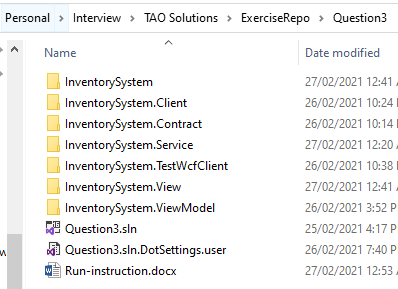
Dear Sir / Madam,

First, thank you for showing interest in running my solution for Question 3 in this coding excise.

I named the solution Question3, every projects inside starts with InventorySystem. A full build of Question3.sln should build everything.

I use <Project Sdk="Microsoft.NET.Sdk"> project format for easier governance. It might mean that in order to compile the solution, you need to install .NET Framework 4.8 SDK, in addition to the normal development targeting pack.



The project InventorySystem is the main WPF application, which can be run as any desktop interactive user. You can run more than one instance of this application.

The InventorySystem.View and InventorySystem.ViewModel projects are logical separations of the WPF application.

The project InventorySystem.Service is the simulated server application written in WCF. It holds in-memory data, which gets reset to the same value every time you restart it.

The service requires Administrator privilege to run. And seeing the console window persisting is the sign of it being running.

The project InventorySystem.TestWcfClient is a console application. It has a loop which checks, retrieves, and updates the inventory periodically. It can be used to test whether the WPF application responds to data changes. It is also used to test the service connectivity, in case there is fire wall or duplicated ports.

The InventorySystem.Contract coordinates the WCF contract and contains the required data contracts.

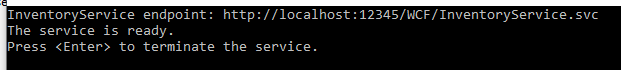
The InventorySystem.Client implements auto reconnectable WCF client to the inventory service. It is the key piece used by the WPF application to reach the WCF service.

At the moment, between the WPF application and the WCF service, the service endpoint URI is hardcoded.

