Note: This document serves as a high-level overview and may not cover all rules or key features. Its purpose is to offer additional clarity for you and your team. If you have any questions, please see your instructor.

Sales and Returns Rules

Access & Authentication:

- Only for in-store events.
- Users must log in.
- Access restricted to those in the `SalesReturns Role`.
- Sales and/or returns handled by an Associate or Store Manager.
- Employee's full name should be displayed on the form.

Sales:

In-Store Shopping:

- Customers bring items to the counter.

- Employees record items in a cart associated with their till.
- Actions in the cart:
    - Add and remove items.
    - Edit quantities.
    - Refresh item subtotals.
- On payment, the cart is processed as a sale.
- A `Cancel` button must always be present.

Business Rules:

- Only one cart per sale, but multiple items allowed.
- No duplicate items in the cart.
- Change item quantities in `View Cart` or automatically increment with `Add from Shopping` .
- Don't display or allow adding discontinued products.
- Quantity (whole number > 0) must be specified when adding.
- Added items go to the shopping cart.
- After adding an item, a confirmation message is shown.
- [Optional] Show the number of items in the cart with `View Cart` .

Checkout:

- Two phases: `View Cart` & `Place Order` .

- In `View Cart`:
  - List items in the cart.
  - Change quantities or remove items.
  - `Continue Shopping` button for more shopping.
  - `Checkout` button for payment.
- In `Place Order`:
  - Enter and apply coupons.
  - Payment types: 'M' (Money), 'C' (Credit), 'D' (Debit).
  - Display: Subtotal, GST, Discount (percentage), Total.
  - On placing the order, a complete server transaction must:
    - Create a sale based on cart info.
    - Adjust stock quantities without going negative.
    - Display the sale ID.
  - No changes allowed post-payment.

Refunds/Returns:

- Need original receipt.
- Require a return reason for each item.

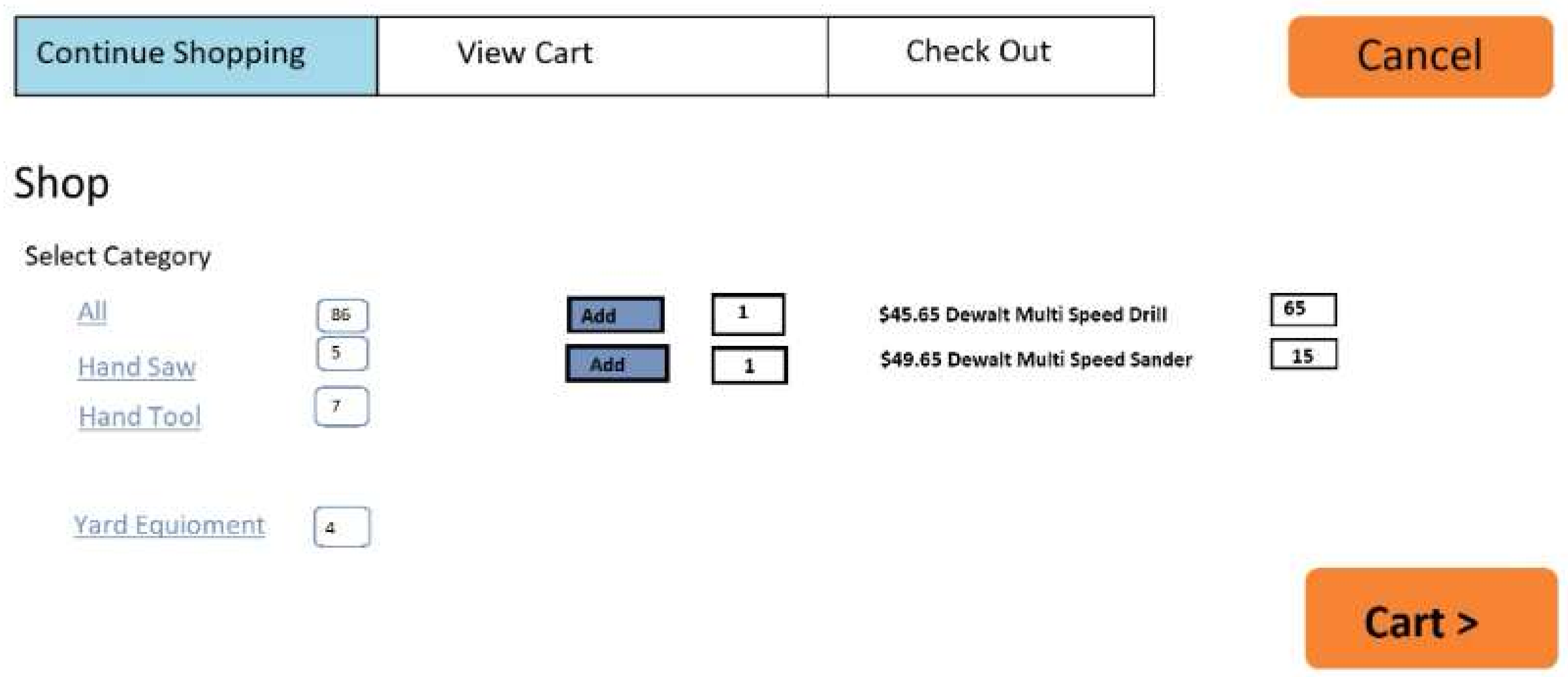
Refunds Rules:

