```
using System;
using System.Globalization;
using System.Text.RegularExpressions;
using MySqlConnector;

namespace MauiApp1.Database
{
    public class CNP
    {
        // Connection string with SSL mode set to 'None'
        private string connectionString =
"server=localhost;port=3306;database=test_schema;user=root;password=mysqlserverprog;";

        // Method to place an order
        public string PlaceOrder(string productName, string price, string receiverName, string
deliveryMethod, string paymentMethod)
        {
            try
            {
                using (var conn = new MySqlConnection(connectionString))
                {
                    conn.Open();

                    string orderId = Guid.NewGuid().ToString();
                    string productId = Guid.NewGuid().ToString();
                    string customerId = Guid.NewGuid().ToString();
                    string deliveryMethodId = Guid.NewGuid().ToString();
                    string paymentMethodId = Guid.NewGuid().ToString();

                    // ✅ Normalize price input
                    price = Regex.Replace(price, @"[^\d,\.]", "").Trim().Replace(',', '.');

                    if (!decimal.TryParse(price, NumberStyles.AllowDecimalPoint,
CultureInfo.InvariantCulture, out decimal parsedPrice))
                    {
                        return $"Error: Invalid price format: '{price}'";
                    }

                    // ✅ Transaction to ensure all-or-nothing
                    using (var transaction = conn.BeginTransaction())
                    {
                        void Insert(string query, Dictionary<string, object> parameters)
                        {
                            using (var cmd = new MySqlCommand(query, conn, transaction))
```

```
                {
                    foreach (var param in parameters)
                        cmd.Parameters.AddWithValue(param.Key, param.Value);
                    cmd.ExecuteNonQuery();
                }
            }

            Insert("INSERT INTO products (ProductId, ProductName, Price) VALUES (@Id,
@Name, @Price)",
                new()
                {
                    ["@Id"] = productId,
                    ["@Name"] = productName,
                    ["@Price"] = parsedPrice
                });

            Insert("INSERT INTO customers (CustomerId, ReceiverName) VALUES (@Id,
@Name)",
                new()
                {
                    ["@Id"] = customerId,
                    ["@Name"] = receiverName
                });

            Insert("INSERT INTO delivery_methods (DeliveryMethodId, MethodName)
VALUES (@Id, @Name)",
                new()
                {
                    ["@Id"] = deliveryMethodId,
                    ["@Name"] = deliveryMethod
                });

            Insert("INSERT INTO payment_methods (PaymentMethodId, MethodName)
VALUES (@Id, @Name)",
                new()
                {
                    ["@Id"] = paymentMethodId,
                    ["@Name"] = paymentMethod
                });

            Insert("INSERT INTO orders (OrderId, ProductId, CustomerId, DeliveryMethodId,
PaymentMethodId) " +
                   "VALUES (@OrderId, @ProductId, @CustomerId, @DeliveryMethodId,
@PaymentMethodId)",
```

```
                    new()
                    {
                        ["@OrderId"] = orderId,
                        ["@ProductId"] = productId,
                        ["@CustomerId"] = customerId,
                        ["@DeliveryMethodId"] = deliveryMethodId,
                        ["@PaymentMethodId"] = paymentMethodId
                    });

                transaction.Commit();
                return "Order placed successfully.";
            }
        }
    }
    catch (Exception ex)
    {
        return "Error: " + ex.Message;
    }
  }
 }
}
```

```
using MauiApp1.Models;
using System;
using Microsoft.Maui.Controls;

namespace MauiApp1
{
   public partial class TransactionPage : ContentPage
   {
      private Item _selectedItem;

      public TransactionPage(Item selectedItem)
      {
         InitializeComponent();
         _selectedItem = selectedItem;
         BindingContext = _selectedItem;
      }