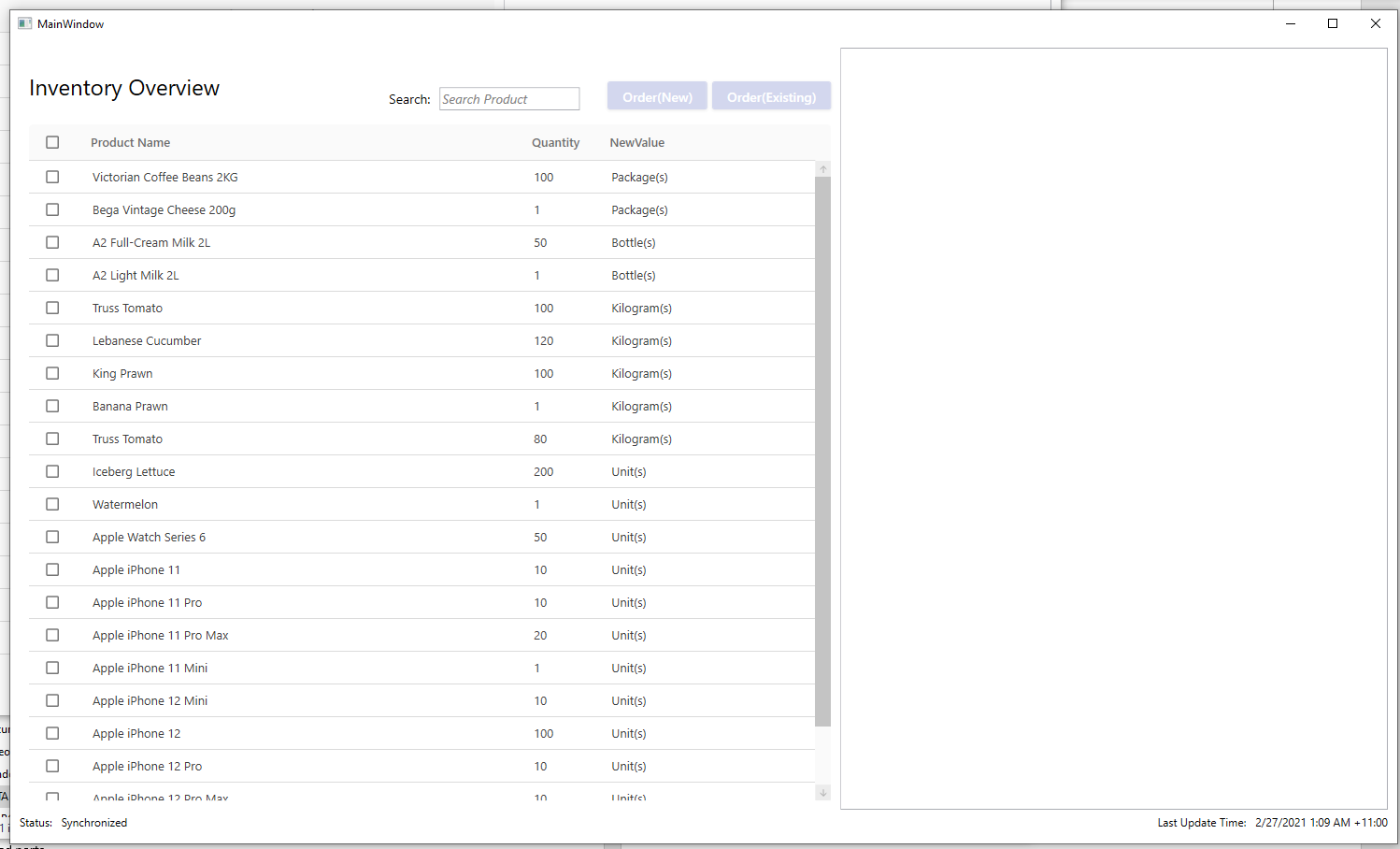
Every projects target .NET Framework 4.8, because this is the last Framework version which still has WCF server modules.

Steps to test:

