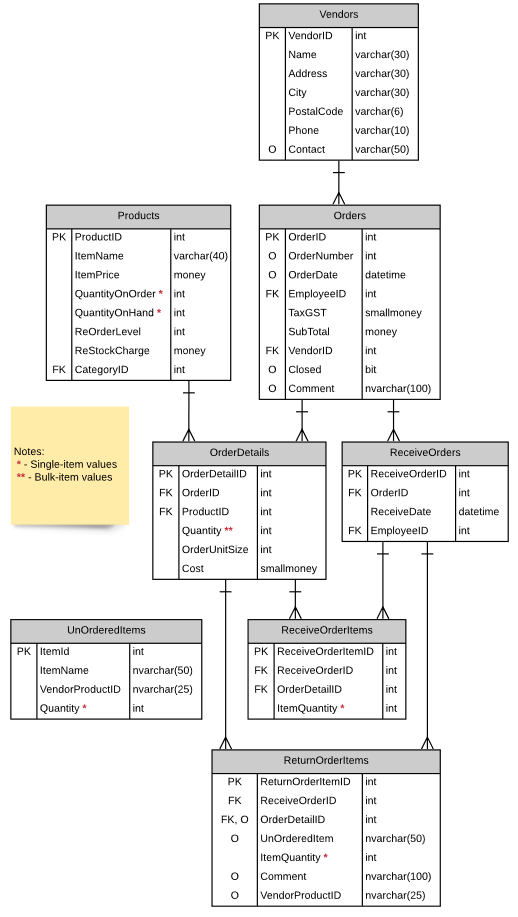
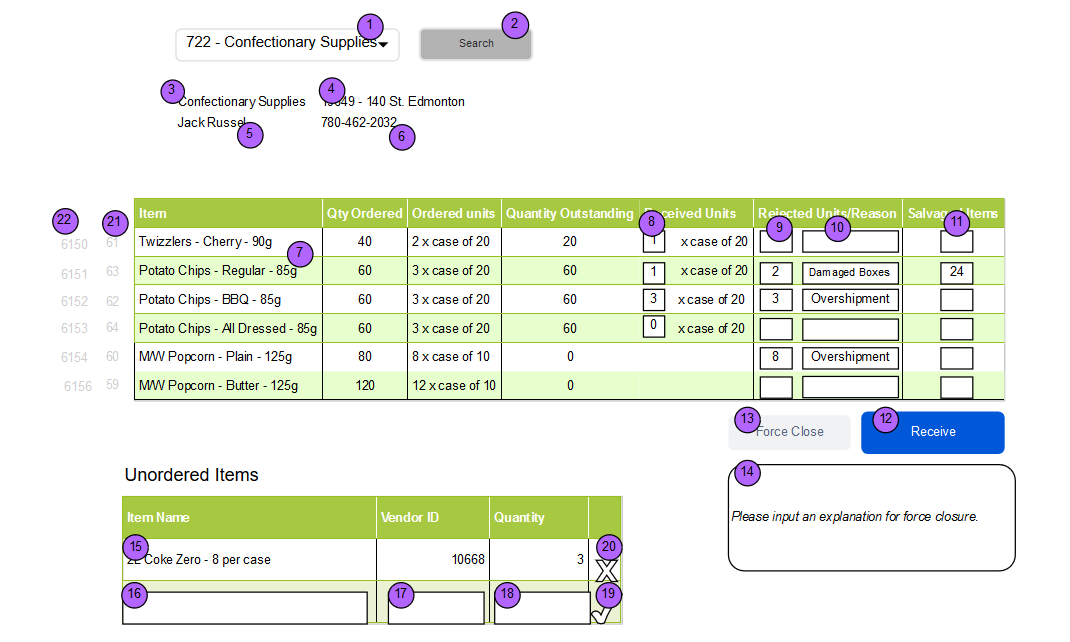
eRace Solution -

Receiving Subsystem



Screen Design



On page load, the dropdown is populated with items containing the OrderID and the name of the associated vendor. When a user selects the chosen Order from the dropdown and clicks **Search**, the vendor information is populated directly below and the OrderDetails of that order are used to populate the left half of the WebApp’s Order Details Table. The UnorderedItems table is also cleared.

