WareMaster

Warehouse Management Application

Jing Wei
Xiaoxing Pan
Shiyuan Xu

Background

WAREMASTER is a desktop application for efficient warehouse management, designed to provide a user-friendly interface for streamline inventory tracking, order processing, and overall warehouse operations.

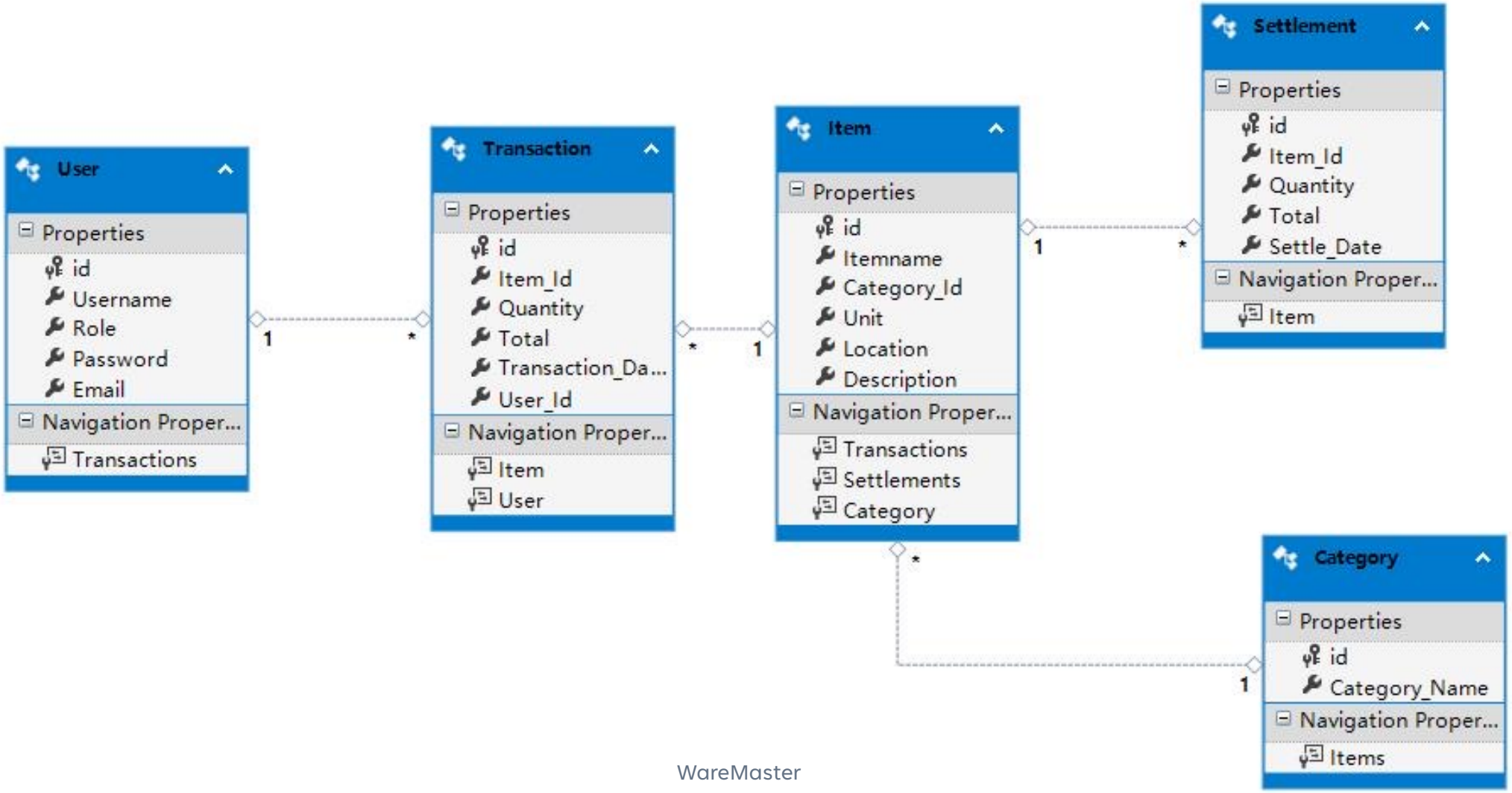
User Features

- User
  - login/out, update password
  - Inventory Management: Initialization, Inbound/outbound, settlement
  - Item Management: Add/edit Items and manage its categories
  - Search and Sort Items/ categories/inventory records
  - Visual data with charts
  - Data Export and Print
- Admin
  - Manage users
  - Delete old records, reset password

Technologies

- WPF, C#
- SQL Server on Azure
- Entity Framework
- Localization
- Logging
- MS test (Unit test)
- MahApps Icons Material
- LiveCharts.Wpf
- FluentValidation
- Export to Excel/PDF
- Print

Database - Azure



Challenge & Solution:

- Simple Logging Solution:
  - During a desktop application's execution, how to recording relevant information, events, or actions.

WMLogger

App.config

```
<configuration>
    <appSettings>
        <add key="logPath" value="C:\WMTemp"/>
    </appSettings>
```

Usage

```
catch(SystemException ex)
{
    Console.WriteLine(ex.ToString());
    WMLogger.WriteLog(ex.ToString());
}
```

WMLogger.cs

```
public static void WriteLog(string message)
{
    try
    {
        string logPath = ConfigurationManager.AppSettings["logPath"];
        string logfile = Path.Combine(logPath, "log.txt"); // Combine with file name
        if (!Directory.Exists(logPath))
        {
            Directory.CreateDirectory(logPath);
        }
        if (!File.Exists(logfile))
        {
            File.WriteAllText(logfile, $"{DateTime.Now} : {message}{Environment.NewLine}");
        }
        else
        {
            File.AppendAllText(logfile, $"{DateTime.Now} : {message}{Environment.NewLine}");
        }
    }catch (Exception ex)
    {
        Console.WriteLine(ex.ToString());
    }
```