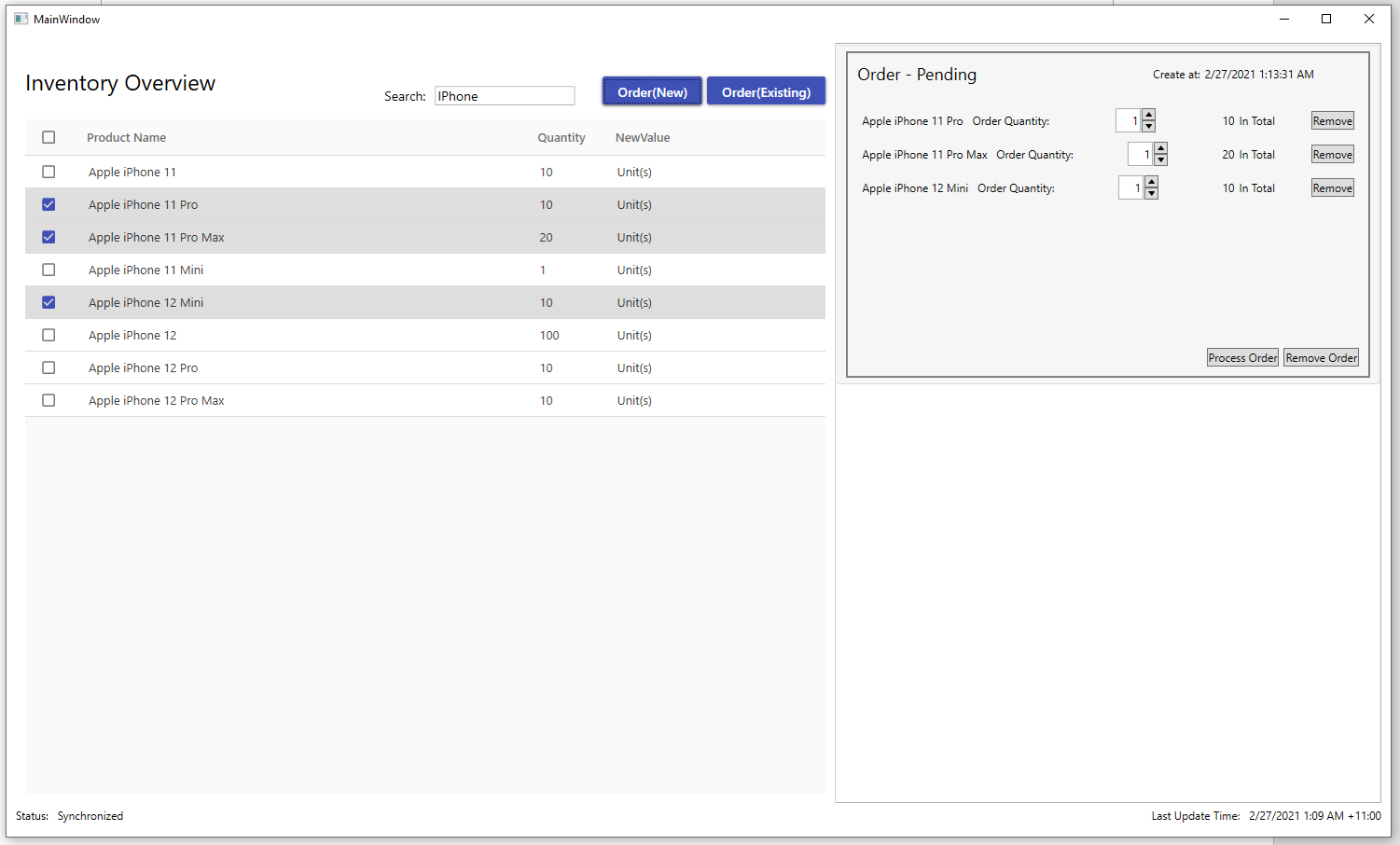
1. After having compiled the whole solution, first, run InventorySystem.Service.exe as Administrator.



