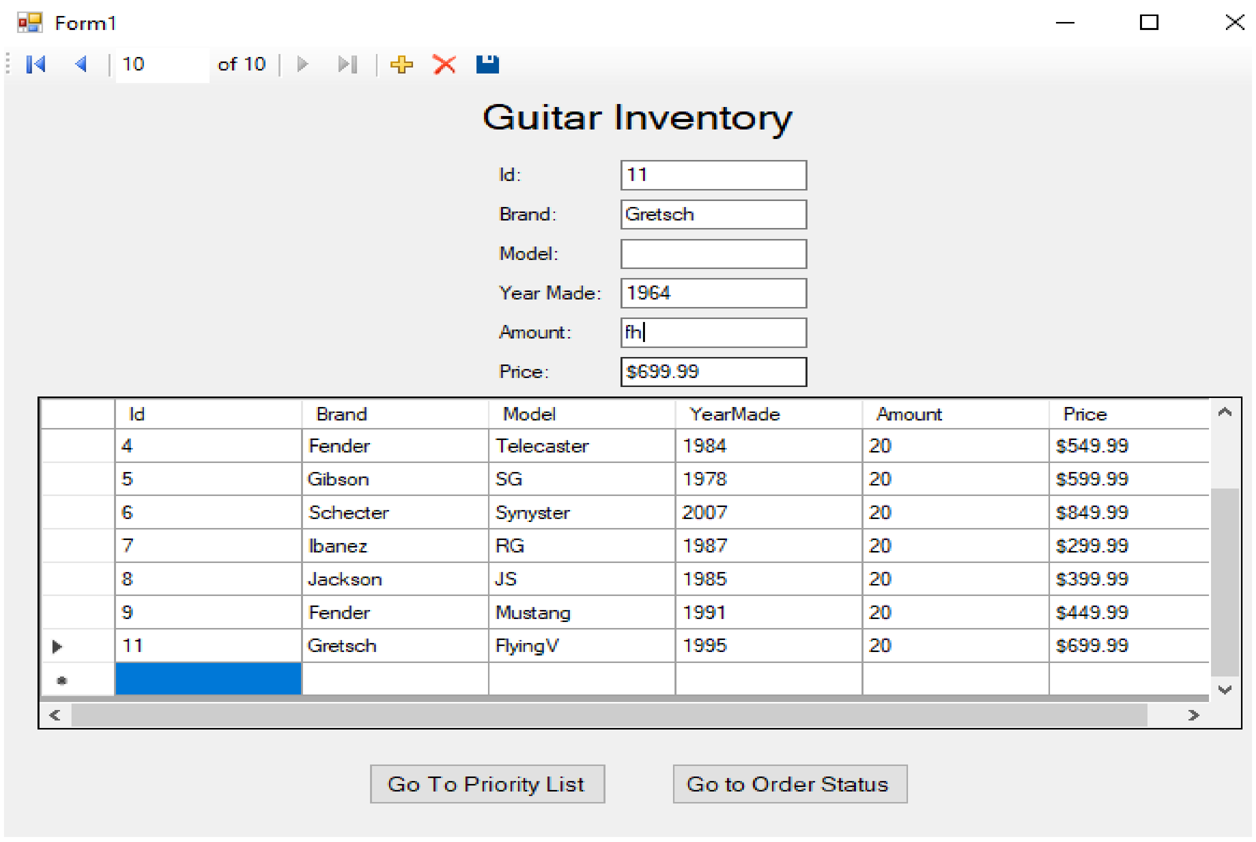


	CustId	CustFirstName	CustLastName	ItemOrdered
	1	Alex	DiDonato	Casino
	2	Zareen	Farooqi	Strat
	3	Billy	Bob	SG
	4	Alex	DiDonato	Casino
	5	Alex	DiDonato	FlyingV
	6	Alex	DiDonato	Casino
	7	Alex	DiDonato	Casino
	8	Alex	DiDonato	Casino
	9	Alex	DiDonato	Casino
▶	10	Alex	DiDonato	Casino
	11	Brandon	Garcia	Casino
*	*NULL*	*NULL*	*NULL*	*NULL*



	Id	Brand	Model	YearMade	Amount	Price
▶	1	Epiphone	Casino	1965	20	$649.99
	2	Gibson	Les Paul	0	20	$799.99
	3	Fender	Strat	1967	20	$499.99
	4	Fender	Telecaster	1984	20	$549.99
	5	Gibson	SG	1978	20	$599.99
	6	Schecter	Synyster	2007	20	$849.99
	7	Ibanez	RG	1987	20	$299.99
	8	Jackson	JS	1985	20	$399.99
	9	Fender	Mustang	1991	20	$449.99
	10	Dean	FlyingV	1995	20	$699.99
*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Invalid data



