This project was created in response to the following prompt:

**Using** [**ASP.NET**](http://ASP.NET) **and a SQL express or a LocalDB instance, you will create a web-based application to create and retrieve grocery information for a mock grocery store. The grocery store name is “Come Get Some Food” and the owner, Billy, has asked for an easy way for his employees to enter inventory so that he can keep track of it from home. He’ll need to know what the item is (description), SKU, brand, date received, quantity received  and current quantity on hand. Please create a simple login mechanism (you can use a built-in library if you’d like), to limit 2 types of users: Owner and Employee. The Owner can Read/Write and Delete data but the Employee can only Read/Write (no delete).**

**“Corporate” requires that the store give access to inventory via a web service. Currently, they support REST or WCF, so it’s up to you. You’ll need to give them 1 API call to give them all inventory by date received and that should return a JSON object containing all DB information for that product. For time purposes, this web service API may be unsecured (no auth).**

**You may use whichever flavor of** [**ASP.NET**](http://ASP.NET) **you wish, along with any naming conventions or 3rd party libraries.**

The administrator login for the app is:

Username: admin

Password: Passw0rd

The web app allows the user to view current inventory, add new items, and edit existing items.  The admin (and any other users added with a role of "Owner") is also able to delete items and add new users.  I have also added functionality that will generate an order based upon current inventory levels and set order thresholds.  In a real-world application functionality could be implemented to allow this order to be sent electronically to a supplier, but in my application the order is generated as a text file and is displayed for the user with the option to print.  The user can view the order history and close outstanding orders (when the hypothetical order arrives).  Closing the order will update the inventory quantities of the order items.  The user may also close an order without updating the inventory (if the order is cancelled, for example).

An API has been implemented that exposes a 'GetInventory()' method with 2 overloads.  The first takes no arguments and will return all inventory items as a JSON string.  The 2nd overload takes a date (as 3 in arguments for month, day, and year) and returns all inventory items that have been last received on or after the supplied date.  The method can be called by:

/api/GetInventory  (all inventory)

/api/GetInventory/{month}/{day}/{year}  (all inventory received on or after supplied date)