

Solidity code submitted by : Shashank Solanki

Date: 19-05-2019

Deployed on: Ropsten Network

USDT

Contract address: 0x39e1e363c2c64134816453588c28187b36fc24a7

Token name : USDT

Decimal point : 18

ABCT

Contract address: 0xd9e33ace876107efdd878b852a271f85857c9c87

Token name : ABCT

Decimal point : 18

ABCStore

Contract address: 0xea85d06d4c5e9982a8e7f4f365850b09740c9f5a

ABI (attached as separate file)

Admin wallet details:

Address: 0x6aC3522D3fe190002467D10D27865B6f3744A177

Private Key:

5AF701AB87718B82E1857A2646BFDED2B2037AB70A7731C23B952048F3B34874

Buyer 1 wallet details:

Address: 0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64

Private Key:

2EDC352E97E733C440777BE82C5DE8E58916A8A7289180AAC91CF2417245ABC5

Buyer 2 wallet details:

Address: 0xB5184a2786EeE8b4fe9d9325683981601e1823F4

Private Key:

57BA3C5C229B0AB038D36AFF03F9B697D5506EA6F14DFFB545FE97E4A59F8DD8

I have already deployed all the contracts on Ropsten as mentioned above and project is ready to use , you can use your own wallet to add more buyers.

STEPS:

1) Load USDT and ABCT in ABCStore


adminAddNewToken

__symbol:

USDT

__cotractAddr:

0x39e1e363c2c64134816453588c28187b36fc24a7



transact


adminAddNewToken

__symbol:

ABCT

__cotractAddr:

0xd9e33ace876107efdd878b852a271f85857c9c87



transact

2) Admin can load item on adhoc basis, already loaded Skirt, Apple , Beer ,Pen and Shirt on contract load time

adminLoadItem ^

name:

Watch

price_in_cents:

800

quantityLeft:


19

canReturn:

true

daysOfReturn:

15



transact

You can check the same using below view

showItem

"Watch"

 v

0: uint256: price_in_cents 800

1: uint256: qtyLeft 19

2: bool: canReturn true

3: uint256: daysOfReturn 15

3) User can register with their personal wallet using below function

registerNewUser ^

walletAddress:

0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64

age:

22

gender:

M



transact

I have already registered
Buyer 1: age 22 , Male (M)

Buye 2: age 19 , Female (F)

User and admin can see user detail using

getUserDetails

0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64

▼

0: address: UserWalletAddress 0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64

1: uint256: age 22

2: string: gender M

3: uint256: purchaseMadeThisMonthInCents 0

4: uint256: ABCT 0

5: uint256: USDT 0

getUserDetails

0xB5184a2786EeE8b4fe9d9325683981601e1823F4

▼

0: address: UserWalletAddress 0xB5184a2786EeE8b4fe9d9325683981601e1823F4

1: uint256: age 19

2: string: gender F

3: uint256: purchaseMadeThisMonthInCents 0

4: uint256: ABCT 0

5: uint256: USDT 0

4) Admin topup user wallet

adminTopupAccountUSDT

^

walletAddredd:

0xB5184a2786EeE8b4fe9d9325683981601e1823F4

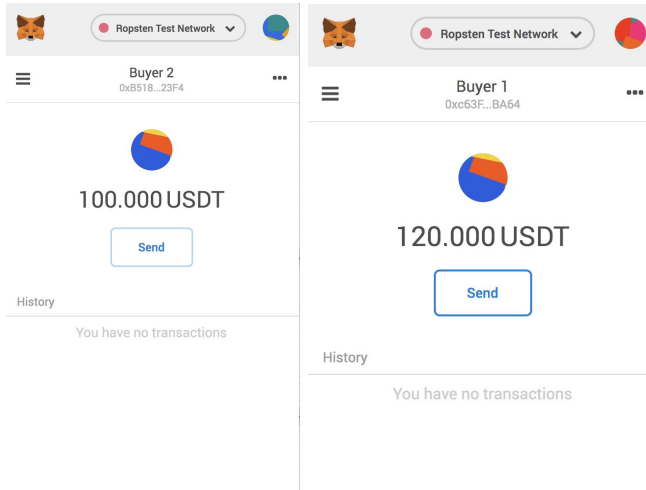
value:

