Inventory Reording System

```
using System;
using System.Collections.Generic;

//Inventory items properties
public class InventoryItem
{
    public string item_id { get; set; }
    public int current_stock { get; set; }
    public int forecasted_demand { get; set; }
    public decimal reorder_cost_per_unit { get; set; }
    public int reorder_batch_size { get; set; }
}

//Reorder plan properties
public class ReorderPlan
{
    public string Item_id { get; set; }
    public int units_to_order { get; set; }
}

//Warehouse inventory
public class WarehouseInventory
{
    public List<ReorderPlan> ReorderingPlan_Cal(List<InventoryItem> items)
    {
        var reorderPlans = new List<ReorderPlan>();

        foreach (var item in items)
        {
            // Calculation
            int shortage = Math.Max(item.forecasted_demand - item.current_stock, 0);

            // if shortage,calculate how many units to order
            if (shortage > 0)
            {
                // Calculate number of batches to order
                int batchesNeeded = (int)Math.Ceiling((double)shortage /
item.reorder_batch_size);
                int unitsToOrder = batchesNeeded * item.reorder_batch_size;

                reorderPlans.Add(new ReorderPlan
                {
                    Item_id = item.item_id,
                    units_to_order = unitsToOrder
                });
            }
        }

        return reorderPlans;
    }

    public static void Main(string[] args)
    {
        var items = new List<InventoryItem>
        {
            new InventoryItem
            { item_id = "HM0123", current_stock = 35, forecasted_demand = 50,
reorder_cost_per_unit = 5.0m, reorder_batch_size = 60 },
            new InventoryItem
            { item_id = "NM078", current_stock = 10, forecasted_demand = 90,
reorder_cost_per_unit = 10.0m, reorder_batch_size = 30 },
            new InventoryItem
            { item_id = "LA0555", current_stock = 80, forecasted_demand = 110,
reorder_cost_per_unit = 8.0m, reorder_batch_size = 55 }
        };

        var warehouseInventory = new WarehouseInventory();
        var reorderPlan = warehouseInventory.ReorderingPlan_Cal(items);

        Console.WriteLine("Reordering Plan are:");
        foreach (var plan in reorderPlan)
        {
            Console.WriteLine($"Item ID: {plan.Item_id}, Units to Order: 
{plan.units_to_order}");
        }
    }
}
```

Code Explaination

Created 3 classes - InventoryItem Class,ReorderPlan Class & WarehouseInventory Class

1. InventoryItem Class
   Properties: item_id, current_stock, forecasted_demand, reorder_cost_per_unit, and reorder_batch_size.

2. ReorderPlan Class - for each item
   Properties : item_id & units_to_order

3. WarehouseInventory Class
   Main method ReorderingPlan_Cal() which calculates the reordering plan
   =>Checking the shortage between the forecasted demand and current stock.
     If current_stock >= forecasted_demand, no reorder is necessary.
     If current_stock < forecasted_demand, calculate the stortage:
     shortage = forecasted_demand - current_stock

   =>Calculating the number of batches needed to order.
     If shortage, then calculate the number of batches required : batches_needed = ceil(shortage / reorder_batch_size)

   =>Returning a list of ReorderPlan objects, each containing the Item_id and units_to_order.
     The total number of units to order : units_to_order = batches_needed * reorder_batch_size
     Reorder batch size is fixed.

   Main Method:

  =>The output is printed to the console application.

....
....

Flowchart

```
      -------------
     |    Start    |
      -------------
           |
           v
 --------------------------
| For each item in the list |
 --------------------------
           |
           v
 ---------------------------------
|Calculate stortage:              |
|shortage = max(forecasted_demand |
| - current_stock, 0)             |
 ----------------------------------
           |
           v
 ---------------------------------------------------
| If shortage > 0, calculate batches needed:   |
| batches_needed = ceil(stortage / batch_size) |
 ----------------------------------------------
           |
           v
 ----------------------------------------------
| units_to_order = batches_needed * batch_size |
 ----------------------------------------------
           |
           v
 ----------------------------------------
| (ItemId, UnitsToOrder) to reorder plan |
 ----------------------------------------
           |
           v
 -----------------------------
| End and return reorder plan |
 -----------------------------
```

....
....

Sample Run

```
Reordering Plan are:
Item ID: HM0123, Units to Order: 60
Item ID: NM078, Units to Order: 90
Item ID: LA0555, Units to Order: 55
```