Receiving

View Models

csharp

```csharp
public class PurchaseOrderView
{
    public int PurchaseOrderID { get; set; }
    public int PurchaseOrderNumber { get; set; }
    public DateTime? OrderDate { get; set; }
    public string VendorName { get; set; }
    public string VendorPhone { get; set; }
}
```

csharp

```csharp
public class ReceivingDetailView
{
    public int PurchaseOrderDetailID { get; set; }
    public int PartID { get; set; }
    public string Description { get; set; }
    public int QtyOnOrder { get; set; }
    public int QtyOutstanding { get; set; }
    public int QtyReceive { get; set; }
    public int QtyReturn { get; set; }
    public string Reason { get; set; }

}
```

Form Base View Models

UnOrderReturns( )

csharp

```csharp
public class UnorderedReturnItemView
{
    public int CartId { get; set; }
    public string Description { get; set; }
    public string VSN { get; set; }
    public int Quantity { get; set; }
}
```

Business Processing Requirements

Query Service Methods

List of outstanding purchase orders

csharp

```csharp
public List<PurchaseOrderView> GetOutstatndingOrder()
```

- Retrieve only Purchase Orders that are not close

Fetch receiving order detail

csharp

```csharp
public List<ReceivingDetailView> Receiving_FetchOrderDetails(int
purchaseOrderID)
```

- create new list of receiving details bases on the purchase order details and current parts information
    - QtyOnOrder
    - QtyOutstanding (QtyOnOrder - (All Receiving QtyReceive))

Command Methods

Receive

csharp

```csharp
public void Receiving_Receive(int PurchaseOrderID, int employedId,
List<ReceivingDetailView> receivingDetails,
List<UnorderedReturnItemView> unorderedReturnItems, string reason)
```

- foreach receiving detail
    - Is received quantity greater than the outstanding order
        - yes
            - errorList.Add(new Exception($"Quantity for receiving item () is greater than the outstanding order"));
    - Received Qty is a positive value
        - no
            - errorList.Add(new Exception($"Quantity for receiving item () must have a positive value"));
    - Return Qty is a positive value
        - no
            - errorList.Add(new Exception($"Return for receiving item () must have a positive value"));
        - yes
            - does it have a reason
                - no
                    - errorList.Add(new Exception($"There must be a reason for the return for receiving item ()"));
- foreach unordered return item
    - is there an item description
        - no
            - errorList.Add(new Exception($"There must be a item description for unorder items));
        - Unorder Qty is a positive value
            - no
                - errorList.Add(new Exception($"Unorder item () must have a positive value"));
- check for errors
    - Yes
        - throw the list of all collected exceptions
    - no
        - save all work to the database
            - create ReceivingOrder
                - PurchaseOrderID
                - ReceiveDate = today
                - EmployeeId
            - create ReceiveOrderDetail for each receivingDetails that has a QtyReceive greater than 0

    - PurchaseOrderDetailID
    - QuantityReceived
    - update Parts
      - QuantityOnHand = QuantityOnHand + QuantityReceived
  - create ReturnOrderDetail for each receivingDetails that has a Return greater than 0
    - ReceivedOrderID
    - PurchaseOrdedrID
    - ItemDescription
    - Quantity
    - Reason
    - VendorPartNumber
  - create ReturnOrderDetail for each unorderedReturnItems that has a Qty greater than 0
    - ReceivedOrderID
    - ItemDescription
    - Quantity
    - Reason
    - VendorPartNumber
  - **Check if all items have been received**
    - yes
      - Update Close

Force Close

```csharp
public void Receiving_ForceClose(int PurchaseOrderID, int employedId
List<ReceivingDetailView> receivingDetails,
List<UnorderedReturnItemView> unorderedReturnItems)
```

Save rules as Receiving_Receive

- On Save
  - Update Close
  - Update Note

Form Methods

Insert Unorder Item

csharp

```csharp
public void InsertUnorderItem()
```

- Add empty row to unorder returns

Delete Unorder Item

csharp

```csharp
public void DeleteUnorderItem()
```

- Remove empty row to unorder returns

Clear Unorder Item

csharp

```csharp
public void ClearUnorderItem()
```

- Clear all values from the current to unorder return row