Log.txt

log.txt - Notepad

File Edit Format View Help

```
2023-11-12 11:47:21 PM : User ID:8 logged in
2023-11-12 11:53:07 PM : User ID:8 logged out
2023-11-13 12:55:02 AM : User ID:13 logged in
2023-11-13 12:57:46 AM : User ID:13 logged out
2023-11-13 12:55:29 AM : User ID:8 logged in
2023-11-13 12:59:12 AM : User ID:8 logged out
```

Localization:

- Static approach

Use Resource File: Strings.fr.resx

Name	Value
addperson	Ajouter une personne
age	Âge
deleteperson	Supprimer une personne
name	Nom
updateperson	Mettre à jour la personne

- Disadvantages
  - You have to restart your application

In your Window.xaml

```
<Label Content="{x:Static rs:Strings.name}"/>
```

In App.xaml.cs

```
public partial class App : Application
{
    1 reference
    App()
    {
        System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("en-Us");
    }
}
```

Localization:

Dynamic approach:

Use ResourceDictionary

```
<ResourceDictionary xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                    xmlns:system="clr-namespace:System;assembly=mscorlib">
    <system:String x:Key="current-storage">Current Storage</system:String>
    <system:String x:Key="menu-Items">Items</system:String>
</ResourceDictionary>
```

In .xaml file

```
<TextBlock Name="TxblItemCount" Text="{DynamicResource totalItems}" />
```

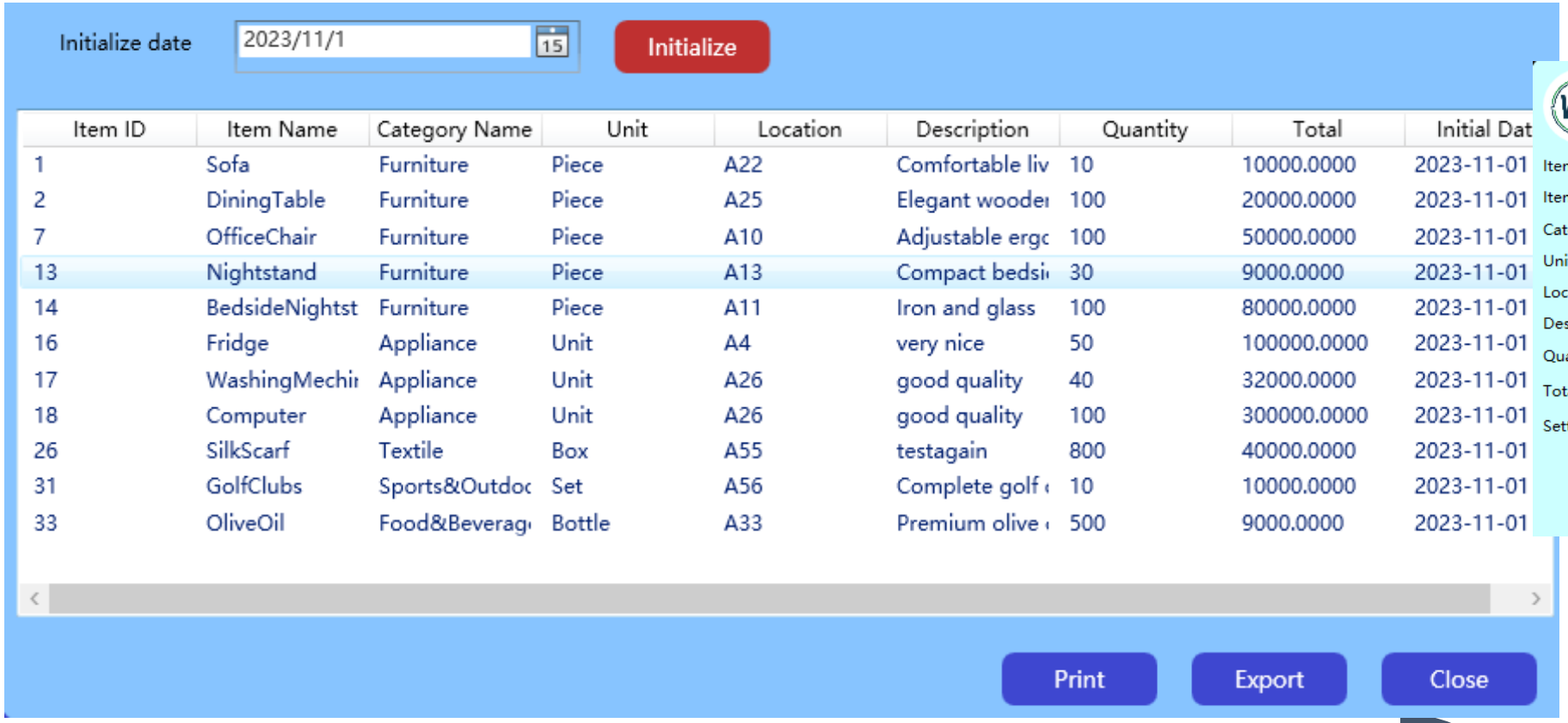
In .xaml.cs file C# code

```
ResourceDictionary dict = new ResourceDictionary();
switch (lang)
{
    case "En":
        dict.Source = new Uri("..\\StringResource.en.xaml", UriKind.Relative);
        break;
    case "Fr":
        dict.Source = new Uri("..\\StringResource.fr.xaml", UriKind.Relative);
        break;
}
this.Resources.MergedDictionaries.Add(dict);
```

Overview: Inventory Initialization

- Initiallize
  - Select initial date
  - Remove all the settlement and transaction data
- Show Data
  - Retrieve and show the first settle data
  - Add / Edit first settle data as initial data
- Print
- Export to Excel

