1. **Define a problem:**

Licensing issues of ERP and security issues at small and medium business partners discourages the sharing of ERP clients/portals with business partner.

In such cases web-based portal is provided to business partner which on completion of transactions pushes some data to ERP further from there some other data could be maintained to complete circle of transactions.

Another approach is to use QR code systems from government IRN portals which involves scanning and is hardware dependent.

Additionally, above two solutions needs separate integration of finance functions.

We may leverage the potential of blockchain to provide trustworthy infrastructure in supply chain and integration of finances, we will discuss the same as applicable to purchase transactions.

However base functionality with minor change is also suitable for sales and reverse auctions.

Further this basic functionality could be extended by adding functions like onsite inspection done by third party, if applicable, clearing agent charges, inhouse inspection and integration of item with manufacturing domain to derive product cost.

1. **Who has the problem (the people who are facing the problem):**

The problem is faced by major industries where additional functionality needs to be mapped using API with ERP to push data from web-based system to ERP system. Such data being pushed inherently carries the threat of security issues. Also, additional finance integration is required to release the payments to business partners.

1. **What are they doing currently for the problem?**

Third party system developed pushing json/xml files to business integrated ERP environment for further completing cycle of logistics transactions and further finance integration to release the payments to Business partners is the way to address this issue with existing systems.

1. **What is the solution proposed and how it will solve the problem? What are the challenges with this solution or implementation?**

Leverage trustworthiness of Existing hyper ledger networks to build secure chain of transactions which on completion forms a block of blockchain which is immutable. Blockchain embedded finance system will be used for releasing business partner payments.

This could be achieved either by means of existing open blockchain systems or more secure cloud based blockchain architecture available with major cloud service providers like AWS and Azure.

In case more security is required company may build own blockchain system but that will have issues similar to existing ERP system integration as business partner payment portals cannot be seamlessly integrated with supply chain.

Since existing blockchain system has inbuilt functionality of creating json files as data files same, could be used for pushing data in ERP for reference purposes.

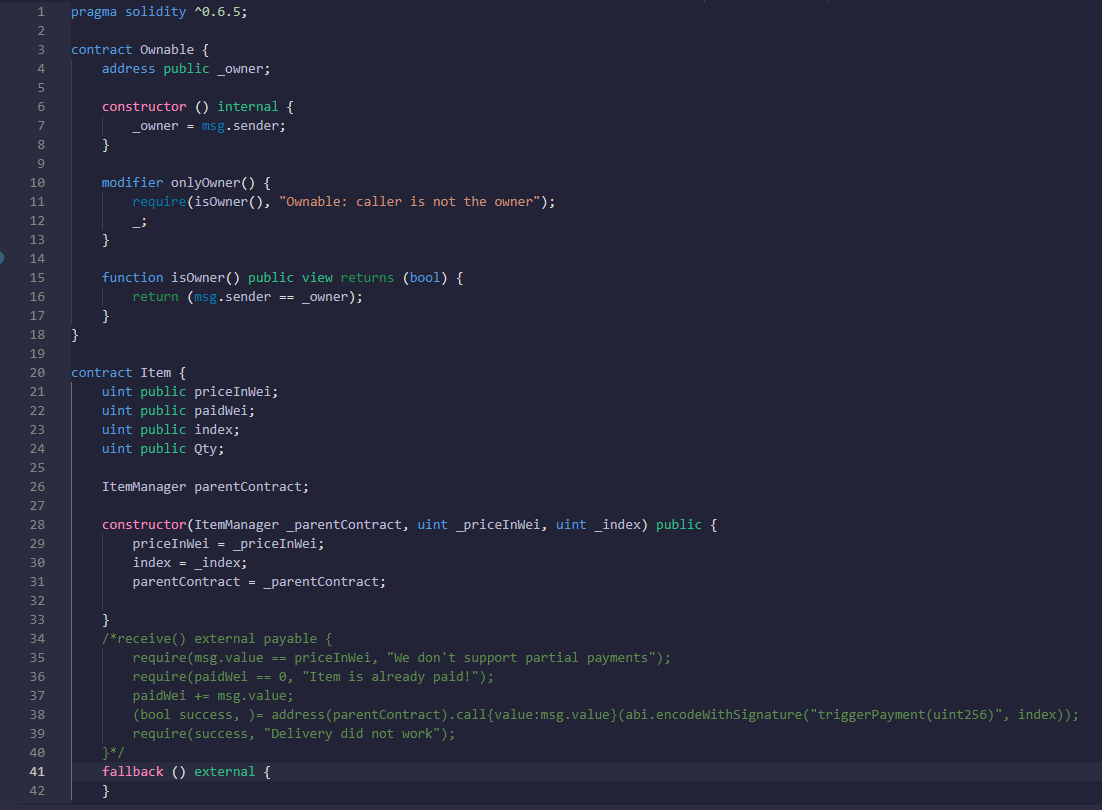
ERP can be provided with seamless integration over existing blockchain technologies on which trustworthy frame work can be built else can be used as data holders with minimal transactions.