- Original receipt number (SaleID) needed for processing.
- Successful refunds generate `SalesRefund` and `SalesRefundDetail` records.
- Allow individual and partial item refunds.
- Refunded price equals original sale price.
- If a sale had a discount, apply it to the refund.
- Refund processes as a single complete transaction.
- Display the refund's `Sale Refund` number post-refund.
- Refunds can't be edited once processed.
- Update in-stock quantities for refunds, but set the product aside for manager inspection.
- A `Clear` button should clear the screen.

Images (Samples)

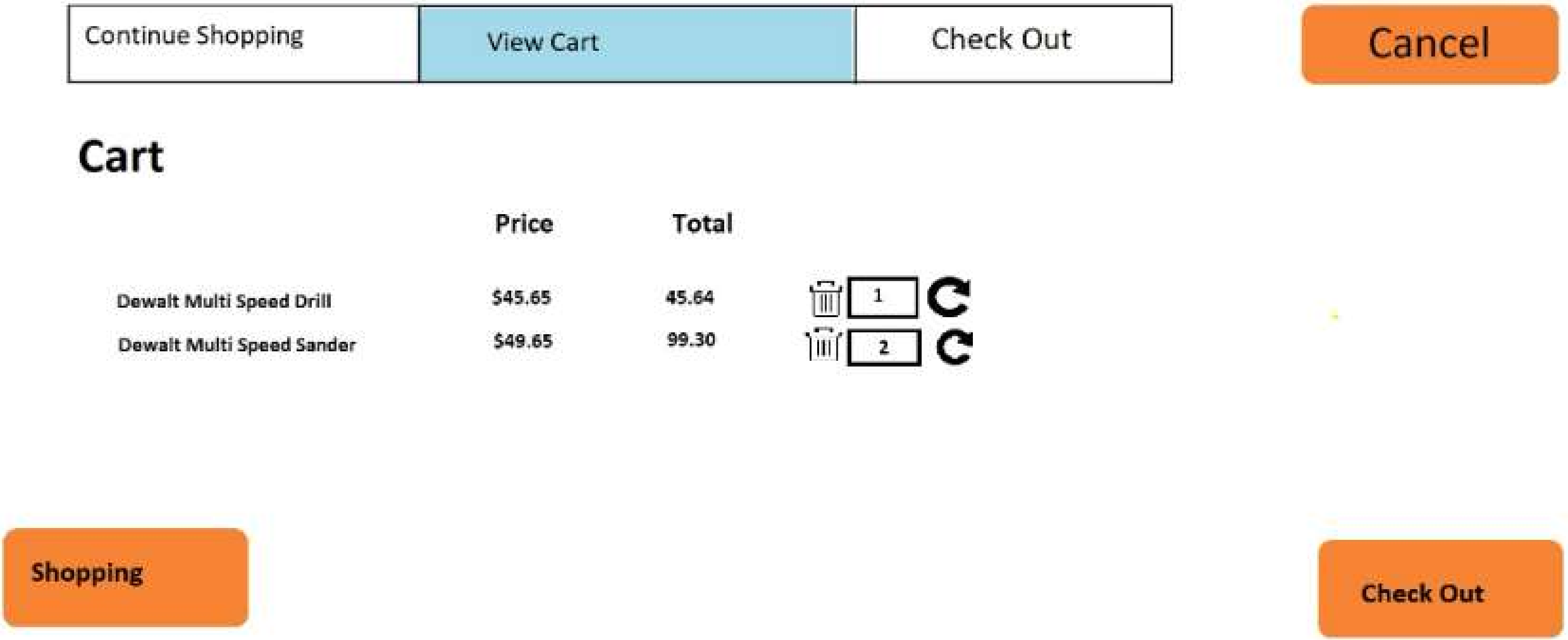
Shopping Cart

Shopping



Checkout

Cart View