In the WebApp’s Order Details Table, ‘Quantity Outstanding’ is derived from subtracting all previous ReceiveOrderItems quantities associated with that OrderDetailID from the OrderDetail Quantity. The user is then able to either a) put all information about the shipment they have received in the table and track the unordered items in the UnOrderedItems table below or b) close out the aged order, with a required explanation for closure. In both cases the UnOrderedItems table is cleared.

If the user presses the **Receive** button, all information entered on the OrderDetails table is validated and captured in a collection, then sent to the database to be persisted. In a transaction, 1) a new record is created for that received shipment in the ReceiveOrders Table, 2) individual line item records are created in the ReceiveOrderItems table, 3) rejected/unordered item records are created in the ReturnOrderItems table, 4)the quantity on order and quantity on hand records in the Product table are updated. If the total quantity ordered from the OrderDetails records are all satisfied, the Order is closed and the Order comment is set to “Order Complete.”

If the user chooses to **Force Close**, Order is closed with user-defined comment, and quantity on order records in the product table are reduced by the outstanding amounts in a transaction

Well done

# Event and Wiring Summations

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| OpenOrderDDL (1) | | None / wired via ObjectDataSource |
| **BBL Class(es) and Method (s)** | | |
| OrderController | List<OpenOrder> List\_OpenOrder()   * Retrieve a list of OpenOrder POCO Where OrderID exists, and OrderDate exists, and Closed == false | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Orders (R), Vendors (R) | | OpenOrder (P) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| SearchButton (2), VendorName(3), VendorAddress(4), VendorContact(5), VendorPhone(6), OpenOrderDetailTable(7)  ProductItemId(21 – hidden)  OpenOrderDetailItemId(22 – hidden) | | OnClick(SearchButton)/wired via ODS   * Send VendorID to Vendor\_FindById(int vendorid) to find Vendor object. * Populate 3, 4, 5, 6 with Vendor information * Run UnOrderedItems\_ClearTable() * Send OrderID to OrderDetail\_FindOpenOrderDetail(int orderid) to generate list of OpenOrderDetail POCO. * Bind OpenOrderDetailTable(7) with each item from List<OpenOrderDetail> (Item, Qty Ordered, Ordered Units, Quantity Outstanding, Static Text in Received Unit column) , ProductItemId(21 – hidden),OpenOrderDetailItemId(22 – hidden) * Refresh display |
| **BBL Class(es) and Method (s)** | | |
| VendorController  UnOrderedItemsController  OrderDetailController | Vendor Vendor\_FindById(int vendorid)   * Retrieve Vendor object by ID.   Void UnOrderedItems\_ClearTable()   * Clear all existing entries in the context UnOrderedItems table   List<OpenOrderDetail> OrderDetail\_FindOpenOrderDetail(int orderid)   * Generate list of OpenOrderDetail POCO for WebApp Detail Table | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Vendors (R), OrderDetails (R), ReceiveOrderItems (R), Products (R)  OrderDetails (D), UnOrderedItems (D) | | OpenOrderDetai l(P), Vendor (E),  UnOrderedItem (E) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| ForceCloseButton (13), CloseText(14), OpenOrderDDL (1) | | OnClick(ForceCloseButton)   * Validate required input and collect string from (14) * Prompt User to confirm close. * Collect OrderID from 1   ForceClose\_Order(Int OrderId, string reason)   * Refresh Display |
| **BBL Class(es) and Method (s)** | | |
| OrderController | Void ForceClose\_Order(Int OrderId, string reason)  TRX   * Find Order by orderid * Validate order is not closed * Set Order.Closed to true * Set Order.Comment to reason string * Find outstanding product balances related to that order ID using ReceiveOrderItem and OrderDetail Table * Find and decrement decrement QuantityOnOrder of the appropriate <Product> collection items by corresponding outstanding balances related to that OrderID * If they exist, remove items from the UnOrderedItems collection. * Save transaction | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| Products(U), Orders(U), UnOrderedItems (D) | | Product (E), Order (E), UnOrderedItem(E) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| ReceiveButton(12)  ReceiveUnit(8), RejectUnit(9), RejectReason(10), SalvageUnit(11)  ProductItemId(21 – hidden)  OpenOrderDetailItemId(22 – hidden) | | OnClick(ReceiveButton)   * Collect Employee ID from Log-In * CollectOrderID from DDL * Collect, validate, and parse input from 8,9,10,11,21,22 into ReceiveRejectUnitDetail and add to collection of List<ReceiveRejectUnitDetail> * Send to OrderController Receive\_Order method * Clear WebApp UnOderedItem table * Refresh display |
| **BBL Class(es) and Method (s)** | | |
| OrderController | Receive\_Order(int OrderId, int employeeid, List<ReceiveRejectUnitDetail>)  TRX   * Validate order has not been closed * Validate that all ReceiveRejectUnitDetail that have ReceivedUnits > 0 have a ProductID that exists on the original OrderDetail table. * Create new ReceiveOrder record * Set ReceiveOrder.ReceiveDate to current time * Add New ReceiveOrderItem record to ReceiveOrderItem collection for each item in ReceiveRejectUnitDetail that has ReceivedUnits > 0 * Find and Decrement QuantityOnOrder for products in the Product collection by corresponding receive quantity * Add new records to the ReturnOrderItems collection for any ReceiveRejectUnitDetail that have a RejectedUnits > 0 * Retrieve items from UnorderedItems collection * Add UnOrderedItems to ReturnOrderItems collection * Clear Context UnOrderedItems Table * If ALL received quantities meet QuantityOrdered for that OrderID, set Order.Closed to true. * Save transaction | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| ReceiveOrders(C), ReceiveOrderItems(C), ReturnOrderItems(C), UnOrderedItems(D), Products(U), Orders(U - Conditional) | | Product (E), ReceiveOrder (E), ReturnOrderItem (E), UnOderedItem (E), Order (E), UnOrderedItem (E), ReceiveRejectUnitDetail (D) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| AddUnorderedItemButton (19), UnorderedItemName(15), UnOrderedItemVendorProdId(16), UnOrderedItemQuantity(17) | | OnClick(AddUnorderedItemButton)   * Validate and Collect itemname(15), vendorproductid(16), and qty(17) from table. * Send to BLL * Refresh UnorderedItemTable(15) |
| **BBL Class(es) and Method (s)** | | |
| UnOrderedItemsController | void Add\_UnOrderedItem(string itemname, string vendorproductid, int quantity)   * Add UnOrderedItem to the UnOrderedItem table | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| UnOrderedItems (C) | | UnOrderedItem (E) |