Blockchain at SCM will require new rules to be implemented within algorithm and from government legal perspective.

Consensus mechanism which needs gas needs to be reevaluated since as per existing mechanism transaction offering higher gas are given first priority for settling transactions because of higher benefits to miners. However same might not be applicable in SCM.

1. **Solidity code for this with explanation.**

Solidity code is divided as below in two different screenshots.

Code part1:

This part discusses creating contract ownable with

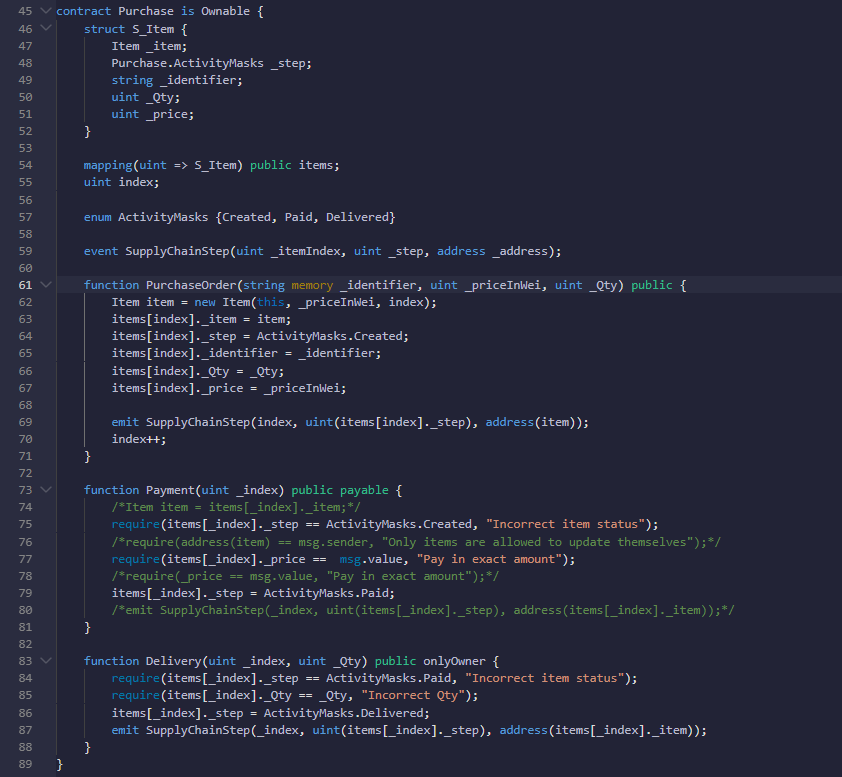
* Constructor internal to set owner as message sender.
* Further modifier “onyowner”

is used in declarative way to change behavious of function in a way that execution of function is owned by initator and no other account is eligble to execute that part of programme.

* Contract “item”

enables functionality of creation of Items and are included variable if required could be used to integrated with PO system built in later part of programme and will be used equivalent as item master data with functionaily of price integration validity extended from supply chain to Item master.