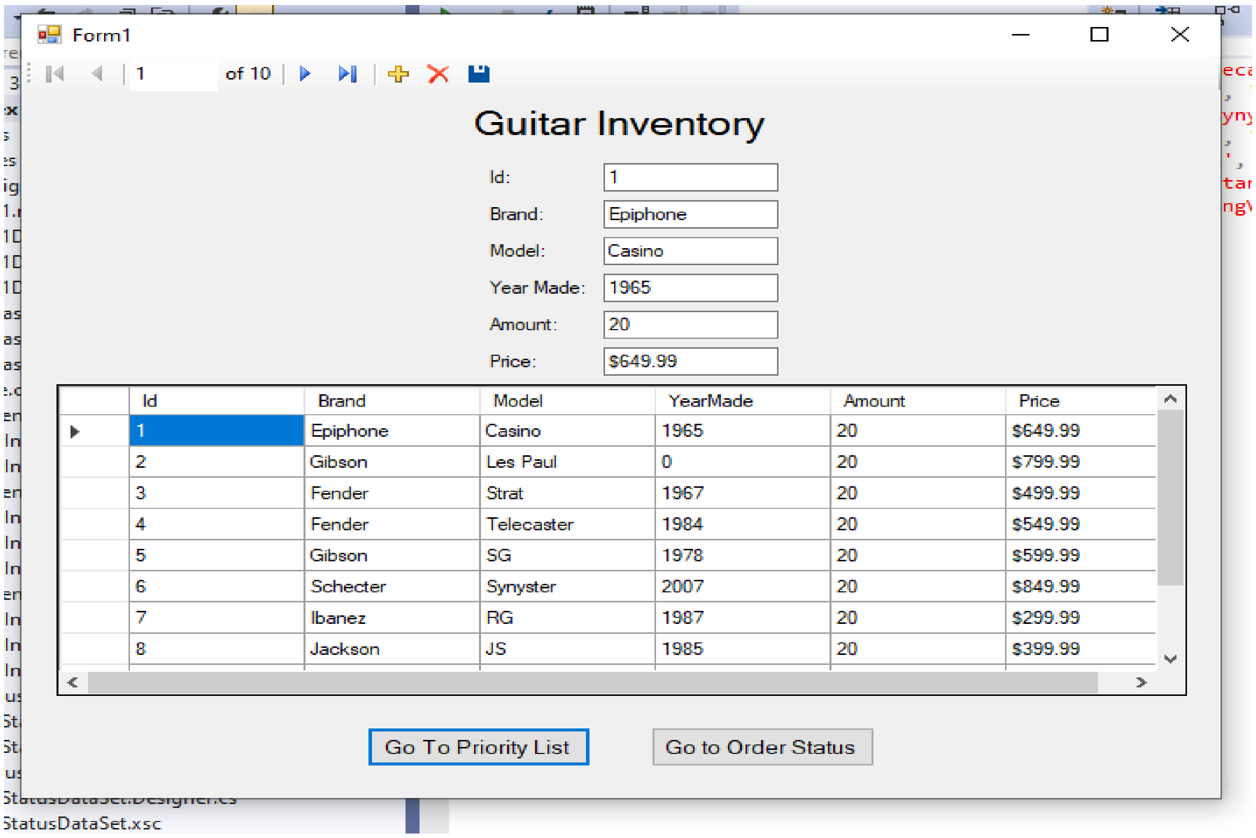
Guitar Inventory

Id:	11
Brand:	Gretsch
Model:	
Year Made:	1964
Amount:	fh
Price:	$699.99

	Id	Brand	Model	YearMade	Amount	Price
	4	Fender	Telecaster	1984	20	$549.99
	5	Gibson	SG	1978	20	$599.99
	6	Schecter	Synyster	2007	20	$849.99
	7	Ibanez	RG	1987	20	$299.99
	8	Jackson	JS	1985	20	$399.99
	9	Fender	Mustang	1991	20	$449.99
▶	11	Gretsch	FlyingV	1995	20	$699.99
*						

I tested putting in a string for the amount even though it says it must be an integer. When I put that in here, it would not let me move textboxes because the information in the textbox is not an integer and it would not move unless someone corrected it.