Challenge & Solution:



Item ID	Item Name	Category Name	Unit	Location	Description	Quantity	Total	Initial Dat
1	Sofa	Furniture	Piece	A22	Comfortable liv	10	10000.0000	2023-11-01
2	DiningTable	Furniture	Piece	A25	Elegant wooder	100	20000.0000	2023-11-01
7	OfficeChair	Furniture	Piece	A10	Adjustable ergc	100	50000.0000	2023-11-01
13	Nightstand	Furniture	Piece	A13	Compact bedsi	30	9000.0000	2023-11-01
14	BedsideNightst	Furniture	Piece	A11	Iron and glass	100	80000.0000	2023-11-01
16	Fridge	Appliance	Unit	A4	very nice	50	100000.0000	2023-11-01
17	WashingMechi	Appliance	Unit	A26	good quality	40	32000.0000	2023-11-01
18	Computer	Appliance	Unit	A26	good quality	100	300000.0000	2023-11-01
26	SilkScarf	Textile	Box	A55	testagain	800	40000.0000	2023-11-01
31	GolfClubs	Sports&Outdoc	Set	A56	Complete golf	10	10000.0000	2023-11-01
33	OliveOil	Food&Beverag	Bottle	A33	Premium olive	500	9000.0000	2023-11-01

Challenge & Solution:

Items left join settlements:

Show zero if no initial data inputed.

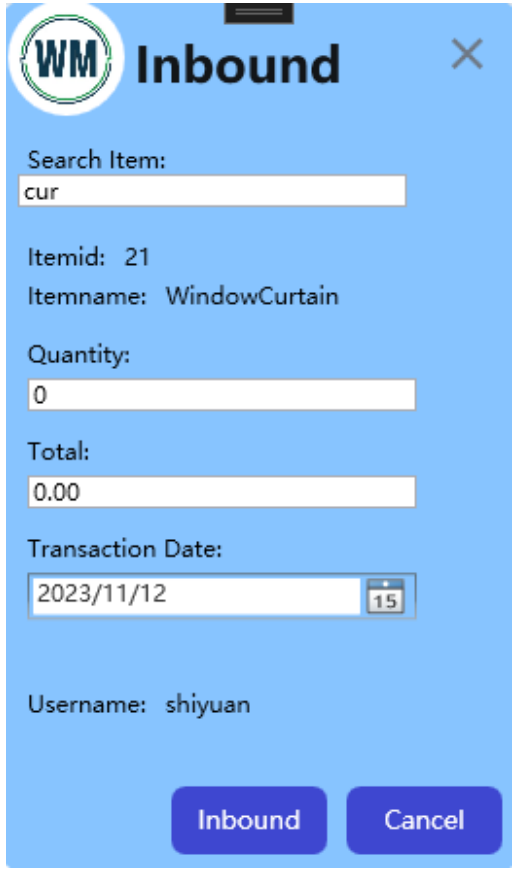
```
var query = from item in Globals.wareMasterEntities.Items
            join settlement in Globals.wareMasterEntities.Settlements
            on item.id equals settlement.Item_Id into gj
            from sub in gj.DefaultIfEmpty()
            where sub == null || sub.Settle_Date == DatePickerInit.SelectedDate
            select new
            {
                ItemId = item.id,
                ItemName = item.Itemname,
                CategoryName = item.Category.Category_Name,
                Unit = item.Unit != null ? item.Unit : string.Empty,
                Location = item.Location != null ? item.Location : string.Empty,
                Description = item.Description != null ? item.Description : string.Empty,
                Quantity = sub != null ? sub.Quantity : 0,
                Total = sub != null ? sub.Total : 0,
                SettleDate = DatePickerInit.SelectedDate,
                SettlementId = sub != null ? sub.id : -1

            };
LvInit.ItemsSource = query.ToList();
```

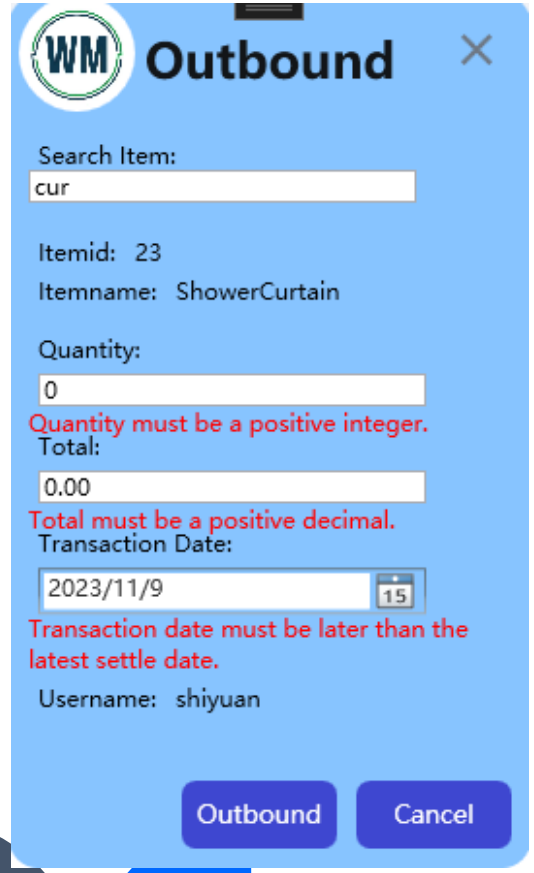
What we learned:

- ❑ Retrieving and binding data
- ❑ PrintDialog & FlowDocument
- ❑ ExcelPackage

Overview: Transactions

- ❑ Inbound & Outbound using same window
- ❑ Creating a transaction object & binding
- ❑ Validation & Processing