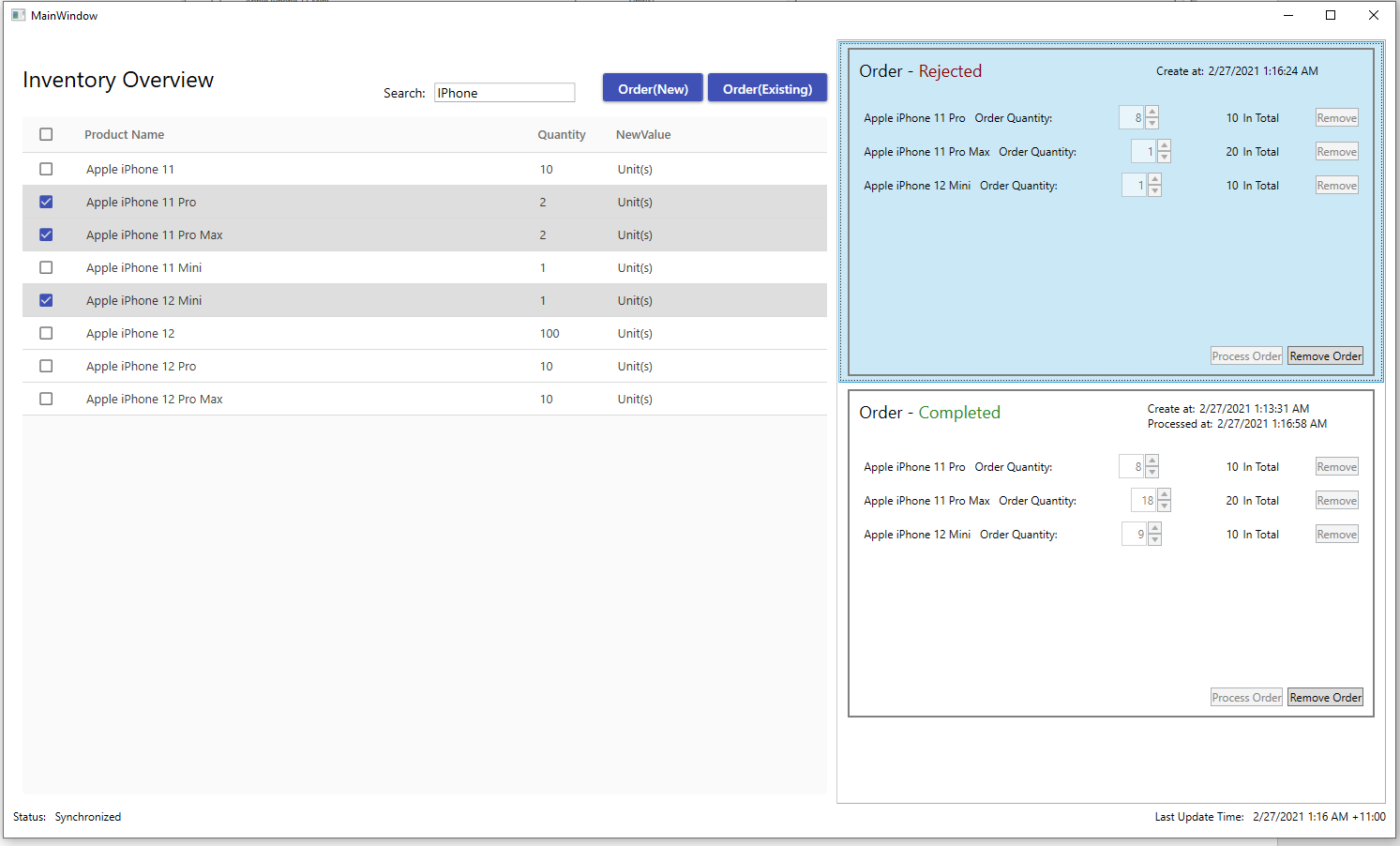
1. Then, run InventorySystem.exe (two instance or more is ok).