Valid data



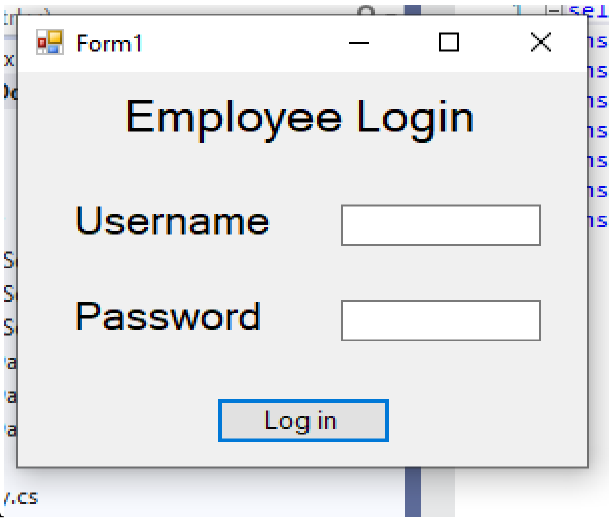
Guitar Inventory

Id:	1
Brand:	Epiphone
Model:	Casino
Year Made:	1965
Amount:	20
Price:	$649.99

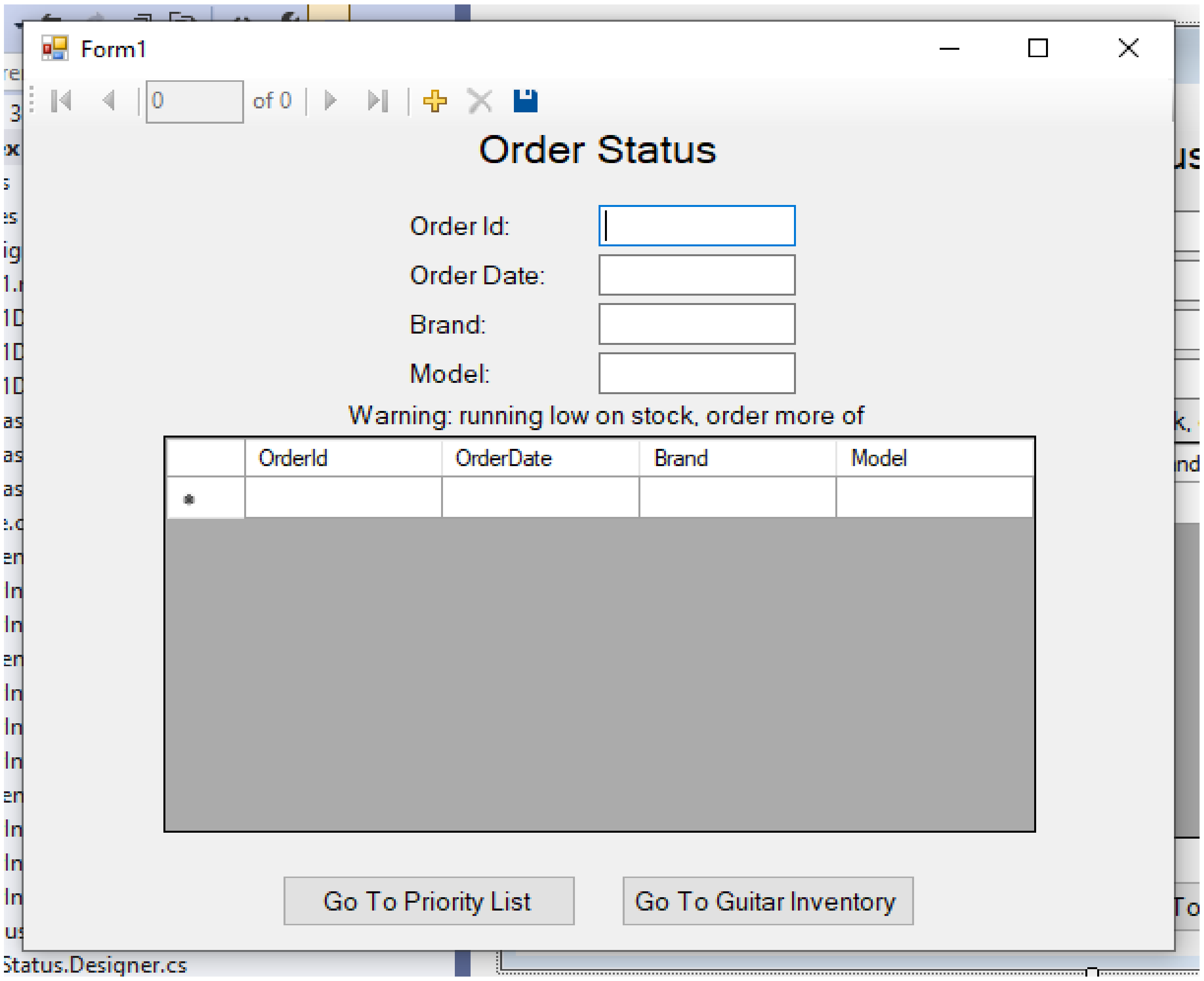
Id	Brand	Model	YearMade	Amount	Price
1	Epiphone	Casino	1965	20	$649.99
2	Gibson	Les Paul	0	20	$799.99
3	Fender	Strat	1967	20	$499.99
4	Fender	Telecaster	1984	20	$549.99
5	Gibson	SG	1978	20	$599.99
6	Schecter	Synyster	2007	20	$849.99
7	Ibanez	RG	1987	20	$299.99
8	Jackson	JS	1985	20	$399.99

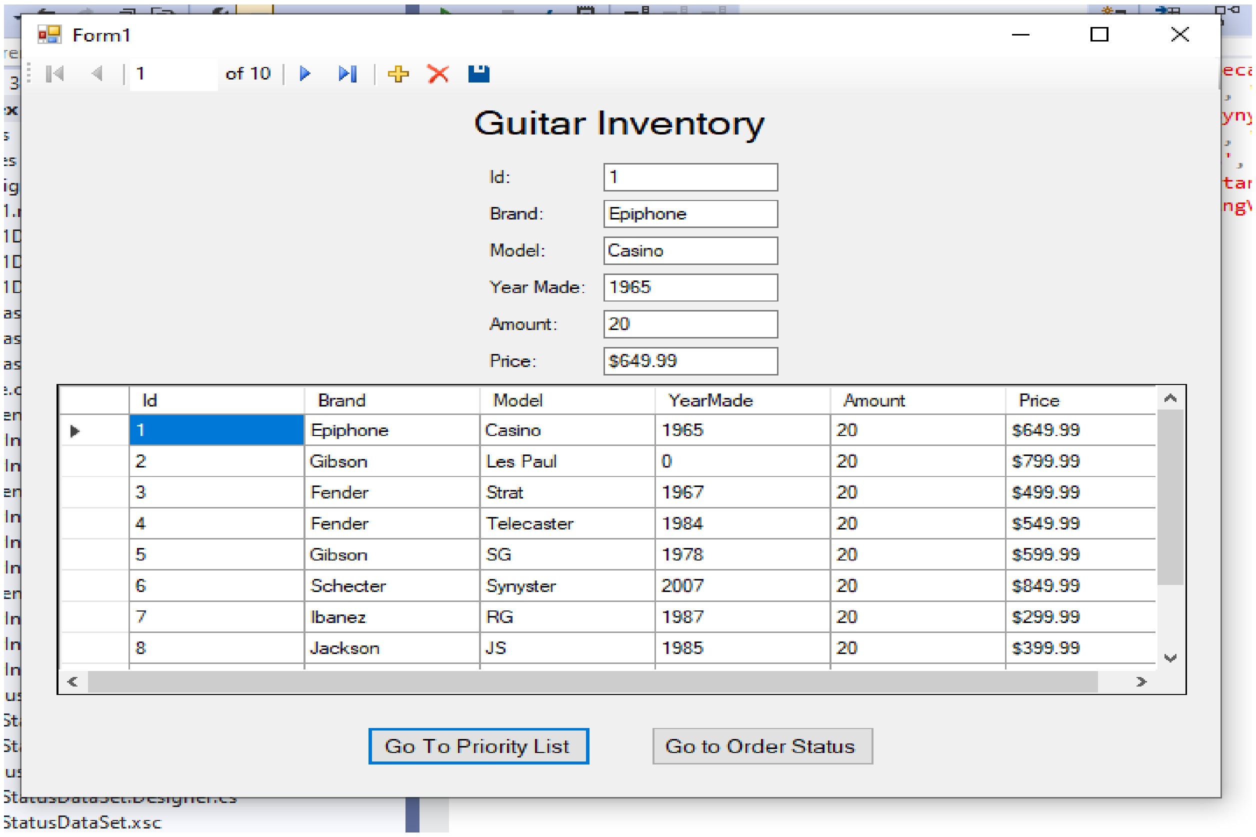
I tested this as valid data and this is how an entry should be entered into a database. I did this as main testing to see if the program would run okay.