      private async void OnPlaceOrderClicked(object sender, EventArgs e)
      {
         if (PaymentPicker == null || DeliveryPicker == null || NameEntry == null)
```

```
            {
                await DisplayAlert("Error", "Something went wrong. Please try again.", "OK");
                return;
            }

            if (PaymentPicker.SelectedItem == null)
            {
                await DisplayAlert("Error", "Please select a mode of payment.", "OK");
                return;
            }

            if (DeliveryPicker.SelectedItem == null)
            {
                await DisplayAlert("Error", "Please select a delivery method.", "OK");
                return;
            }

            if (string.IsNullOrWhiteSpace(NameEntry.Text))
            {
                await DisplayAlert("Error", "Please enter the name of the receiver.", "OK");
                return;
            }

            // Success message when all requirements are met
            await DisplayAlert("Success", "Order has been placed!", "OK");
        }
    }
}

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MauiApp1.TransactionPage">

    <ContentPage.Content>
        <StackLayout Padding="20">
            <!-- Product Image -->
            <Image Source="{Binding Image}" HeightRequest="200" Aspect="AspectFit"/>

            <!-- Product Name -->
            <Label Text="{Binding Name}"
                   FontAttributes="Bold"
                   FontSize="20"
                   Margin="0,10,0,5"/>
```

```
        <!-- Product Price -->
        <Label Text="{Binding Price}"
               FontSize="18"
               TextColor="Green"
               Margin="0,5,0,10"/>

        <!-- Mode of Payment -->
        <Label Text="Mode of Payment" FontAttributes="Bold" FontSize="18"
Margin="0,10,0,5"/>
        <Picker x:Name="PaymentPicker" Title="Select Mode of Payment" FontSize="16">
           <Picker.ItemsSource>
              <x:Array Type="{x:Type x:String}">
                 <x:String>Cash on Delivery</x:String>
                 <x:String>Credit/Debit Card</x:String>
                 <x:String>Gcash</x:String>
              </x:Array>
           </Picker.ItemsSource>
        </Picker>

        <!-- Delivery Method -->
        <Label Text="Delivery Method" FontAttributes="Bold" FontSize="18" Margin="0,10,0,5"/>
        <Picker x:Name="DeliveryPicker" Title="Select Delivery Method" FontSize="16">
           <Picker.ItemsSource>
              <x:Array Type="{x:Type x:String}">
                 <x:String>Standard Delivery</x:String>
                 <x:String>Express Delivery</x:String>
              </x:Array>
           </Picker.ItemsSource>
        </Picker>

        <!-- Name of Receiver -->
        <Label Text="Name of the Receiver" FontAttributes="Bold" FontSize="18"
Margin="0,10,0,5"/>
        <Entry x:Name="NameEntry" Placeholder="Enter receiver's name" FontSize="16"/>

        <!-- Contact Info -->
        <Label Text="For customization, please contact this number: 09946976100"
               FontSize="16"
               TextColor="Red"
               Margin="0,10,0,0"/>

        <!-- Submit Order Button -->
        <Button Text="Place Order"
                HorizontalOptions="Center"
```

```
                Margin="0,20,0,0"
                Clicked="OnPlaceOrderClicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>

using MauiApp1.Models;
using MvvmHelpers;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace MauiApp1.ViewModels
{
    public class MainPageViewModels : LocalBaseViewModel
    {
        public MainPageViewModels()
        {
            LoadItemCategories();
            LoadPopularItemsCollection();
        }

        public List<ItemCategory> Categories { get; set; }
        public List<Item> AllPopularItems { get; set; } // master list for filtering

        public ObservableRangeCollection<ItemCategory> ItemCategories { get; set; }
        public ObservableRangeCollection<Item> PopularItems { get; set; }

        public void LoadItemCategories()
        {
            ItemCategories = new ObservableRangeCollection<ItemCategory>();
            Categories = new List<ItemCategory>();

            Categories.Add(new ItemCategory
            {