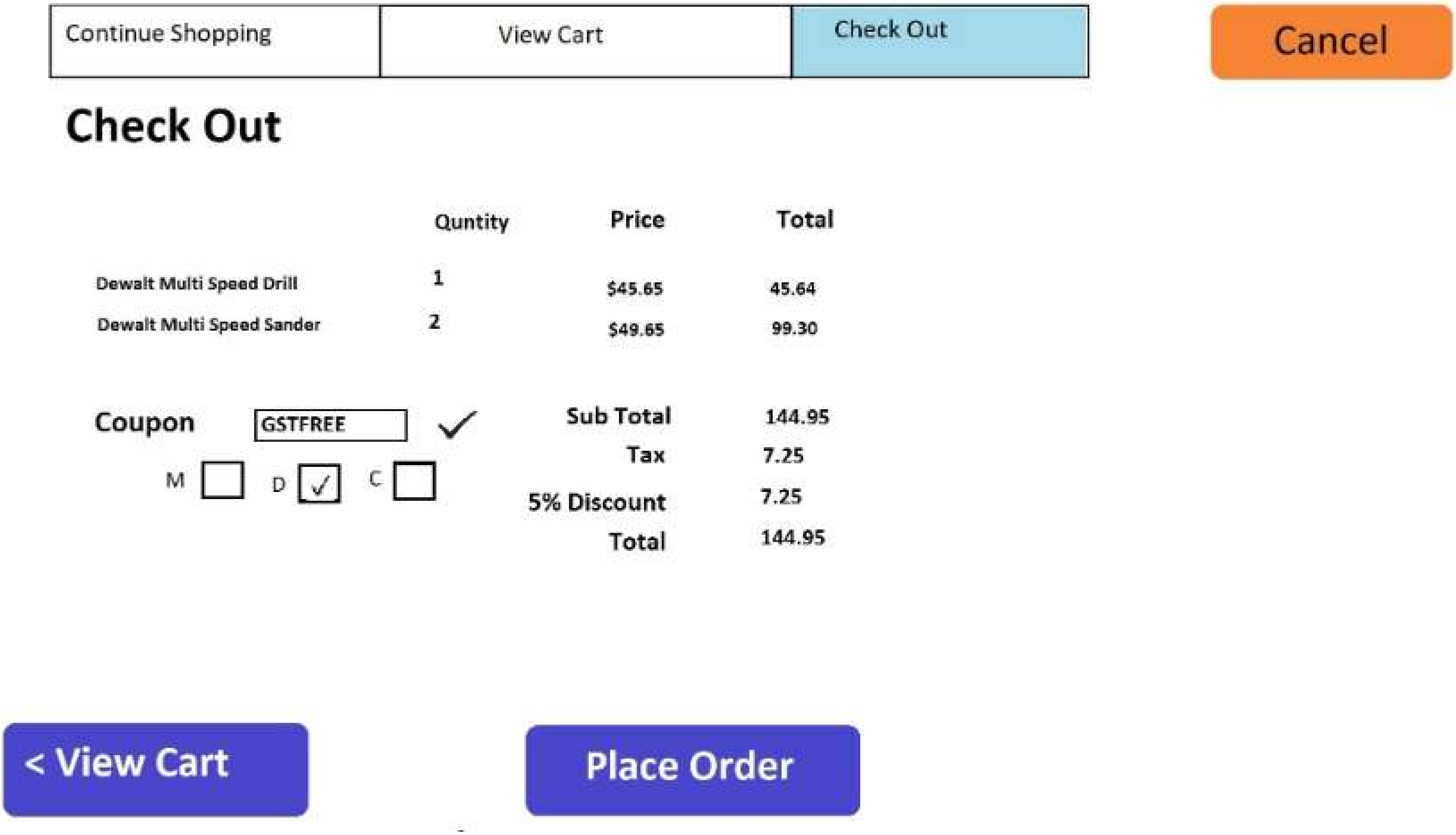


Cart

	Price	Total			
Dewalt Multi Speed Drill	$45.65	45.64	🗑	1	↻
Dewalt Multi Speed Sander	$49.65	99.30	🗑	2	↻

Checkout

Check Out



Check Out

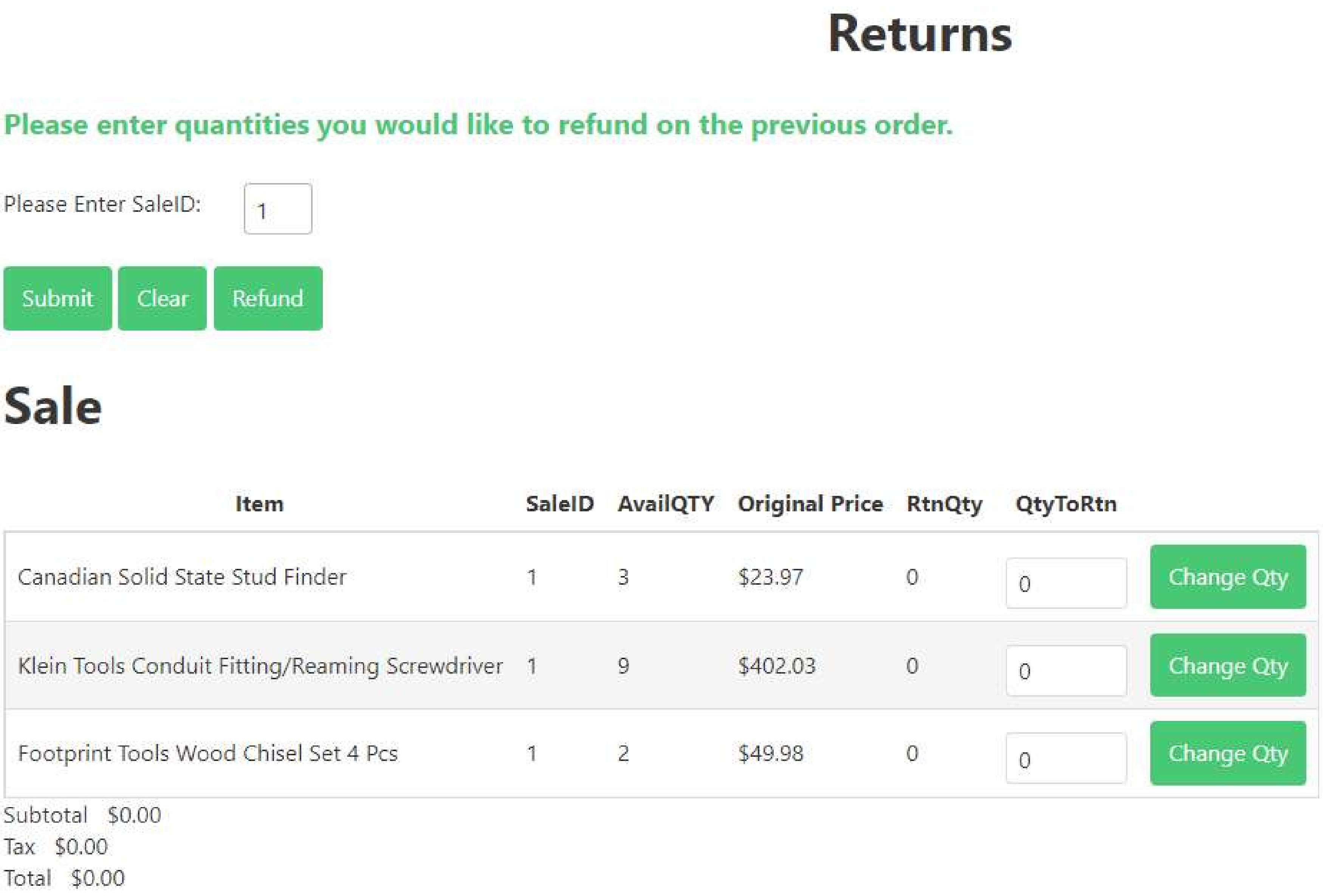
	Quntity	Price	Total
Dewalt Multi Speed Drill	1	$45.65	45.64
Dewalt Multi Speed Sander	2	$49.65	99.30