Interfaces at runtime



A





Form1

1 of 10

Guitar Inventory

Id: 1

Brand: Epiphone

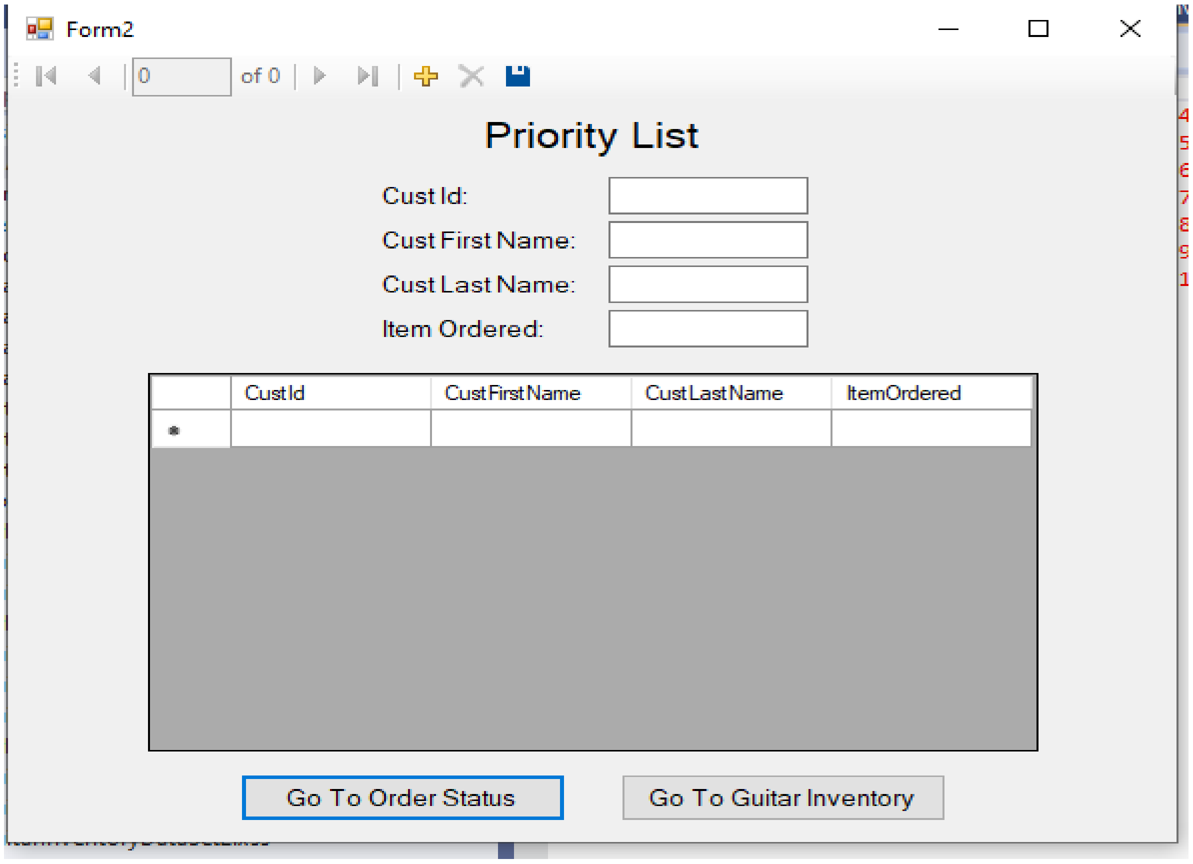
Model: Casino

Year Made: 1965

Amount: 20

Price: $649.99

Id	Brand	Model	YearMade	Amount	Price
1	Epiphone	Casino	1965	20	$649.99
2	Gibson	Les Paul	0	20	$799.99
3	Fender	Strat	1967	20	$499.99
4	Fender	Telecaster	1984	20	$549.99
5	Gibson	SG	1978	20	$599.99
6	Schecter	Synyster	2007	20	$849.99
7	Ibanez	RG	1987	20	$299.99
8	Jackson	JS	1985	20	$399.99



