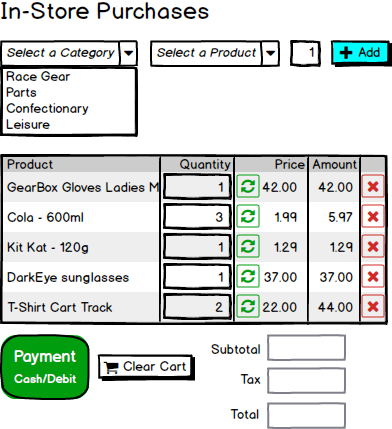
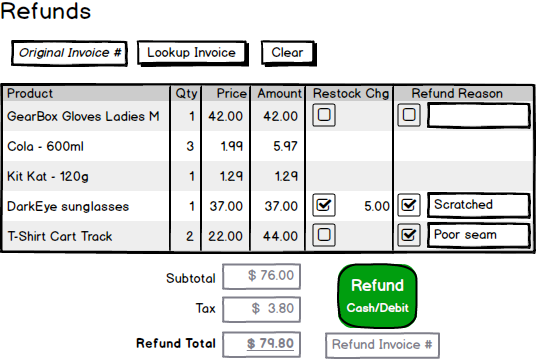
UXDesign Title by Your Name

# ERD for UX process

# Possible UX Interface



# ViewModels:

Place your ViewModel(s) here.

public class CategorySelectionListItem

{

public int CategoryId { get; set; }

public string CategoryText { get; set; }

}

public class ProductSelectionListItem

{

public int ProductId { get; set; }

public string ProductText { get; set; }

}

public class PurchaseListItem

{

public int ProductId { get; set; }

public int InvoiceId { get; set; }

public string ProductDescription { get; set; }

public int Quantity { get; set; }

public double Price { get; set; }

public double RestockCharge { get; set; }

}

public class PurchaseInvoice

{

public int PurchaseInvoiceId { get; set; }

public double SubTotal { get; set; }

public double GST { get; set; }

public double Total { get; set; }

public IEnumerable<PurchaseListItem> PurchaseItems { get; set; }

}

public class RefundInvoice

{

public int RefundInvoiceId { get; set; }

public double SubTotal { get; set; }

public double GST { get; set; }

public double Total { get; set; }

public IEnumerable<RefundListItem> RefundItems { get; set; }

}

public class RefundListItem

{

public int ProductId { get; set; }

public int InvoiceId { get; set; }

public string ProductDescription { get; set; }

public int Quantity { get; set; }

public double Price { get; set; }

public double RestockCharge { get; set; }

public string RefundReason { get; set; }

}

# Event and Wiring Summations

Place your Event and Wiring Summations here.

public List< CategorySelectionListItem> Category\_List()

{

//get all categories and return keyvalue pair list of CategorySelectionListItem’s

}

public List< ProductSelectionListItem > ProductsbyCategoryList(int CategoryId)

{

//get a keyvalue pair list of ProductsSelectionListItem’s by the CategoryId passed from the //category selection DDL on item select.

}

public PurchaseListItem PurchaseListItemByProductId(int ProductId)

{

//retrieve product information to create a PurchaseListItem by ProductId retrieved from

//product selection DDL on add button click.

}

public int CreatePurchaseInvoice(PurchaseInvoice invoice, int EmployeeID)

{

//get sysdate for Invoice insert

//insert record on Invoices Table

//assign all invoice.PurchaseItems the returned InvoiceID

//insert record’s on InvoiceDetails Table from invoice.PurchaseItems

//update QuantityOnHand attribute for products sold.

//return InvoiceID

}

public List<RefundListItem> GetPurchaseInvoiceItemsForRefund(int PurchaseInvoiceID)

{

//get all InvoiceDetails records where InvoiceID == PurchaseInvoiceID

//if confectionary item then do not allow refund.

//get all StoreRefunds records that have already been entered

//where OriginalInvoiceID == PurchaseInvoiceID

//if item in InvoiceDetails results exists in StoreRefunds results, then do not display for refund

//convert remaining InvoiceDetails records into RefundListItem’s

//return List<RefundListItem> to display. Confectionary items not able to be refunded and //previously refunded Items will not be in the list.

}

public int CreateRefundInvoice(RefundInvoice invoice, int EmployeeID)

{

//get sysdate for Invoice insert

//insert invoice object into Invoices table

// assign all invoice.RefundItems the returned InvoiceID

//insert record’s on StoreRefunds from invoice.RefundItems

//update QuantityOnHand attribute for products sold.

//return InvoiceID

}