* Fallback function “external” uses inbulit functionality of solidity for fallbacking transactions. For enabling revert of transaction in case of failed
* Another constructor to define price in wei, wei is smallest denomination of Ethereum.

Code part 2: 

This part discusses creation of

* contract “Purchase orders” having

Item details fetched from contract “Item” and with.

Activity mask as enum variable as purchase activty steps “Purchase.ActivityMasks \_step” to identify status of purchase order

Identifier as Purchase order position as when first time PO-Item is created it will have identifier as 0, when second PO-Item is created it will have identifier as 1 and so on.

Qty as ordered qty.

Price as ordered price.

If successful then sets PO status to ActivityMask as delivered.

And emits ActivityMask and address to hyperledger.

* Function purchase order when called creates PO with details

activity mask as created which is status of Purchase order.

Identifier which is position of PO starting from 0 and so on.

Qty, agreed price and and Item description to be used at the creation of Purchase order.

* Function Delivery for enabling user to maintain delivery.

When called validates if ActivityMask status is at “created” else fallback function is initiated.

Checks if qty delivered is in full else fallback is initiated.

If successful then sets PO status to ActivityMask as delivered.

And emits ActivityMask and address to hyperledger.

Qty and price could also be written to hyperledger if required.

In this case it is assumed that creator of order will make delivery in their own system and hence function is made “ownable” to ensure correct qty received are entered in system.

However same could be enabled for business partner to make the entries by adding variable “Address” at the time of creation of PO and selecting ethereum account address of validator.

Same can be validated at the time of Delivery if delivery is being done from same address by using inbuilt function “address(this)”. And any qty received short physically could be sent back by creating return PO with amount equivalent to deficient qty. Additional function to verify qty received could be built.

* Function Payble for enabling payment.

When called validates if ActivityMask status is at “created” else fallback function is initiated.

Checks if paymnet done is in correct amount else fallback function initiated.

If successful then sets PO status to ActivityMask as Paid.

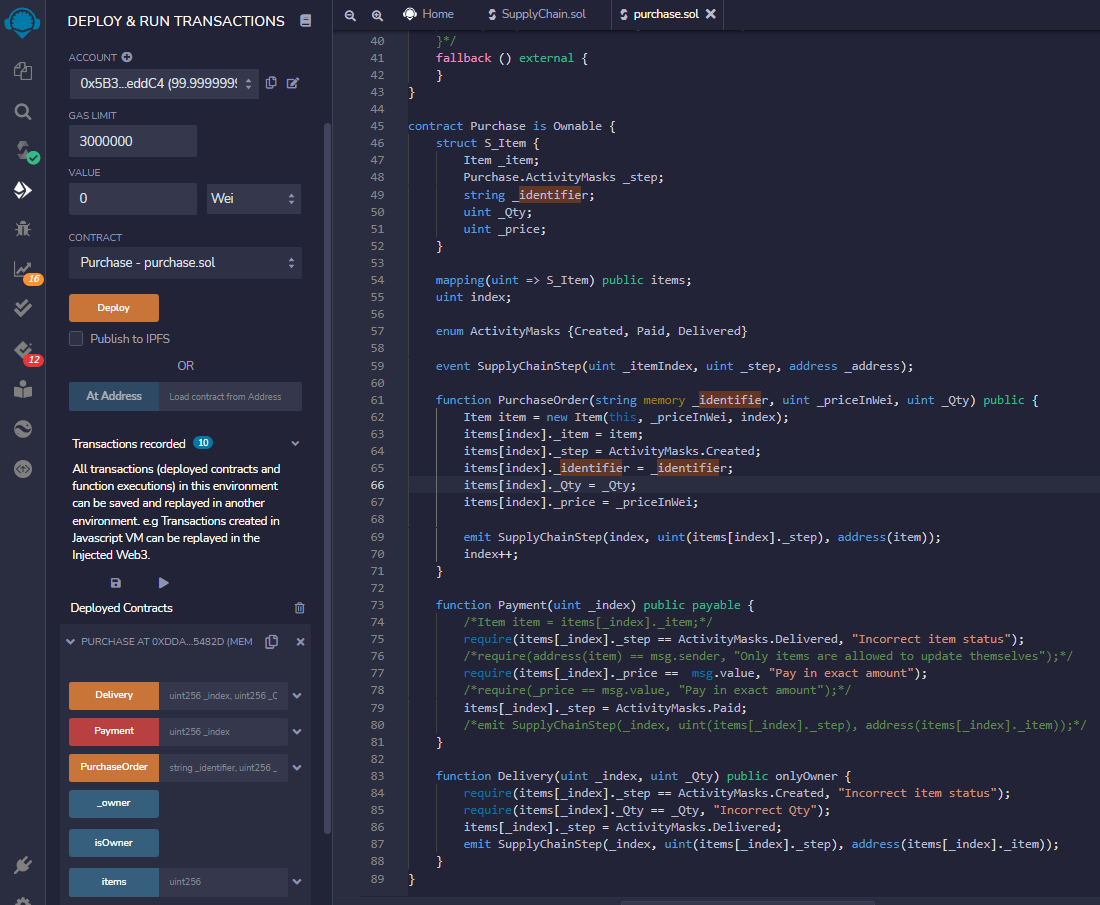
And if require will emit ActivityMask and address to hyperledger alongwith payment done value.