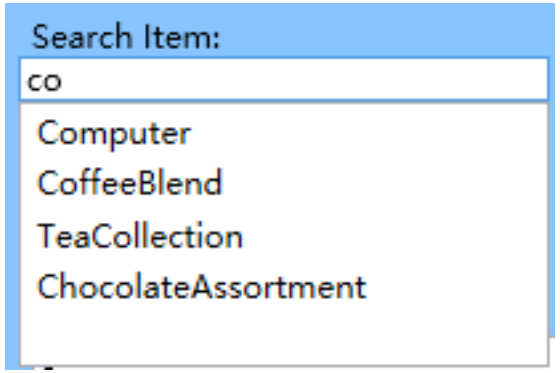
Challenge & Solution:

```
private string option;
private Transaction transaction;
private User user;
private Item item;
2 references
public InventoryChange(String option)
{
    this.option = option;
    InitializeComponent();
    Title = option;
    txtTitle.Text = option;
    ConfirmButton.Content = option;

    user = Globals.wareMasterEntities.Users
        .FirstOrDefault(u => u.Username == Globals.Username);
    TransactionInit();

}
```

```
DateTime lastSettleDate=Inventory.GetLastSettleDate();
if (transaction.Transaction_Date <= lastSettleDate)
{
    datelValidation.Text = "Transaction date must be later than
    isValid=false;
}
else
{
    datelValidation.Text = "";
}
```

What we learned:

Search: popup list when text changed



Overview: Settlement

Settle

- The last settlement data
- Inbounds
- Outbounds
- No transactions before the settle date are allowed

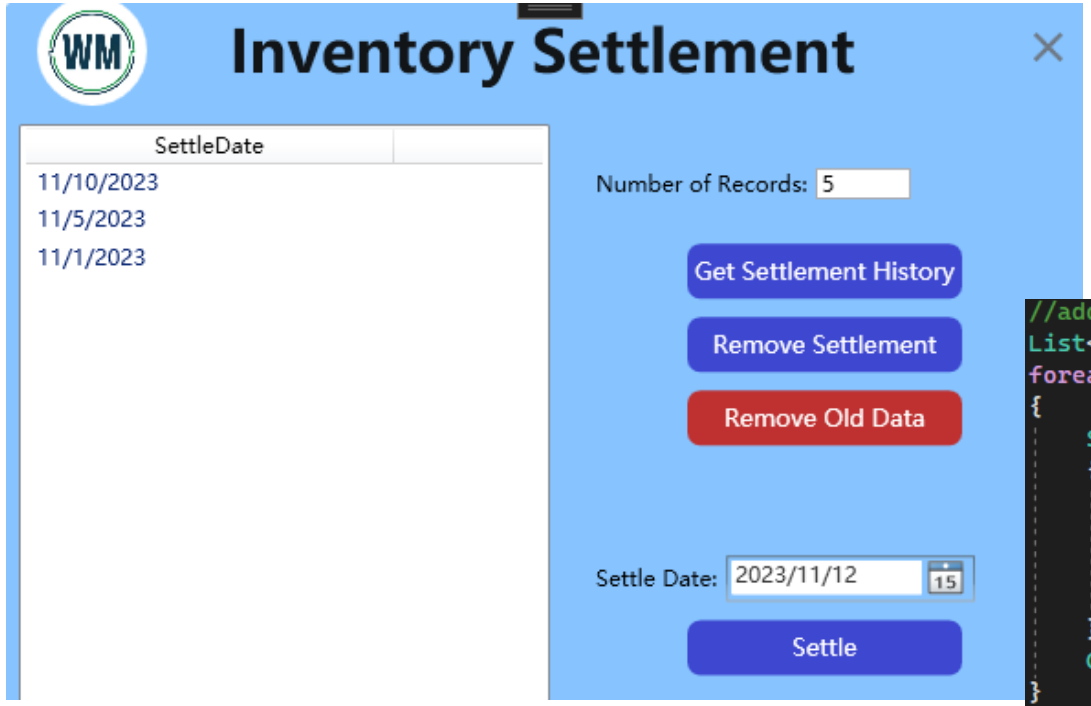
Settlement history

- Retrieve settlement dates
- Delete settlement data

Remove Old Data

- Dangerous operation
- For admin only
- Remove all settlement & transaction data before a settle_date to improve database efficiency

Challenge & Solution:




```
List<DateTime> recentSettlementDates = Globals.wareMasterEntities
    .Settlements
    .Select(s => s.Settle_Date)
    .Distinct()
    .OrderByDescending(date=>date)
    .Take(numOfRecords)
    .ToList();
LVSettle.ItemsSource = recentSettlementDates;
```

```
//add settlements
List<InventoryData> inventorys = Inventory.GetAllInventoriesByItem(settleDate);
foreach (InventoryData inventory in inventorys)
{
    Settlement newSettlement = new Settlement
    {
        Item_Id = inventory.id,
        Settle_Date = settleDate,
        Quantity = inventory.Quantity,
        Total = inventory.Total,
    };
    Globals.wareMasterEntities.Settlements.Add(newSettlement);
}
// save changes
try
{
    Globals.wareMasterEntities.SaveChanges();
```

What we learn:

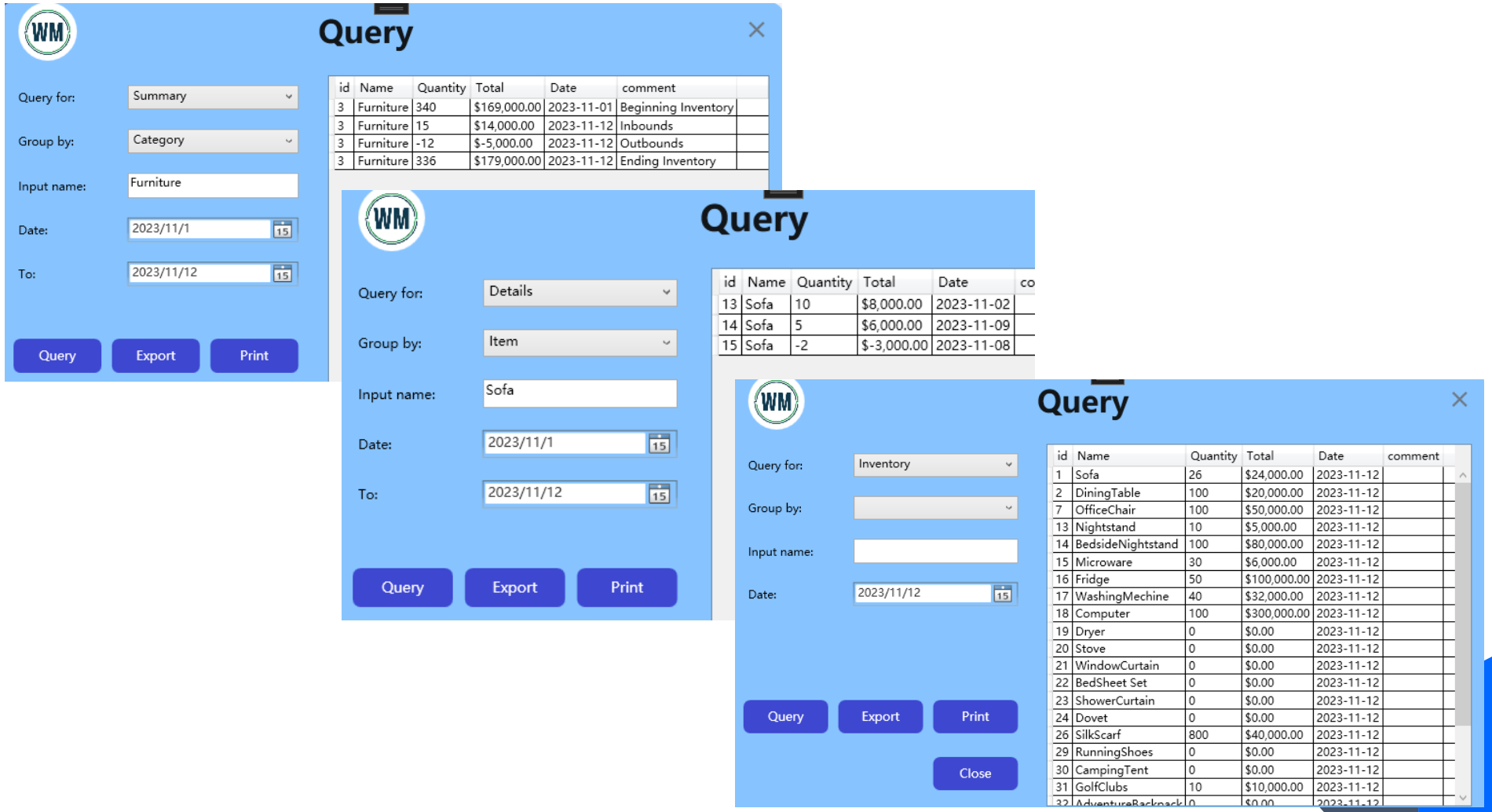
- User experience:
  - Warnings for risky operations
  - Different from ordinary information

```
if (LVSettle.SelectedItem == null)
{
    MessageBox.Show("Please select a settle date",
        "Information",
        MessageBoxButton.OK,
        MessageBoxImage.Information);
    return;
}
DateTime settleDate = (DateTime)LVSettle.SelectedItem;
if (MessageBoxResult.No == MessageBox.Show($"Are you sure to remove all
    "Confirm",
    MessageBoxButton.YesNo,
    MessageBoxImage.Warning))
{
    return;
}
```

Overview: Query

- ❑ Query Inventory, Summary or Details
- ❑ Group by Items or Categories
- ❑ Search Item/Category by Name
- ❑ Print & Export

Challenge & Solution:

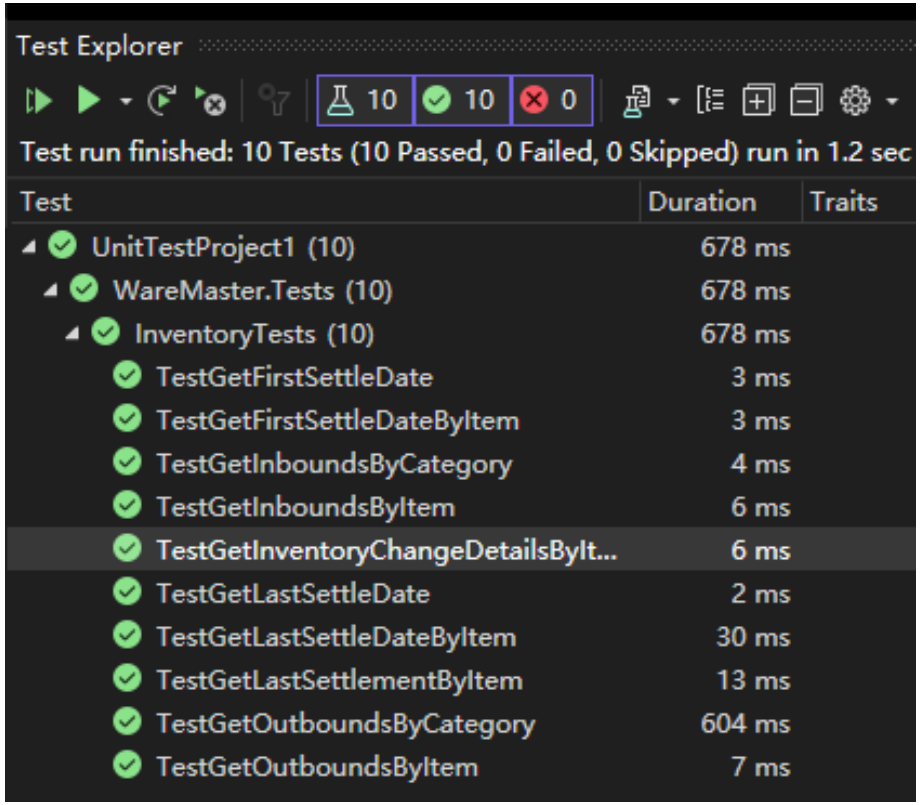


Challenge & Solution: Class for Query

```
Inventory.cs
  InventoryData
  Inventory
    GetFirstSettleDate() : DateTime
    GetLastSettleDate() : DateTime
    GetFirstSettleDateByItem(Item) : DateTime
    GetLastSettleDateByItem(Item) : DateTime
    GetLastSettlementByItem(Item, DateTime) : Settlement
    GetFirstInventories() : List<InventoryData>
    GetInventoryByItem(Item, DateTime) : InventoryData
    GetInventoryByCategory(Category, DateTime) : InventoryData
    GetAllInventoriesByItem(DateTime) : List<InventoryData>
    GetAllInventoriesByCategory(DateTime) : List<InventoryData>
    GetInboundsByItem(Item, DateTime, DateTime) : List<InventoryData>
    GetInboundsByCategory(Category, DateTime, DateTime) : List<InventoryData>
    GetSumInboundsByItem(Item, DateTime, DateTime) : InventoryData
    GetSumInboundsByCategory(Category, DateTime, DateTime) : InventoryData
    GetOutboundsByItem(Item, DateTime, DateTime) : List<InventoryData>
    GetOutboundsByCategory(Category, DateTime, DateTime) : List<InventoryData>
    GetSumOutboundsByItem(Item, DateTime, DateTime) : InventoryData
    GetSumOutboundsByCategory(Category, DateTime, DateTime) : InventoryData
    GetSummaryByItem(Item, DateTime, DateTime) : List<InventoryData>
    GetAllSummaryByItem(DateTime, DateTime) : List<InventoryData>
    GetSummaryByCategory(Category, DateTime, DateTime) : List<InventoryData>
    GetAllSummaryByCategory(DateTime, DateTime) : List<InventoryData>
    GetInventoryChangeDetailsByItem(Item, DateTime, DateTime) : List<InventoryData>
    GetAllInventoryChangeDetailsByItem(DateTime, DateTime) : List<InventoryData>
    GetInventoryChangeDetailsByCategory(Category, DateTime, DateTime) : List<InventoryData>
```

What we learn:

- ❑ Encapsulation and reuse
- ❑ Unit Test with MSTest



Overview: Items/Categories Management

- Items => Add/Edit, View Items list, Search by Name, Sort
- Categories => Add/Edit, View Categories list, Search by Name, Sort

Challenges & Solutions: Input Validation

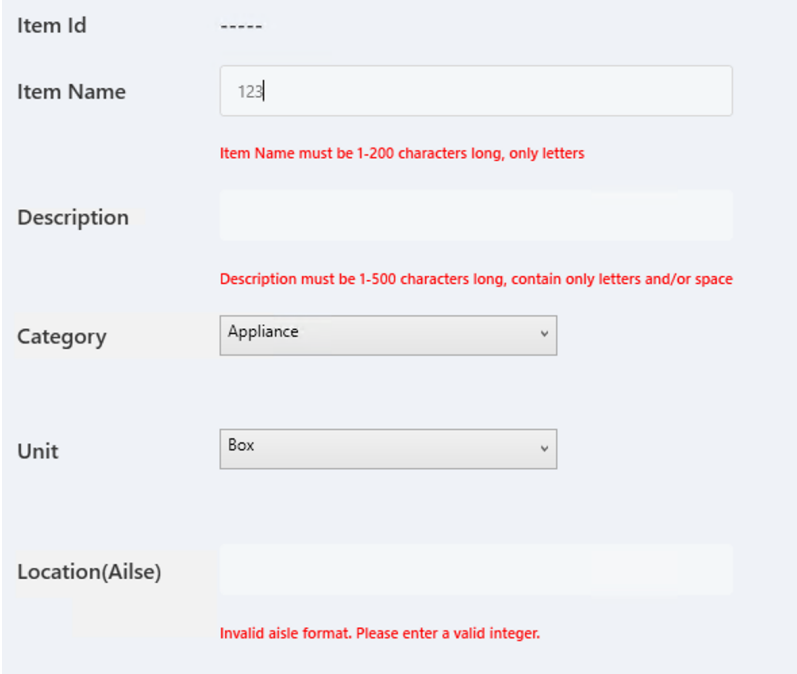
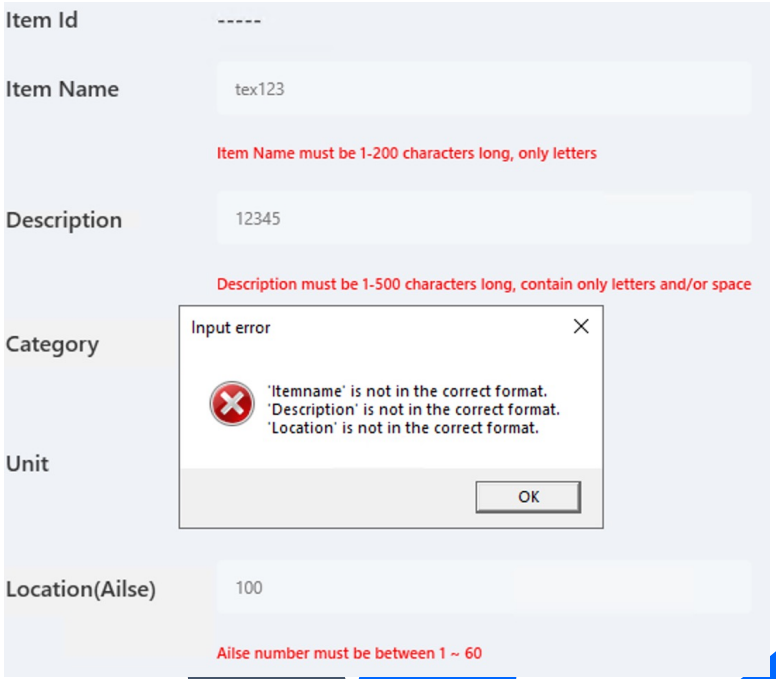


