

Blockchain Coding Challenge

# Blockchain Coding Challenge

This coding test is designed to demonstrate understating of blockchain development with Ethereum. There is no time limit.

**\*\*There is no requirement to publish any contracts onto the Ethereum main net. Local is fine\*\***

# Expected

1. Demonstrate a good general understanding of developing blockchain solutions, the tools that are used, and best practices for blockchain specific development.
2. Make use of best practices, such as SOLID principles, design patters, TDD, or BDD.
3. This solution should be returned in full

# Problem Domain

Acme Corporation makes widgets. These widgets are produced in factories, stored in warehouses, and are shipped in batches to their customers.

Acme Corp recognises that blockchain is an excellent tool to model their supply chain, and would like to build a solution on the Ethereum network to model their supply chain.

The widgets that Acme Corp produce are indistinguishable from each other – they are fungible.

## Users

There are two user classes in the system

* **Warehouse managers** are responsible for stock keeping in the Acme Corp factories. They will interact with the system to do tasks such as
  + Update the inventories of the stock
  + Approve customer orders
* **Customers** will interact with the system by placing orders for widgets, and transferring payment in native blockchain tokens

## User Stories

The provided system should facilitate the following user stories

### Inventory Management

Inventory management in the warehouses is a key part of the system, and the warehouse manager users should be able to manage the amount of stock that they have

Feature: Warehouse manager can add stock to the inventory

Given that the user is a warehouse manager

Given that the factory has produced the stock

When the user inputs the amount of stock in the factory

Then the amount of stock in the factory is updated



Feature: Customer can NOT alter stock in inventory

Given that the user is a customer

When the user changes the amount of stock in the factory

Then the amount of stock in the factory is NOT updated

### Customer purchase

A customer should be able to place an order for a certain number of widgets, if the warehouse has sufficient stock

Feature: Customer can place an order

Given that the user is a customer

Given that the warehouse has enough widgets to fulfil the customer order

Given that the customer has enough funds to pay for the order

Then the order is accepted

Feature: Customer can place an order

Given that the user is a customer

Given that the warehouse does NOT have enough stock

Then the order is rejected

Feature: Warehouse manager can ship a customer order.

Given that the user is a warehouse manager

Given that a customer order has been accepted

Then that order can be flagged as shipped

And the amount of stock in the warehouse is updated

# Suggestions

* This solutions should take advantage of common blockchain development tools.
* Appropriate testing should be added to ensure that the above user stories are correctly implemented
* User interaction (transactions) can be initiated either from a UI, or by an API call, or from a script.
* There is no requirement to manage private keys at this stage. Local test networks with unlocked accounts are acceptable.