Form2

0 of 0

Priority List

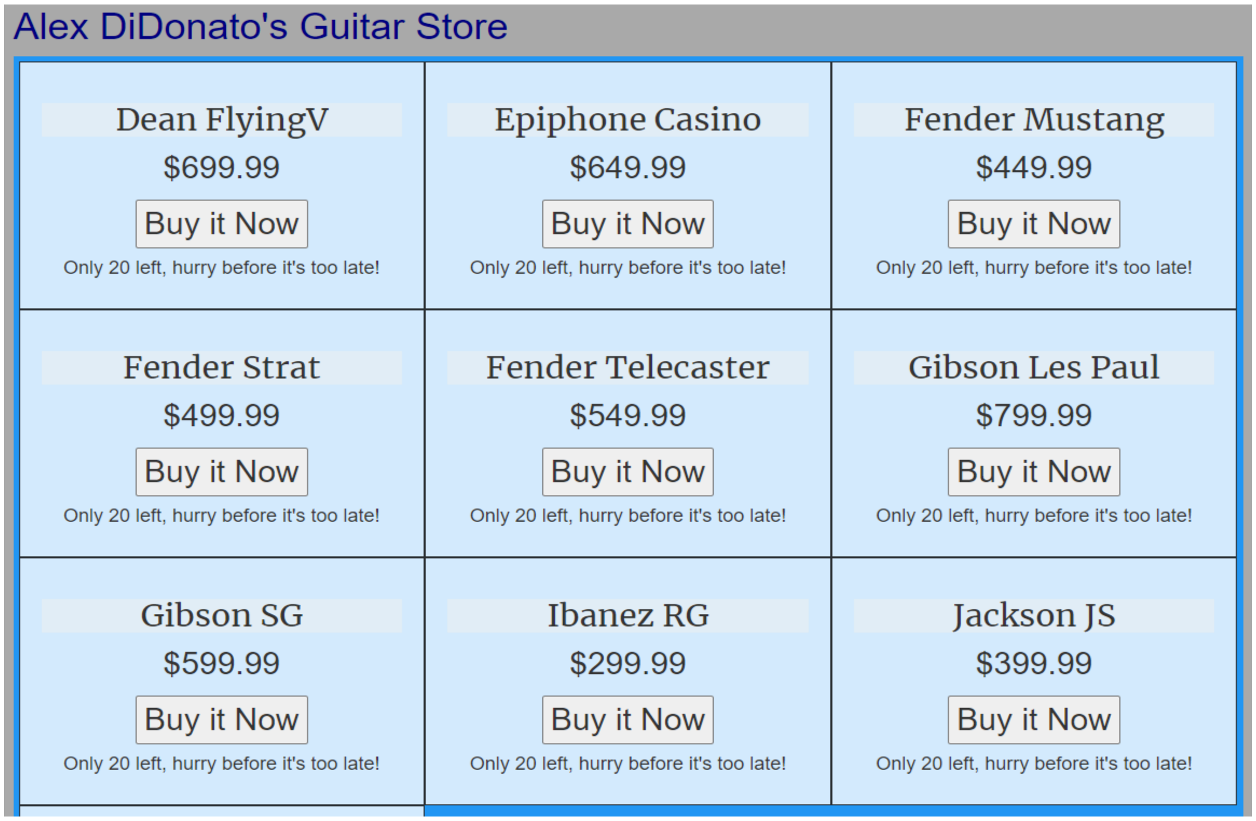
Cust Id:

Cust First Name:

Cust Last Name:

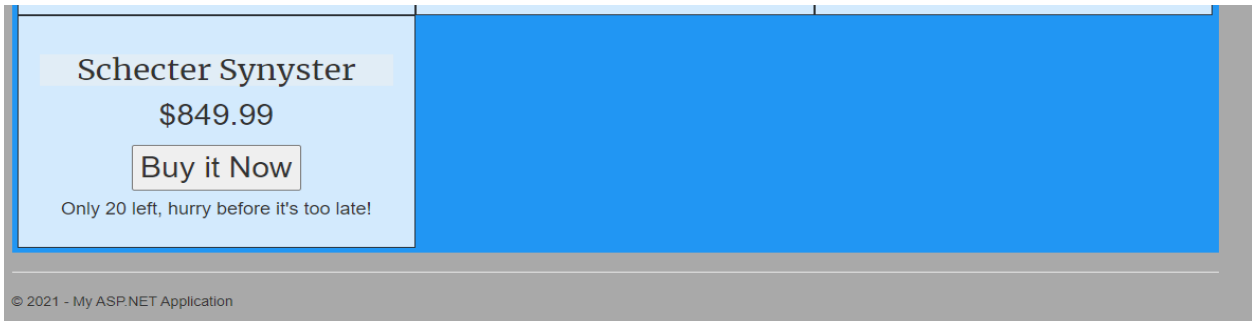
Item Ordered:

CustId	CustFirstName	CustLastName	ItemOrdered



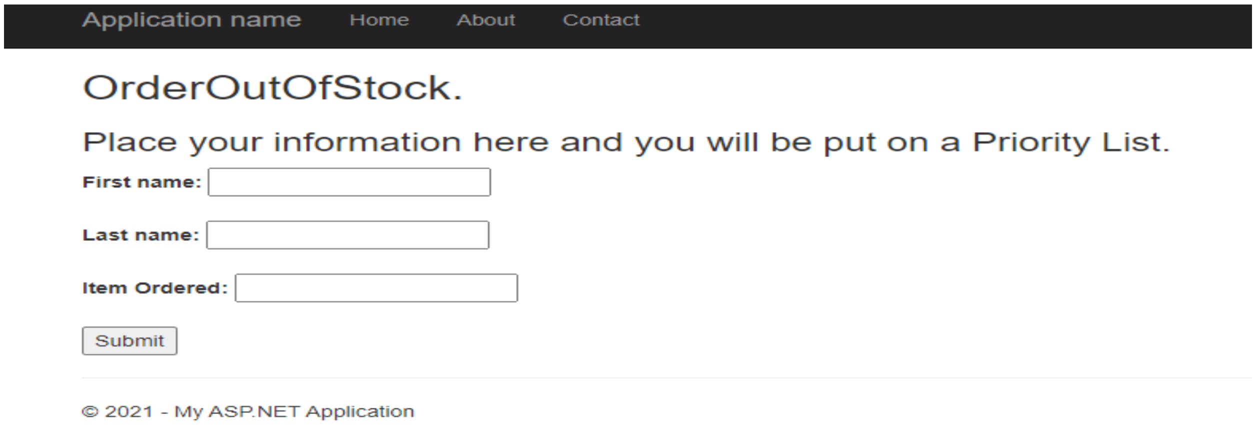
Alex DiDonato's Guitar Store

Dean FlyingV $699.99 Buy it Now Only 20 left, hurry before it's too late!	Epiphone Casino $649.99 Buy it Now Only 20 left, hurry before it's too late!	Fender Mustang $449.99 Buy it Now Only 20 left, hurry before it's too late!
Fender Strat $499.99 Buy it Now Only 20 left, hurry before it's too late!	Fender Telecaster $549.99 Buy it Now Only 20 left, hurry before it's too late!	Gibson Les Paul $799.99 Buy it Now Only 20 left, hurry before it's too late!
Gibson SG $599.99 Buy it Now Only 20 left, hurry before it's too late!	Ibanez RG $299.99 Buy it Now Only 20 left, hurry before it's too late!	Jackson JS $399.99 Buy it Now Only 20 left, hurry before it's too late!



Schecter Synyster $849.99 Buy it Now Only 20 left, hurry before it's too late!		

© 2021 - My ASP.NET Application



Application name Home About Contact

OrderOutOfStock.

Place your information here and you will be put on a Priority List.

First name:

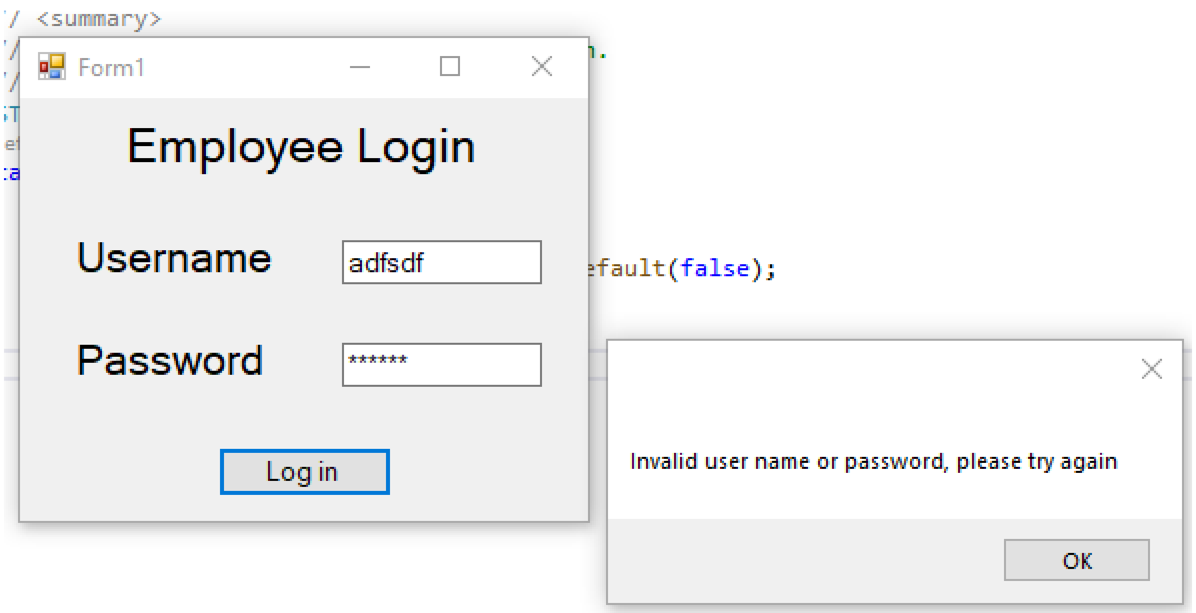
Last name:

Item Ordered:

Submit

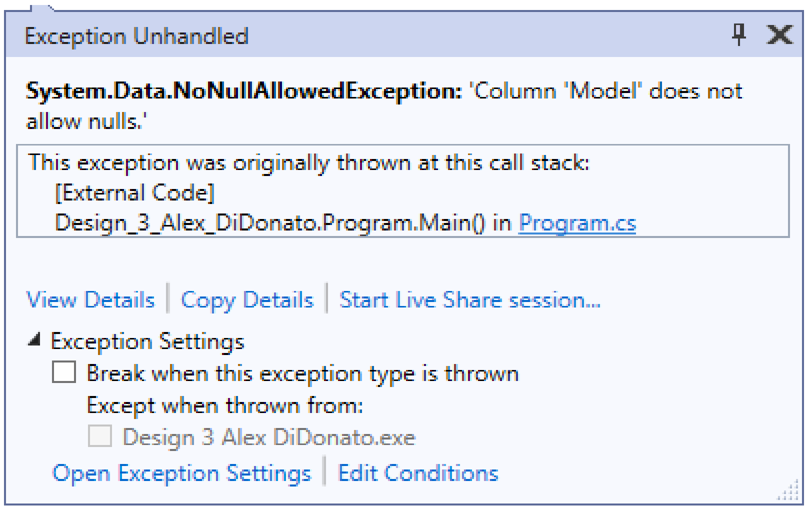
© 2021 - My ASP.NET Application

Output



```
Invalid user name or password, please try again
```

Error messages



Exception Unhandled

System.Data.NoNullAllowedException: 'Column 'Model' does not allow nulls.'

```
This exception was originally thrown at this call stack:
    [External Code]
    Design_3_Alex_DiDonato.Program.Main() in Program.cs
```

View Details | Copy Details | Start Live Share session...