Coupon GSTFREE ✓

M □ D ☑ C □

Sub Total	144.95
Tax	7.25
5% Discount	7.25
Total	144.95

Refund



Returns

Please enter quantities you would like to refund on the previous order.

Please Enter SaleID: 1

Sale

Item	SaleID	AvailQTY	Original Price	RtnQty	QtyToRtn	
Canadian Solid State Stud Finder	1	3	$23.97	0	0	Change Qty
Klein Tools Conduit Fitting/Reaming Screwdriver	1	9	$402.03	0	0	Change Qty
Footprint Tools Wood Chisel Set 4 Pcs	1	2	$49.98	0	0	Change Qty

Subtotal $0.00
Tax $0.00
Total $0.00

View Models

View models and methods are used as examples, and you are welcome to modify them as needed in your coding example. Please include comments to explain any modifications or choices you make.

```csharp
public class CategoryView
{
    public int CategoryID { get; set; }
    public string Description { get; set; }
    public int StockItemCount { get; set; }
}
```

```csharp
public class StockItemView
{
    public int StockItemID { get; set; }
    public decimal SellingPrice { get; set; }
    public string Description { get; set; }
    public int QuantityOnHand { get; set; }
}
```

```csharp
public class ShoppingCartView
{
    public int StockItemID { get; set; }
    public string Description { get; set; }
    public int Quantity { get; set; }
    public decimal SellingPrice { get; set; }
}
```

```csharp
public class SaleView
{
    public int SaleID { get; set; }
    public int EmployeeID { get; set; }
    public string PaymentType { get; set; }
    public decimal TaxAmount { get; set; }
    public decimal SubTotal { get; set; }
    public int CouponID { get; set; }
    public List<ShoppingCartView> Items { get; set; } = new(); //  Nov 15
}
```

Refund Models

```csharp
public class ReturnSaleView
{
    public int SaleID { get; set; }
    public int EmployeeID { get; set; }
    public decimal TaxAmount { get; set; }
    public decimal SubTotal { get; set; }
    public int CouponID { get; set; }
    public List<ReturnSaleDetailCartView> ReturnSaleDetails { get; set; } = new(); //  Nov 15
}
```

```csharp
public class ReturnSaleDetailCartView
{
    public int StockItemID { get; set; }
    public string Description { get; set; }
    public int OrginialQty { get; set; }
    public decimal SellingPrice { get; set; }
    public int PreviouReturnQty { get; set; }
    public int QtyReturnNow { get; set; }
}
```

```csharp
public class CouponView //  Nov 15
{
    public int CouponID { get; set; }
    public int Discount { get; set; }
    public string Message { get; set; }
}
```

Methods

```csharp
// Required for Deliverable #1
public List<CategoryView> GetCategories()
public List<StockItemView> GetItemsByCategoryID(int categoryid)
public int GetCoupon(string couponName) //  Nov 15
public int SaveSales(SaleView sale)
public ReturnSaleView GetSaleRefund(int saleID)
public int SaveRefund(ReturnSaleView refundSale)
```