|  |  |  |
| --- | --- | --- |
| **Controls** | | **Events** |
| RemoveUnOrderedItemButton (20) | | OnClick(RemoveUnorderedItemButton)   * Collect primary key (hidden) * Send to BLL to remove item. * Refresh UnorderedItemTable(15) |
| **BBL Class(es) and Method (s)** | | |
| UnOrderedItemsController | void Remove\_UnOrderedItem(int unordereditemid)   * Find and remove UnorderedItem from table. | |
| **SQL Table(s): (C,R,U,D)** | | **Entities/DTOs/POCOs** |
| UnOrderedItems (D) | | UnOrderedItem (E) |

|  |
| --- |
| **OpenOrder - POCO** |
| Int **OrderID** = Order.OrderID  String **VendorName** = Vendor.VendorName  Int **VendorID** =Vendor.VendorID |

|  |
| --- |
| **OpenOrderDetail - POCO** |
| int **OrderID** =OrderDetail.OrderID  int **OrderDetailID** = OrderDetail.OrderDetailID  string **Item** = Product.ItemName  int **ProductID** = Product.ProductID  int **QuantityOrdered** = OrderDetail.Quantity \* OrderDetail.OrderUnitSize  string **type** = VendorCatalog.OrderUnitType  int **QuantityOutstanding** = QuantityOrdered - ReceiveOrderItem.ItemQuantity.Sum() |

|  |
| --- |
| **ReceiveRejectUnitDetail - DTO** |
| Int **ProductID**  int **OrderDetailID**  int? **ReceivedUnits** ((Received \* UnitQuantity) + Salvage)  int? **RejectedUnits** ((Rejected \* UnitQuantity) – Salvage)  string? **RejectReason** |