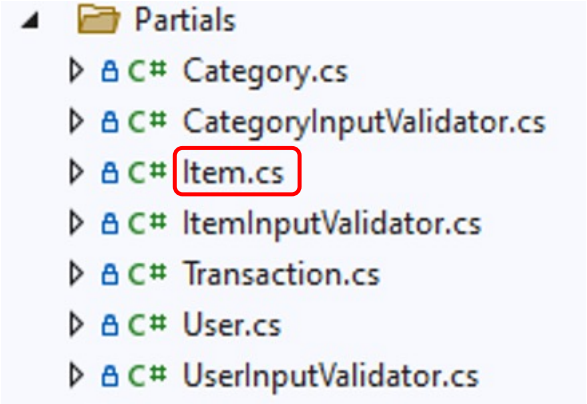
Fig1. When LostFocus

Fig2. When BtnSave_Click



Challenges & Solutions: Input Validation

- When LostFocus

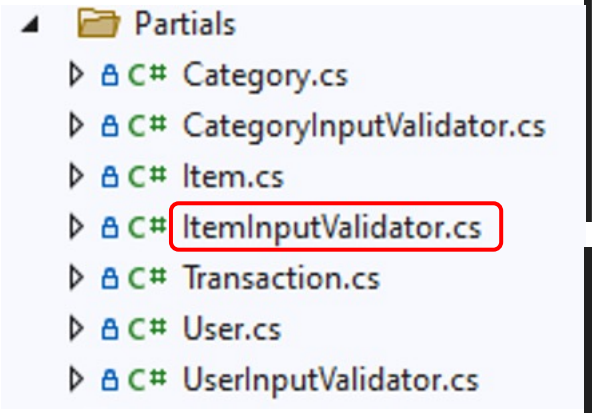



```
namespace WareMaster
{
    public partial class Item
    {

        public static bool IsItemNameValid(string itemname, int index, int itemid, out string error)
        {
            List<string> allNames = Globals.wareMasterEntities.Items.Select(item => item.Itemname.ToLower()).ToList();
            List<string> otherNames = Globals.wareMasterEntities.Items
            .Where(item => item.id != itemid)
            .Select(item => item.Itemname.ToLower())
            .ToList();
            if (itemname.Length < 1 || itemname.Length > 200 || !Regex.IsMatch(itemname, "^[a-zA-Z]+$"))
            {
                error = "Item Name must be 1-200 characters long, only letters";
                return false;
            }
            else if (index == 0 && allNames.Contains(itemname.ToLower()) || index == 1 && otherNames.Contains(itemname.ToLower()))  // 0 is add, 1 is edit
            {
                error = "Itemname must be unique";
                return false;
            }
            error = null;
            return true;
        }
```

Challenges & Solutions: Input Validation

FluentValidation by Jeremy Skinner
A validation library for .NET that uses a flue

- When BtnSave_Click

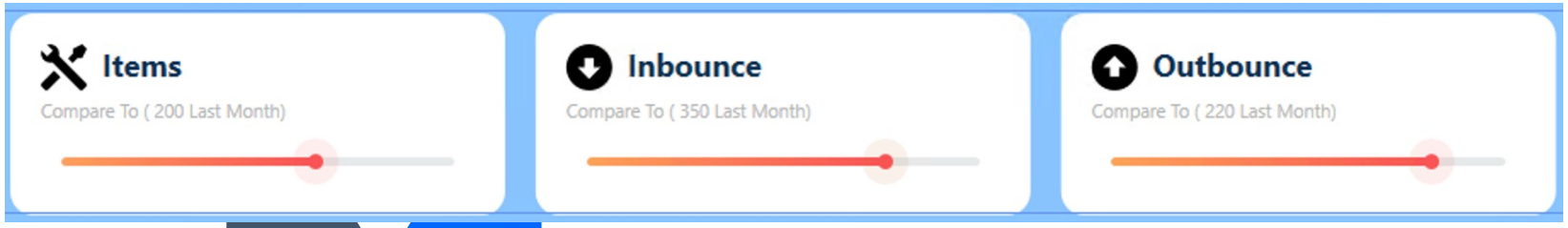



```
namespace WareMaster
{
    public class ItemInputValidator : AbstractValidator<Item>
    {
        public ItemInputValidator(int index, int itemid)
        {
            RuleFor(Item => Item.Itemname).NotNull().NotEmpty().Length(1, 200).Matches("^[a-zA-Z]+$").Must((item, Itemname) => IsItemnameUnique(Itemname, index, itemid))
            .WithMessage("Itemname must be unique");  // only contains letters
            RuleFor(Item => Item.Description).NotNull().NotEmpty().Length(1, 500).Matches("^[a-zA-Z\\s]+$");  // only contains letters and/or space
            RuleFor(Item => Item.Category_Id).NotNull().NotEmpty().Must((item, Category_Id) => IsCategoryIdExist(Category_Id)).WithMessage("Category does not exist.");
            RuleFor(Item => Item.Unit).NotNull().NotEmpty();
            RuleFor(Item => Item.Location).NotNull().NotEmpty().Matches("^A([1-9]|[1-4][0-9]|60)$");  // A1
        }

        private bool IsItemnameUnique(string itemname, int index, int itemid)
        {
```

```
var validator = new ItemInputValidator(index, currItem.id);
var result = validator.Validate(itemToUpdate);
if (!result.IsValid)
{
    throw new ArgumentException(result.ToString(Environment.NewLine));
}
```