**Actual working of contract :**

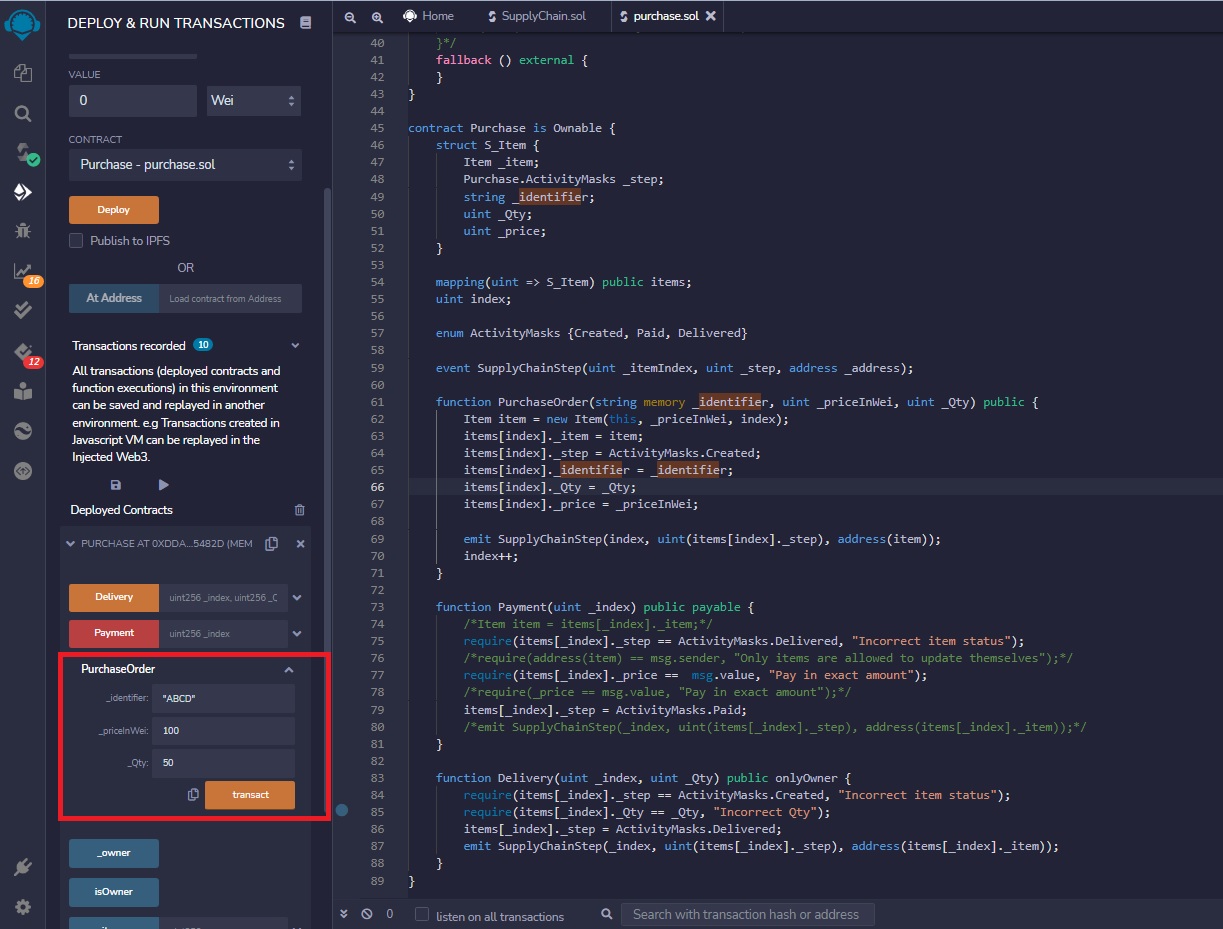
when “Purchase” is called will enable purchase function from.

