Make a solution in OutSystem that solves the following problem.

A company sells hundreds of products on-line and people place orders from all over the world, just like eBay. Each item has a different weight and price.

And each order can be a combination of different number of items. Now each of these orders are to be put into different packages and sent to the courier company for delivery.

But there are certain rules while splitting items into packages, they are as below:

1. If the cost of all the items in an order is more than $250, split those items into multiple packages, otherwise one package would be enough.

2. If the items in the same order are split into multiple packages, then the weight of the packages should be equally distributed over the packages to save courier charges.

3. While splitting, NO PACKAGE can have a total price equal or above $250

Create a page (You may want to use UI Pattern) which has the following:

1. A simple vertical list of items (in the format Name - price - weight), with a check box next to item
2. A button saying "Place order"

When the user clicks on "Place order", the selected items should be submitted to the same page without refreshing the page and the above splitting rules should be applied to divide this particular order into multiple packages, and display the output result on the same page. Below is a sample output on how it should look like:

This order has following packages:

Package 1

Items - Item 1, Item 3, item 7
Total weight - 510g
Total price - $240
Courier price - $15

Package 2

Items - Item 4, Item 6, item 2
Total weight - 530g
Total price - $160
Courier price - $15

Note: Items and courier prices are in the Test_info.xls attached, you may want to use OutSystem, C# or Java, Its default DB (MSSQL), You may want to split the backend code as API and consume, data transition in JSON format. The code should be split between UI and backend (API). The application will be marked based on the specifically comments (where required), formatting, naming conventions, functionality, modularity, coding standard, the approach taken on best practice/plugins taken on both front end and backend.