                Name = "Wooden Made",
                Image = "wood.png"
            });

            Categories.Add(new ItemCategory
            {
                Name = "Bracelet",
                Image = "breslet.png"
            });
```

```
    Categories.Add(new ItemCategory
    {
        Name = "Tumbler",
        Image = "tamblir.png"
    });

    Categories.Add(new ItemCategory
    {
        Name = "Bag",
        Image = "bagvector.png"
    });

    ItemCategories.AddRange(Categories);
}

public void LoadPopularItemsCollection()
{
    PopularItems = new ObservableRangeCollection<Item>();
    AllPopularItems = new List<Item>();

    AllPopularItems.Add(new Item
    {
        Name = "Phone Holder",
        Brand = "Crafts and Prints",
        Image = "woodenphoneholder.jpeg",
        Price = "₱ 150.00"
    });

    AllPopularItems.Add(new Item
    {
        Name = "Fan",
        Brand = "Crafts and Prints",
        Image = "souvfan.jpeg",
        Price = "₱ 60.00"
    });

    AllPopularItems.Add(new Item
    {
        Name = "Sunglass",
        Brand = "Crafts and Prints",
        Image = "woodenglass.jpeg",
        Price = "₱ 80.00"
    });
```

```
AllPopularItems.Add(new Item
{
    Name = "Digital Clock",
    Brand = "Crafts and Prints",
    Image = "woodendigitalclock.jpeg",
    Price = "₱ 150.00"
});

AllPopularItems.Add(new Item
{
    Name = "Bracelet",
    Brand = "Crafts and Prints",
    Image = "souvbracelet.jpeg",
    Price = "₱ 60.00"
});

AllPopularItems.Add(new Item
{
    Name = "Tumbler",
    Brand = "Crafts and Prints",
    Image = "souvtumb.jpeg",
    Price = "₱ 100.00"
});

AllPopularItems.Add(new Item
{
    Name = "Bamboo Pen",
    Brand = "Crafts and Prints",
    Image = "bamboopen.jpeg",
    Price = "₱ 10.00"
});

AllPopularItems.Add(new Item
{
    Name = "Carved Can Opener",
    Brand = "Crafts and Prints",
    Image = "carvedcanopener.jpeg",
    Price = "₱ 85.00"
});

AllPopularItems.Add(new Item
{
    Name = "Custom Coaster",
```

```
    Brand = "Crafts and Prints",
    Image = "customizedcoaster.jpeg",
    Price = "₱ 60.00"
});

AllPopularItems.Add(new Item
{
    Name = "Custom Mug",
    Brand = "Crafts and Prints",
    Image = "customizedmug.jpeg",
    Price = "₱ 60.00"
});

AllPopularItems.Add(new Item
{
    Name = "Custom Pin",
    Brand = "Crafts and Prints",
    Image = "customizedpin.jpeg",
    Price = "₱ 30.00"
});

AllPopularItems.Add(new Item
{
    Name = "Custom Wallet",
    Brand = "Crafts and Prints",
    Image = "customizedwallet.jpeg",
    Price = "₱ 50.00"
});

AllPopularItems.Add(new Item
{
    Name = "Custom Wine Glass",
    Brand = "Crafts and Prints",
    Image = "customizedwineglass.jpeg",
    Price = "₱ 350.00"
});

AllPopularItems.Add(new Item
{
    Name = "Sanitizer",
    Brand = "Crafts and Prints",
    Image = "sanitizer.jpeg",
    Price = "₱ 20.00"
});
```

```
            AllPopularItems.Add(new Item
            {
                Name = "Small Bag v1",
                Brand = "Crafts and Prints",
                Image = "souvbag.jpeg",
                Price = "₱ 150.00"
            });

            AllPopularItems.Add(new Item
            {
                Name = "Small Bag v2",
                Brand = "Crafts and Prints",
                Image = "souvbag2.jpeg",
                Price = "₱ 150.00"
            });

            PopularItems.AddRange(AllPopularItems);
        }

        // ✅ Filter method to be called from code-behind
        public void FilterItems(string searchText)
        {
            if (string.IsNullOrWhiteSpace(searchText))
            {
                PopularItems.ReplaceRange(AllPopularItems);
            }
            else
            {
                var filtered = AllPopularItems
                    .Where(i => i.Name.ToLower().Contains(searchText.ToLower()))
                    .ToList();

                PopularItems.ReplaceRange(filtered);
            }
        }
    }
}



using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using MauiApp1.Models;


namespace MauiApp1.Models
{
  public class Transaction
   {
       public int Id { get; set; }
       public string ItemName { get; set; }
       public int Quantity { get; set; }
       public decimal Price { get; set; }