A)When compiled and deployed will create activity tasks for calling functions.

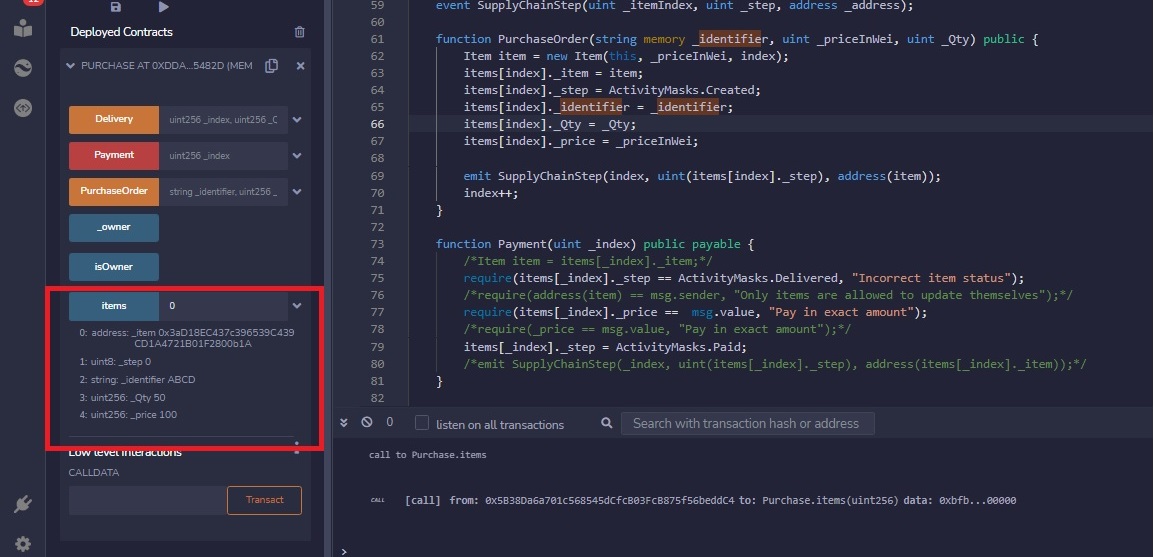
* + Purchase Order
  + Delivery
  + Payments
  + \_Owner
  + Isowner
  + Items



