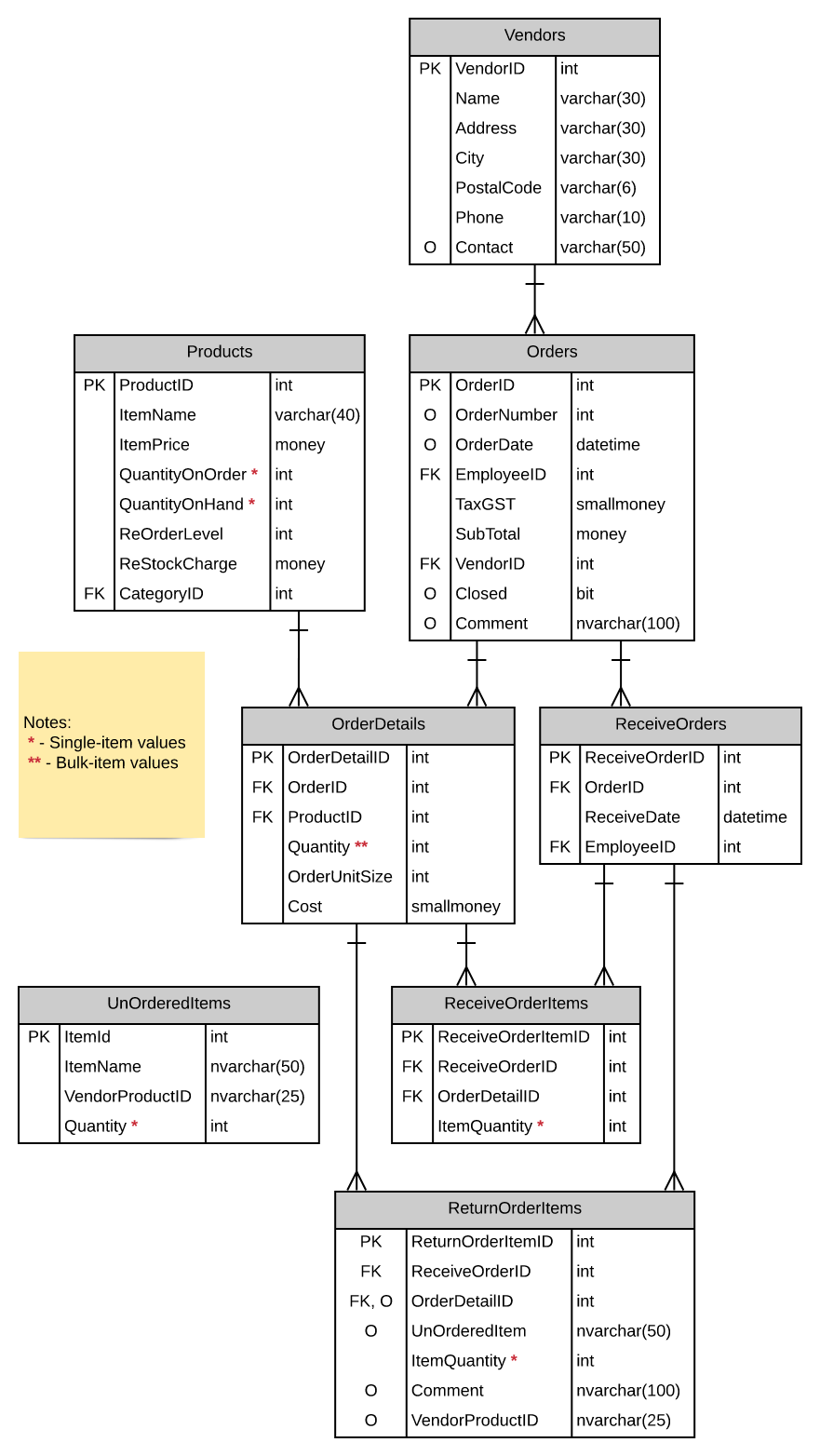
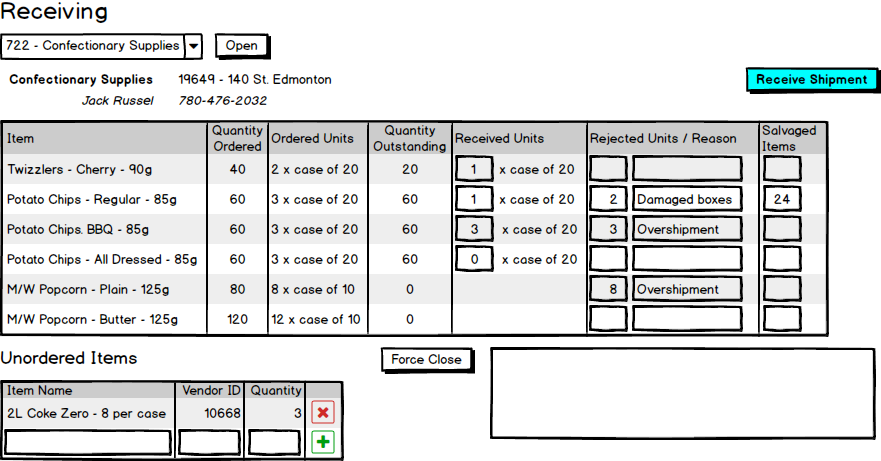
Receiving Solution