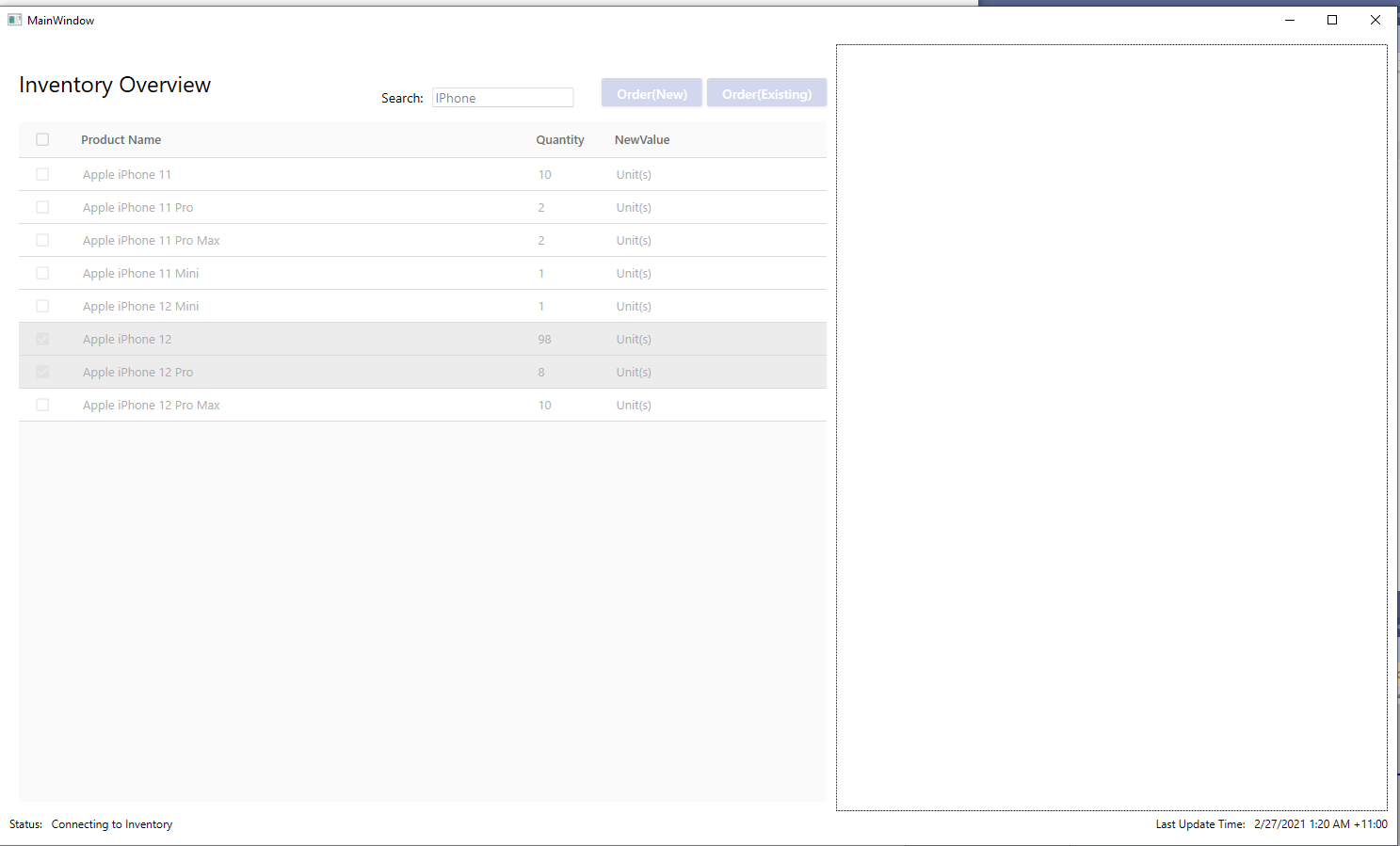
1. Then select any product (hold shift or drag mouse to select more), the Order buttons will be enabled. One of which always put product orders into a new order card, while the other would try adding to a selected order card first, or create a new one if none is found.



1. The Order Card contains workflow to let you finish processing the order with various product quantities. Two cards can operate simultaneously. If both of them are challenging a product with a rather low quantity, one of the operations may fail. The order processing is all or nothing. So having one product having insufficient quantity would lead to none of the product quantity being reduced on server side.



1. Once the server is offline or there is a connectivity issue, The application will try reconnecting regularly (but not aggressively).



1. And once the server is detected again, the view will recover to its normal business logic.
2. Across multiple instance of WPF application. Order Cards are not synchronized, but the Inventory Overviews do synchronize.
3. Due to the restriction of WCF technology, I cannot make the server update instantly show up on the WPF application. I believe doing so with SignalR broadcasting would help, but that would require ASP.NET Core 2.1 +, which may cause some compiling issue on your side due to SDK version issues. So I ended up chose WCF as the service technology.
4. As a result, there is a slight window of out-of-sync right after an order process but before a next scheduled sync.
5. Due to time constraint and limited UI resources (lots of my usual library are company assets, thus not suitable for this exercise). I had to either hack up some views and themes, or use whatever free 3rd party libraries, which sometimes do collide with each other.

I hope you understand the the application has rather ugly visual effect (especially nowadays the bar of UI quality is so high).

I am certain there are lots of imperfection in my solution. Please feel free to provide me constructive feedbacks, and let me say “Thank you!” in advance.

Yours Sincerely

Jiping Zhu (James)