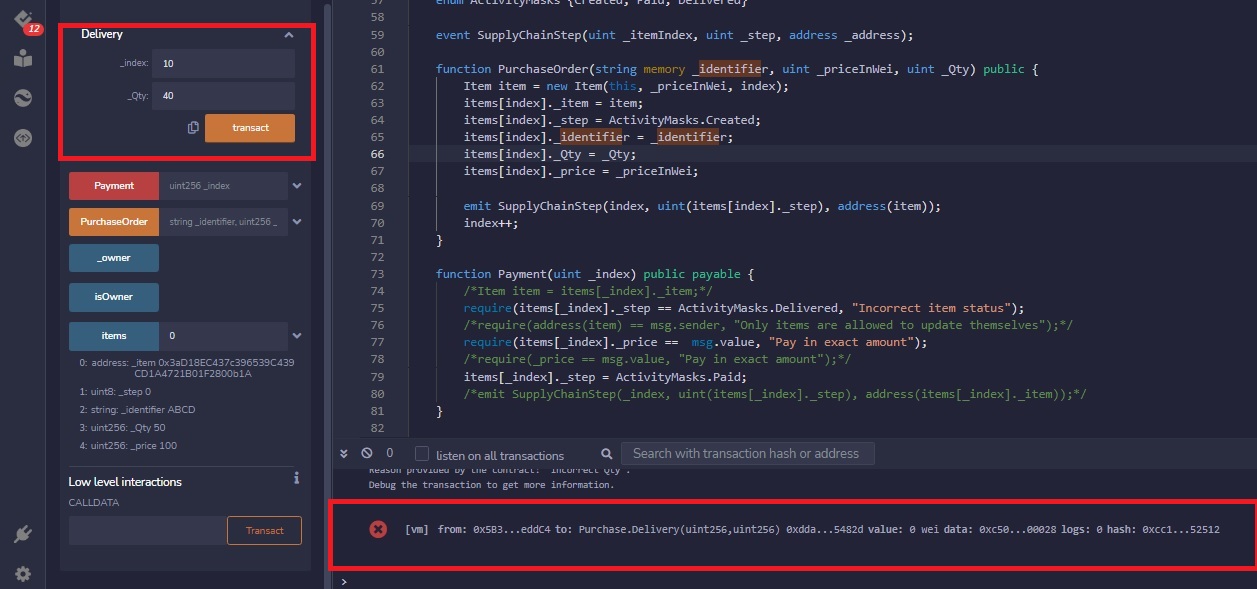
B)Purchase Order function when called enable us to make Item PO order line.



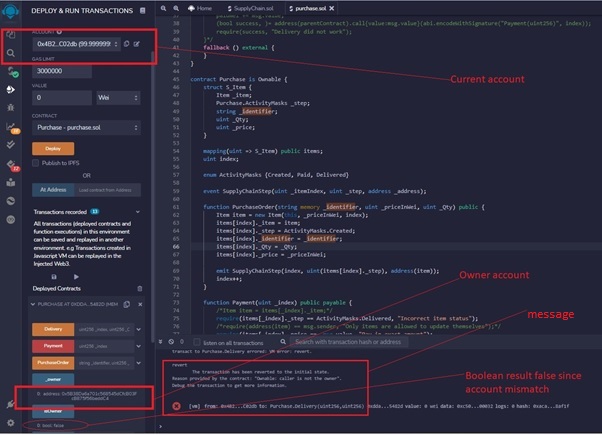
Zeroth Sequence created is can be called from items button for checking at the time of delivery it also shows hash address of item if it is to be ensured that same address is to be ued for delivery.