Overview:
Visual Data with Charts

- Items => Monthly Total inbound/outbound
- Categories => Percentage of each Category
- Goal vs Actual, Last Month vs This Month

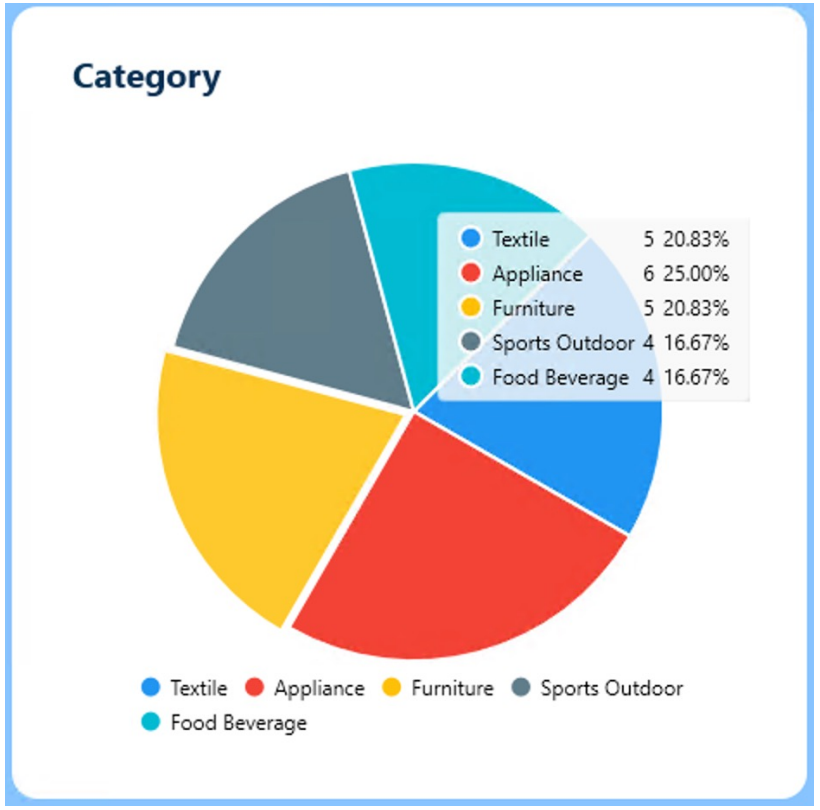


Challenge & Solution: Charts

LiveCharts.Wpf by Beto Rodriguez
Simple, flexible, interactive and powerful


```xml
<!--chart-->
<lvc:CartesianChart Grid.Row="1" Margin="5 60 0 10" LegendLocation="None">
    <lvc:CartesianChart.DataTooltip>
        <lvc:DefaultTooltip Background="Red" Foreground="#ffffff" BulletSize="10" Opacity="0.7" />
    </lvc:CartesianChart.DataTooltip>
    <lvc:CartesianChart.AxisX>
        <lvc:Axis Foreground="#90a3b2" ShowLabels="True" MinValue="1" MaxValue="12">
            <lvc:Axis.Separator>
                <lvc:Separator StrokeThickness="0" Step="1"/>
            </lvc:Axis.Separator>
        </lvc:Axis>
    </lvc:CartesianChart.AxisX>
    <lvc:CartesianChart.AxisY>
        <lvc:Axis Foreground="#90a3b2" ShowLabels="True" MinValue="0" >
            <lvc:Axis.Separator>
                <lvc:Separator StrokeThickness="0"/>
            </lvc:Axis.Separator>
        </lvc:Axis>
    </lvc:CartesianChart.AxisY>
```

Challenge & Solution: Charts




```
<!--category chart-->
<Border Style="{StaticResource whiteBorder}" Margin="0 20 20 0" Grid.Column="1">
    <Grid Margin="15 5">
        <!--Title-->
        <TextBlock Text="Category" Style="{StaticResource titleTextA}" />
        <!--chart-->
        <lvc:PieChart LegendLocation="Bottom" Hoverable="True" Grid.Row="1" Margin="5 60 0 10" Series="{Binding PieSeriesCollection}">
            <lvc:PieChart.ChartLegend>
                <lvc:DefaultLegend BulletSize="15" TextBlock.FontSize="8"></lvc:DefaultLegend>
            </lvc:PieChart.ChartLegend>
            <lvc:PieChart.DataTooltip>
                <lvc:DefaultTooltip BulletSize="20">
            </lvc:DefaultTooltip>
            </lvc:PieChart.DataTooltip>
        </lvc:PieChart>
    </Grid>
</Border>
```

```
    var categoryItems = query.ToList();
    PieSeriesCollection = new SeriesCollection();

    foreach (var categoryItem in categoryItems)
    {
        PieSeriesCollection.Add(new PieSeries
        {
            Title = categoryItem.CategoryName,
            Values = new ChartValues<double> { categoryItem.TotalItems },
            DataLabels = true
        });
    }
    DataContext = this;
}

public SeriesCollection PieSeriesCollection { get; set; }
```

What we learnt:

- Documentation + Tutorial

Future Work

More Analyzation

Monthly & Weekly Inventory reports

Multi-warehouse

Mult-tenant Database

Loading & Unloading Scheduling

Chief Operation Officer

Cost & Profit Accounting

Accounting reports

Summary

We built a warehouse management application which has a log and implement the user authentication , inventory initialization and settlement, inbound/outbound tracking, custom search the inventory record, items/categories/users CRUD, data export and print.

In addition, we did unit test and localization.

Thank you