       public string PaymentMode { get; set;
   }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiApp1.Models
{
   public class Item
   {
       public string Name { get; set; }
       public string Brand { get; set; }

       public string Price { get; set; }

       public string Image { get; set; }
   }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using MauiApp1.Models;


namespace MauiApp1.Models
```

```
{
    public class Items
    {
        public string Name { get; set; }
        public string Brand { get; set; }
        public string Image { get; set; }
        public string Price { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MauiApp1.Models
{
    public class ItemCategory
    {
        public string Name { get; set; }
        public decimal Price { get; set; }
        public string Image { get; set; }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Threading.Tasks;

namespace MauiApp1.ViewModels
{
    public class LocalBaseViewModel : INotifyPropertyChanged
    {

        public LocalBaseViewModel()
        {

        }
```

```
        protected bool SetProperty<T>(ref T backingStore, T value, [CallerMemberName] string
propertyName = "", Action? onChanged = null)
        {
            if (EqualityComparer<T>.Default.Equals(backingStore, value))
            {
                return false;
            }
            backingStore = value;
            onChanged?.Invoke();
            OnPropertyChanged(propertyName);
            return true;
        }

        public event PropertyChangedEventHandler PropertyChanged;
        protected virtual void OnPropertyChanged([CallerMemberName] string propertyName = "")
        {
            var changes = PropertyChanged;
            if (changes == null)
            {
                return;
            }
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}

using Microsoft.Extensions.Logging;

namespace MauiApp1;

public static class MauiProgram
{
            public static MauiApp CreateMauiApp()
            {
                        var builder = MauiApp.CreateBuilder();
                        builder
                                    .UseMauiApp<App>()
                                    .ConfigureFonts(fonts =>
                                    {
                                                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                                                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
                                    });

#if DEBUG
```

```
            builder.Logging.AddDebug();
#endif

            return builder.Build();
        }
}
```

DATABASE QUERY

```
-- Products table
CREATE TABLE products (
    ProductId VARCHAR(40) PRIMARY KEY,
    ProductName VARCHAR(80),
    Price DECIMAL(10,2)
);

-- Customers table (Receiver)
CREATE TABLE customers (
    CustomerId VARCHAR(40) PRIMARY KEY,
    ReceiverName VARCHAR(80)
);

-- DeliveryMethods table
CREATE TABLE delivery_methods (
    DeliveryMethodId VARCHAR(40) PRIMARY KEY,
    MethodName VARCHAR(80)
);

-- PaymentMethods table
CREATE TABLE payment_methods (
    PaymentMethodId VARCHAR(40) PRIMARY KEY,
    MethodName VARCHAR(80)
);

-- Orders table (normalized)
CREATE TABLE orders (
    OrderId VARCHAR(40) PRIMARY KEY,
    ProductId VARCHAR(40),
    CustomerId VARCHAR(40),
    DeliveryMethodId VARCHAR(40),
    PaymentMethodId VARCHAR(40),
    FOREIGN KEY (ProductId) REFERENCES products(ProductId),
    FOREIGN KEY (CustomerId) REFERENCES customers(CustomerId),
    FOREIGN KEY (DeliveryMethodId) REFERENCES
delivery_methods(DeliveryMethodId),
```

```
    FOREIGN KEY (PaymentMethodId) REFERENCES
payment_methods(PaymentMethodId)
);

SELECT
    o.OrderId,
    p.ProductName,
    p.Price,
    c.ReceiverName,
    dm.MethodName AS DeliveryMethod,
    pm.MethodName AS PaymentMethod
FROM orders o
JOIN products p ON o.ProductId = p.ProductId
JOIN customers c ON o.CustomerId = c.CustomerId
JOIN delivery_methods dm ON o.DeliveryMethodId = dm.DeliveryMethodId
JOIN payment_methods pm ON o.PaymentMethodId = pm.PaymentMethodId;

SELECT * FROM products;
SELECT * FROM customers;
SELECT * FROM delivery_methods;
SELECT * FROM payment_methods;
SELECT * FROM orders;
```