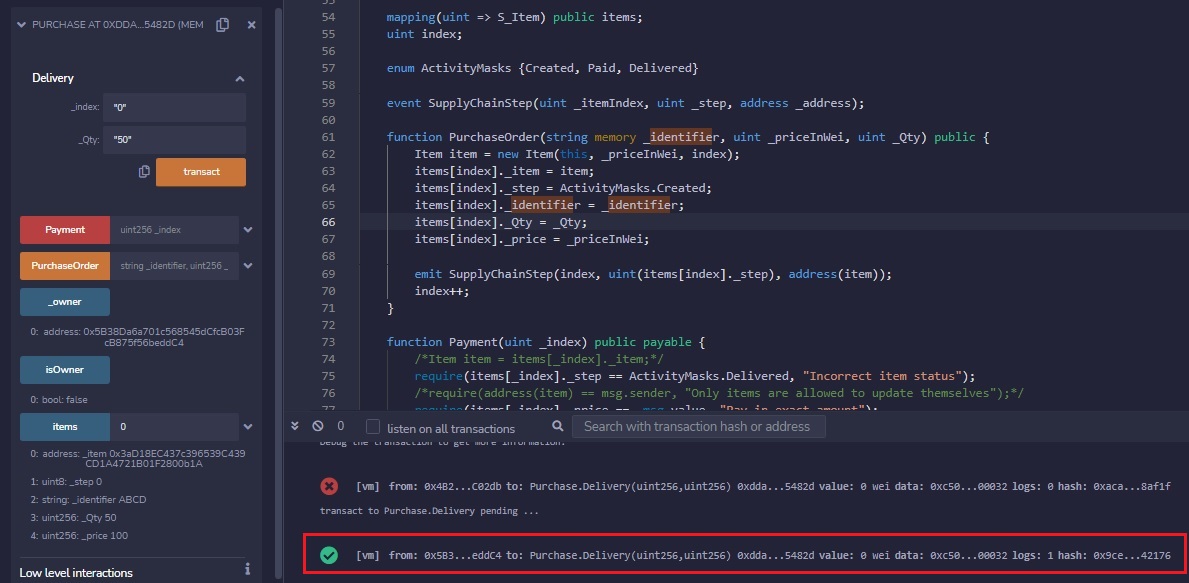
C)At the time of delivery if status of PO is wrong or qty is wrong or if wrong PO position is selected system will give us error message.



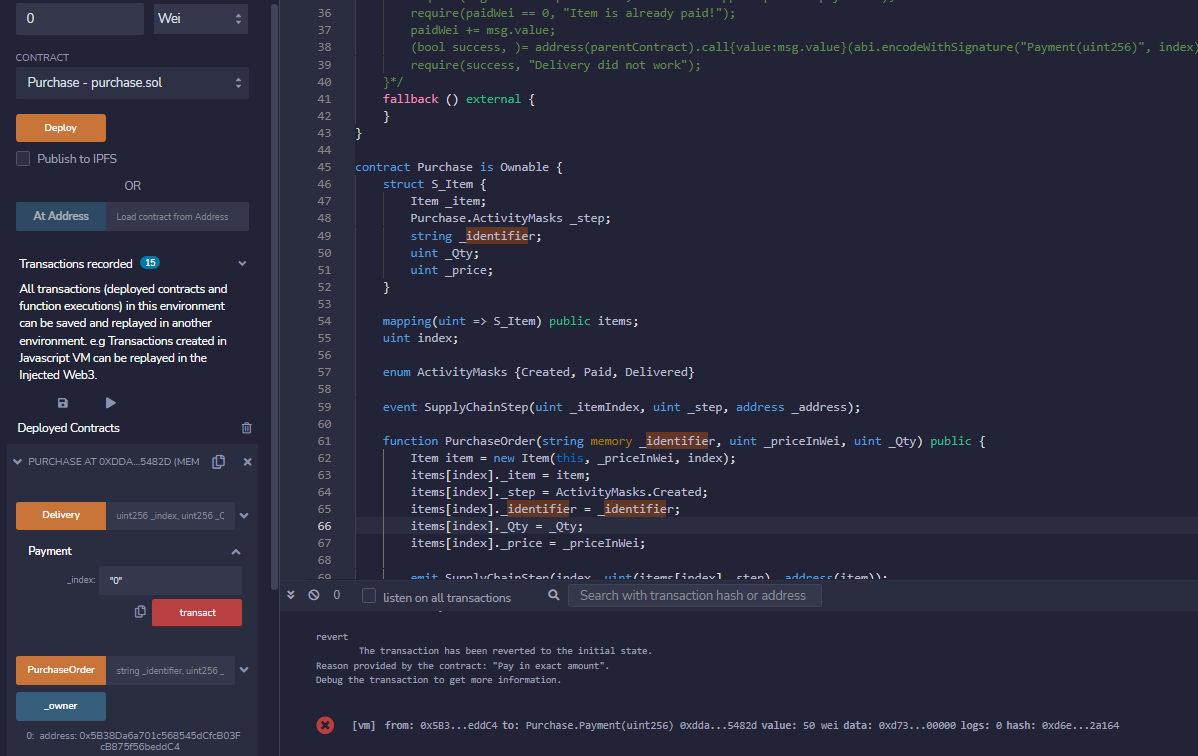
If owner is making delivery then he can check same account is used for delivery by checking boolean output is true or false.