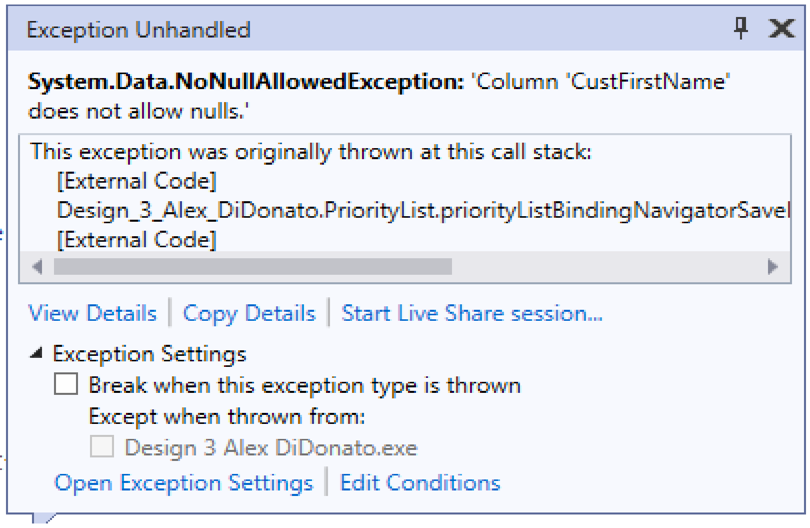
Exception Settings

Break when this exception type is thrown

Except when thrown from:

Design 3 Alex DiDonato.exe

Open Exception Settings | Edit Conditions



Exception Unhandled

System.Data.NoNullAllowedException: 'Column 'CustFirstName' does not allow nulls.'

```
This exception was originally thrown at this call stack:
    [External Code]
    Design_3_Alex_DiDonato.PriorityList.priorityListBindingNavigatorSaveI
    [External Code]
```

View Details | Copy Details | Start Live Share session...

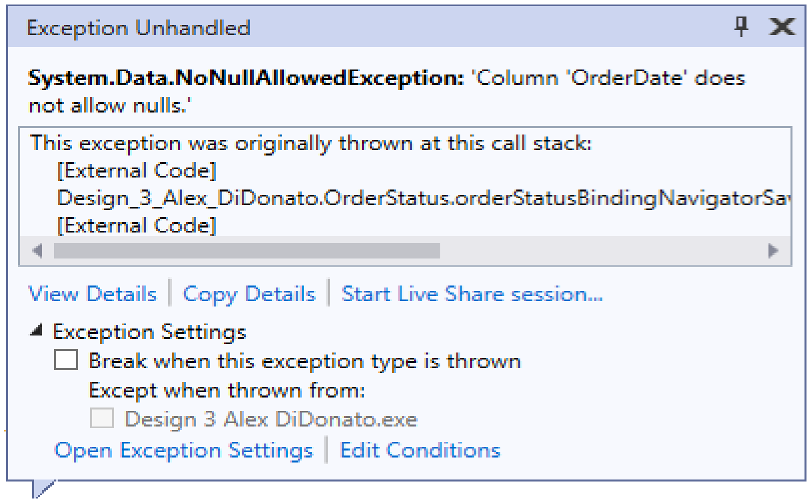
Exception Settings

Break when this exception type is thrown

Except when thrown from:

Design 3 Alex DiDonato.exe

Open Exception Settings | Edit Conditions



Exception Unhandled

System.Data.NoNullAllowedException: 'Column 'OrderDate' does not allow nulls.'

```
This exception was originally thrown at this call stack:
    [External Code]
    Design_3_Alex_DiDonato.OrderStatus.orderStatusBindingNavigatorSa
    [External Code]
```

View Details | Copy Details | Start Live Share session...