# ERD for UI/UX process



Prototype UX Screen





Prototype UX Processing Plan

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| PODDL (**1**) | | None / wired via ObjectDataSource |
| **BBL Class(es) and Method (s)** | | |
| OrdersController | List<OrderVendorList> List\_VendorNamesWithOrderID()  Retrieve a list of names with vendorid of vendor, its name and order number for DDL   * Check if its is an active PO (Closed = false) | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Orders (R), Vendors (R) | | OrderVendorList (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Open (**2**), Vendor Name (**4**), Vendor Address (**5**), Contact (**6**), Phone (**7**), UnOrderedItems (**8**) | | OnClick / wired via ObjectDataSource   * Locates order that was chosen from the VendorDDL |
| **BBL Class(es) and Method (s)** | | |
| ReturnOrderItemsController | List<OrderList> List\_VendorOrdersListSelection()   * Display the vendor’s name, address, phone, contact person.  * Retrieve state of goods for the PO. * Clear the contents of the UnOrderedItems Table.   Retrieve a list of the orders that the active vendor has made  Return ItemName, Quantity \* OrderUnitSize, Quantity “ x cases of ” OrderUnitSize, Quantity – QuantityOnHand, QuantityOnHand “ x cases of ” OrderUnitSize, ItemQuantity, comment, and OrderUnitSize x QuantityOnHold. Display in ReturnOrderItems  RetrieveContact Vendor\_Get(int VendorID)  Retrieve the vendor record matching orderid | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| ReturnOrderItems (R), OrderDetails (R), RecieveOrderDetails (R), Products (R),  Vendors(R) | | RetrieveContact(P), OrderList (P) |

from x in ReceiveOrderItems

// need to include OrderID

where x.OrderDetail.Order.Vendor.VendorID == 9

select new

{

OrderDetailID = x.OrderDetail.OrderDetailID,

Item = x.OrderDetail.Product.ItemName,

QuantityOrdered = x.OrderDetail.Quantity \* x.OrderDetail.OrderUnitSize,

OrderedUnits = x.OrderDetail.Quantity + " x case of " + x.OrderDetail.OrderUnitSize,

QuantityOutstanding = (x.OrderDetail.Quantity \* x.OrderDetail.OrderUnitSize) - x.OrderDetail.Product.QuantityOnHand,

RecievedUnits = x.OrderDetail.Product.QuantityOnHand + " x case of " + x.OrderDetail.OrderUnitSize,

RejectedUnitsReason = x.ItemQuantity + " " + x.ReceiveOrder.Order.Comment,

SalvagedItems = x.OrderDetail.OrderUnitSize - x.OrderDetail.Product.QuantityOnHand}

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Receive Shipment (**9**) | | OnClick / wired via ObjectDataSource |
| **BBL Class(es) and Method (s)** | | |
| ReturnOrderItemsController | List<OrderList> List\_RecieveVendorShipment (String name, string address, string contact, string phone)   * Ensure this shipment was not previously closed * Check validation for table constraints as business rules * Check if order items and return order items must be generated with correct quantities * Check If received quantity is the sum of salvaged and returned * Make a single record in RecieveOrder entry and set RecieveDate to current date and time(sysdate) * Check if outstanding items are zero, if they are close the order (set Closed = true) * Overage does not exceed the order unit size on the original order * Check if the quantity on hand and quantity on order are adjusted correctly (ordered 100, got 50 on hand now, then switch the ordered to 50 then setting ordered to 50) * If underordered items are available, then set a comment as the itemname as well as “ not ordered”, send it to ReturnOrderItems using crud, and clear out the UnOrderItems Table | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| ReturnOrderItems (U), OrderDetails (U), RecieveOrderDetails (U), Products (U), ReceiveOrders(C), ReturnOrderItems(C) | | OrderList (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Plus Sign (**10**), X Sign (**11**) | | OnClick   * Managed in a CRUD Manner * Does not affect anything else |
| **BBL Class(es) and Method (s)** | | |
| UnOrderedItemsController | public void AddandDelete\_ToUnOrderedItemsList()   * Check if you are clicking the + sign or x sign * Add UnorderItems to List * Remove UnOrderedItems that was chosen | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| UnOrderedItems (R, C, D) | |  |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| Force Close (**12**) | | OnClick / wired via ObjectDataSource   * Prompt user in browser if they actually want to close it. |
| **BBL Class(es) and Method (s)** | | |
| ReturnOrderItemsController | List<OrderList> List\_ForceClose (String name, string address, string contact, string phone)   * Check if the reason is filed out. * Set purchase order to close. * Reduce the quantity on order for all products by the remaining outstanding quantites of each item on the purchase order. * Clear out UnorderedItems Table. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| ReturnOrderItems (U), OrderDetails (U), RecieveOrderDetails (U), Products (U),  RecieveOrders (U) | | OrderList (P) |