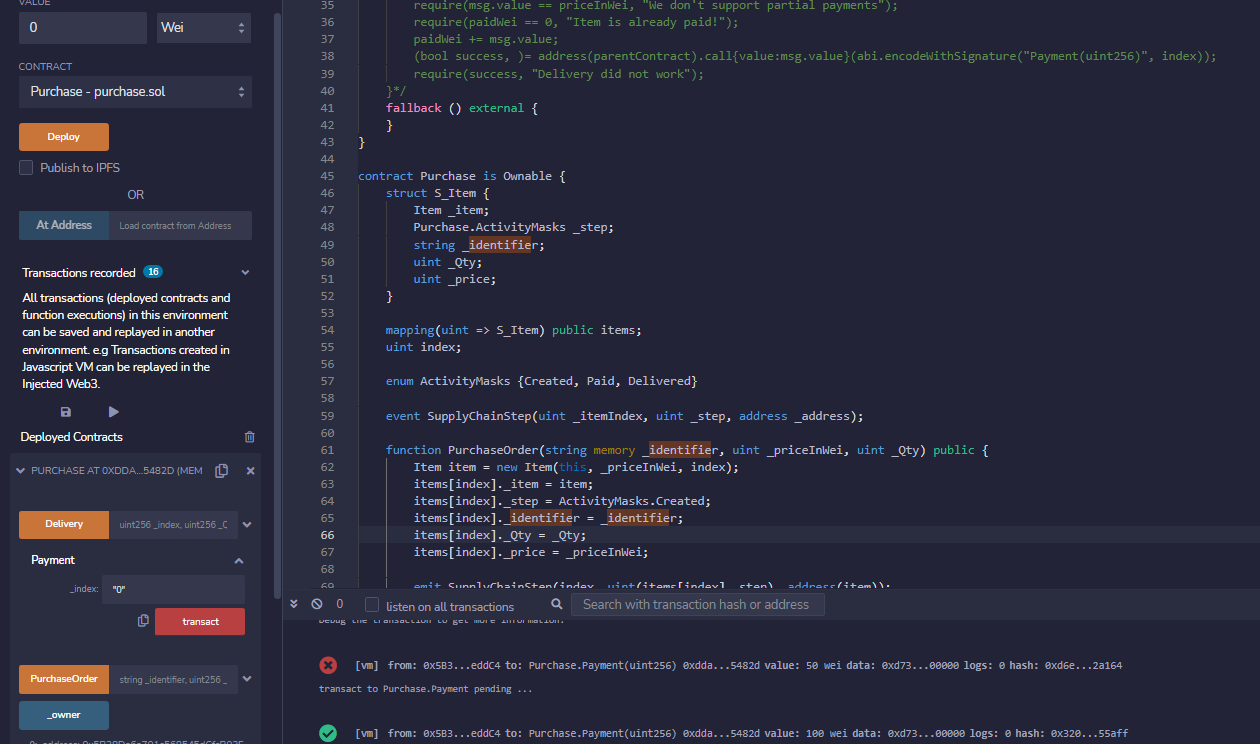


With correct address, position and qty delivery is successful.



D)If paymnet amount, status of PO or sequence selected is wrong error will be thrown.



If if is same paymnet will go throgh.

1. **Demo video.** Desktop recording with audio explaining the code and its working. Upload the video in cloud source [google drive, YouTube, Vimeo, etc.] Provide the url and allow me to access the link provided.

<https://drive.google.com/file/d/1AqPleogs6Q78OMF2YwxjCW0gbxe1Igt5/view?usp=sharing>