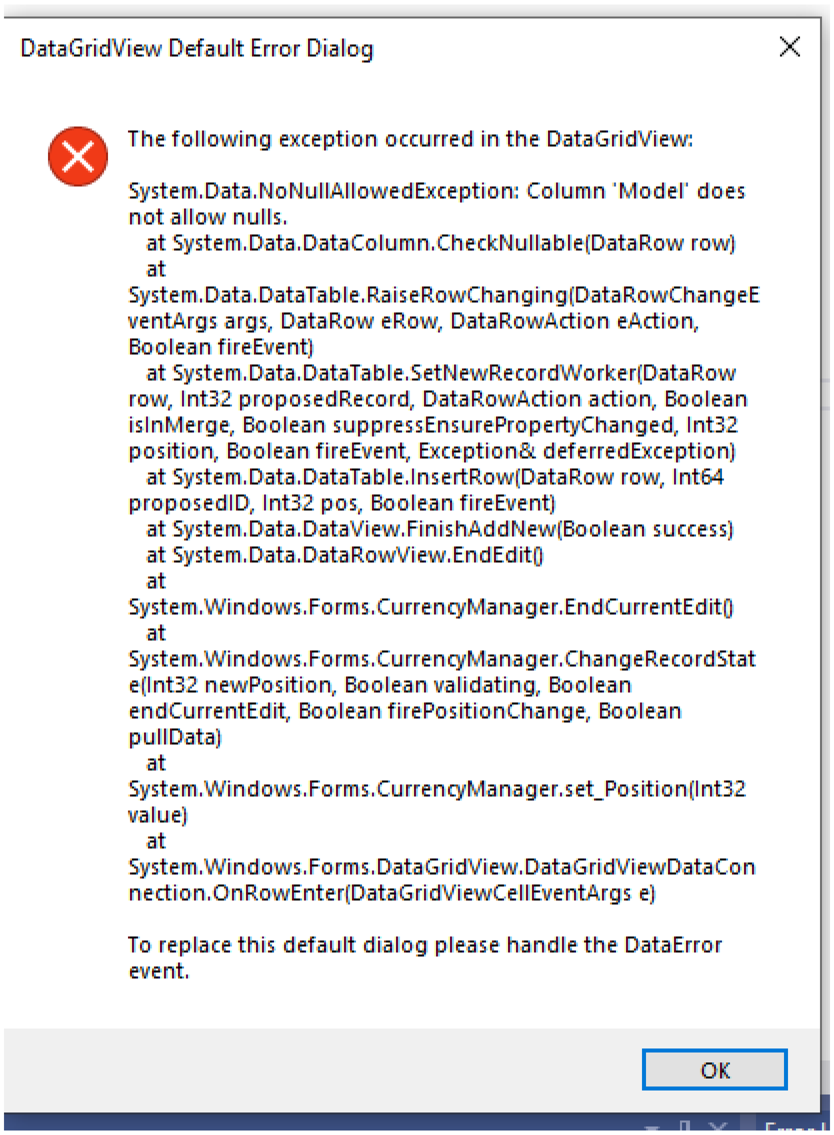
Exception Settings

Break when this exception type is thrown

Except when thrown from:

Design 3 Alex DiDonato.exe

Open Exception Settings | Edit Conditions



```
The following exception occurred in the DataGridView:

System.Data.NoNullAllowedException: Column 'Model' does
not allow nulls.
   at System.Data.DataColumn.CheckNullable(DataRow row)
   at
System.Data.DataTable.RaiseRowChanging(DataRowChangeE
ventArgs args, DataRow eRow, DataRowAction eAction,
Boolean fireEvent)
   at System.Data.DataTable.SetNewRecordWorker(DataRow
row, Int32 proposedRecord, DataRowAction action, Boolean
isInMerge, Boolean suppressEnsurePropertyChanged, Int32
position, Boolean fireEvent, Exception& deferredException)
   at System.Data.DataTable.InsertRow(DataRow row, Int64
proposedID, Int32 pos, Boolean fireEvent)
   at System.Data.DataView.FinishAddNew(Boolean success)
   at System.Data.DataRowView.EndEdit()
   at
System.Windows.Forms.CurrencyManager.EndCurrentEdit()
   at
System.Windows.Forms.CurrencyManager.ChangeRecordStat
e(Int32 newPosition, Boolean validating, Boolean
endCurrentEdit, Boolean firePositionChange, Boolean
pullData)
   at
System.Windows.Forms.CurrencyManager.set_Position(Int32
value)
   at
System.Windows.Forms.DataGridView.DataGridViewDataCon
nection.OnRowEnter(DataGridViewCellEventArgs e)

To replace this default dialog please handle the DataError
event.
```

Conclusion

A. I was able to bring most of my vision to reality but some corners had to be cut in order to make some deadlines. I was able to bring the main concepts that I wanted but tiny details had to be cut. Making the employee portal and the website were fine, the main issue I have is integrating the database to the website.

B. The problem that I did not anticipate was the scope of the project as it should have been bigger. I only had the employee side because I was trying to cut corners because I thought it was too big and it was too late to do anything about it.

C. I learned how to connect a database to a website and how to apply the concepts I learned in Systems analysis into this project. This also teaches me time management as I had to balance this project, other classes, and life events that I did not expect to happen.

D. Considering this is only version 1 of my project, there is still a lot more I could have done. For example, I could actually add a login form for the customers and make it more like a shopping site as well. Other minor things that I would worry about are bug fixes and changes to make the program run better.

Users Manual

This manual will show you everything you need to know about using this application, whether an employee or a customer. To use the employee side, enter your username and password, and then hit the log in button. This takes you to the Order Status page where you can see all of the orders that have been placed. From there, you can click on the two buttons down on the bottom that say “Priority List” and “Guitar Inventory”. The Guitar Inventory button takes you to the Guitar Inventory Page and the Priority List takes you to the Priority List. When you go to either of those pages, the format is the same and you can add things to the Guitar Inventory or the Priority List. At the bottom of the two pages, there are also two buttons at the bottom that say “Priority List” and “Order Status” for the Guitar Inventory Page and “Order Status” and “Guitar Inventory” on the Priority List page. All of these buttons function the same, you click on a certain button and they take you to the page that is listed on the button.

Customer Side

For the Customer, They will go to the shop and see 10 guitars that have a buy it now button on all of them. When you click buy it now on any of the guitars, the number below it will go down one as that shows it was purchased. When the amount hits 0 and you click buy it now, it will take you to the OrderOutOfStock page. This is where you will insert your first name, last name, and item you tried to order. When you click submit, a page that says you will be in contact with us shows up, and the order you submitted gets sent to the PriorityList table in the database.