100



transact

Load token contract in wallet to see balance (I used Metamask)



5) Lets buy something


Buyer 1 is a mail 22 year age , so he can buy beer but not Skirt (modifier is in solidity code)

Login with buyer 1 address in Metamask and place order as below

Before order place, Item details:

showItem

item:



0: uint256: price_in_cents 300

1: uint256: qtyLeft 100

2: bool: canReturn false

3: uint256: daysOfReturn 0

User details

getUserDetails

0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64

▼

0: address: UserWalletAddress 0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64
1: uint256: age 22
2: string: gender M
3: uint256: purchaseMadeThisMonthInCents 0
4: uint256: ABCT 0
5: uint256: USDT 12000000000000000000

BUY

buy

^

item:

Beer

qty:

5



transact

After successful purchase (5 less qty in inventory)

showItem

^

item:

Beer



call

0: uint256: price_in_cents 300
1: uint256: qtyLeft 95
2: bool: canReturn false
3: uint256: daysOfReturn 0

User information after purchase has been made , \$15 USDT deducted and in purchase made this month it is also showing \$15 (1500 cents)

getUserDetails

0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64



0: address: UserWalletAddress 0xc63F007E49cB947E6C4f7a9726C86Dd32773BA64
1: uint256: age 22
2: string: gender M
3: uint256: purchaseMadeThisMonthInCents 1500
4: uint256: ABCT 0
5: uint256: USDT 10500000000000000000

In Logs we can find transaction ID as below

```
{
  "from": "0xea85d06d4c5e9982a8e7f4f365850b09740c9f5a",
  "topic": "0xbf729ac16e44f3f27f9351d4a6f6f586e7a5925e25e29048f530474bc3b642b6",
  "event": "_loggerInt",
  "args": {
    "0": "1223133",
    "logger": "1223133",
    "length": 1
  }
}
```

If we check is this Item eligible for return we can use below function (Beer is not eligible)

checkReturnEligibility

1223133



0: bool: success false

Buyer 2 : Female 19 year cannot buy beer but can buy skirt

Transaction ID as below

```
{
  "from": "0xea85d06d4c5e9982a8e7f4f365850b09740c9f5a",
  "topic": "0xbf729ac16e44f3f27f9351d4a6f6f586e7a5925e25e29048f530474bc3b642b6",
  "event": "_loggerInt",
  "args": {
    "0": "1223134",
    "logger": "1223134",
    "length": 1
  }
}
```

If we check is this Item eligible for return we can use below function (Skirt is eligible for return within 15 days)

checkReturnEligibility 1223134



0: bool: success true

6) Month end scheduler can be written using Ethereum Alarm Clock or can be handled by external oracles of DAPPs

Functions to convert into elite member and to mint ABCT on the basis of purchase made are as below

```
/**
 * We can schedule this via Ethereum Alarm clock or external oracles, or from our DAPP,
 * To check at every month end if any shopper has made transaction of more than 500 USD ,he/she turns
to elite member
 *
 **/
```

```
function checkEliteMembership() internal ownerCheckAdmin
{
    for(uint i=0;users.length>i;i++){
        if(_userList[users[i]].purchaseMadeThisMonthInCents>=50000)//checking in cents
        {
            _userList[users[i]].isEliteShoper=true;
        }
    }
}
```

```
/**
 * We can schedule this via Ethereum Alarm Clock or external oracles, or from our DAPP (client side)
 * This is to mint and deposit ABC tokens at the month end as same amount of USDT spent
 *
 **/
```

```
function monthEndABCTMint() internal ownerCheckAdmin
{
    for(uint i=0;users.length>i;i++){

        require(_userList[users[i]].age!=0);
        bytes32 symbol_ = stringToBytes32( "ABCT");
        address usdtContractAddr = _tokenMap[symbol_];
        ERC20InterfaceUSDTether=ERC20(usdtContractAddr);
```



```
ERC20InterfaceUSDTether.topupAccount(users[i],getBackPriceUpto2dp(convertToWei(_userList[users[i]].purchaseMadeThisMonthInCents)),msg.sender);
```

```
    }  
}
```