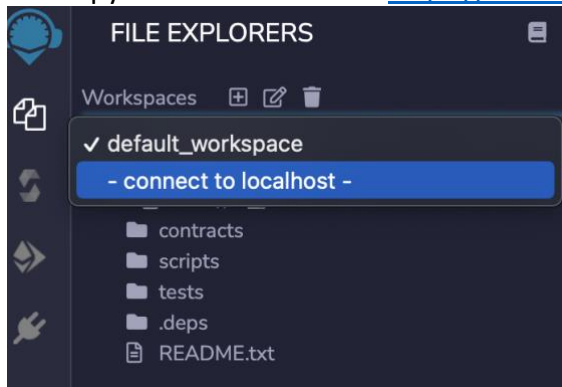


Steps to run the smart contract with Remix (<https://remix.ethereum.org/>)

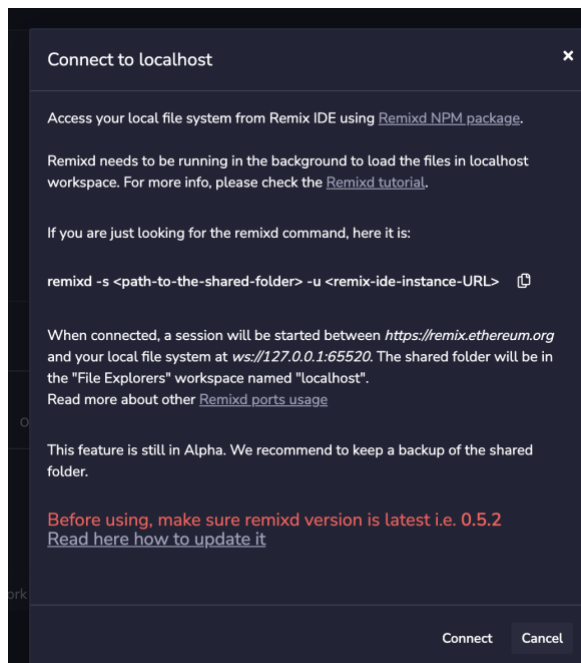
- Ref link (<https://remix-ide.readthedocs.io/en/latest/remixd.html>)
- `sudo npm install -g @remix-project/remixd`  
(if any error related to access denied while installing, use `chown` to give access)
- `remixd -s <your project path> --remix-ide https://remix.ethereum.org/#optimize=false`

```
RMacBookPro:NFTMarketPlace raviranjanchoudhary$ remixd -s /Users/raviranjanchoudhary/Documents/GitHub/NFTMarketPlace --remix-ide https://remix.ethereum.org/#optimize=false
[INFO] you are using the latest version 0.5.2
[WARN] You may now only use IDE at https://remix.ethereum.org/#optimize=false to connect to that instance
[WARN] Any application that runs on your computer can potentially read from and write to all files in the directory.
[WARN] Symbolic links are not forwarded to Remix IDE
[INFO] Mon Nov 01 2021 02:36:21 GMT+0800 (Hong Kong Standard Time) remixd is listening on 127.0.0.1:65520
[INFO] Mon Nov 01 2021 02:36:21 GMT+0800 (Hong Kong Standard Time) slither is listening on 127.0.0.1:65523
```

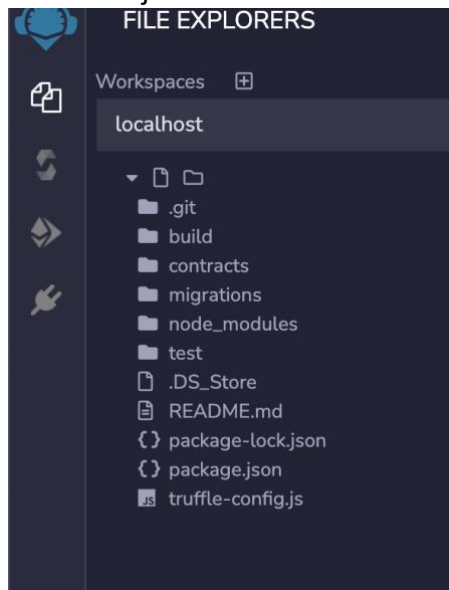
- Copy this url in browser : <https://remix.ethereum.org/#optimize=false>



- Click connect to localhost -> click connect



- Project loaded in remix



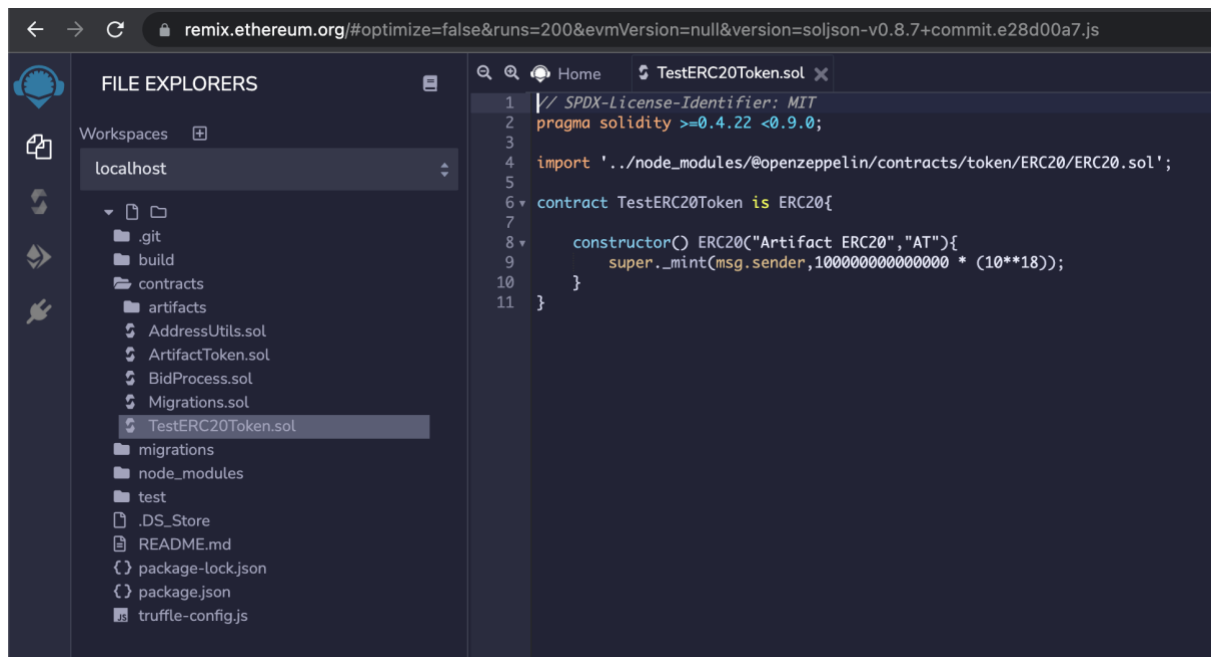
#### Contract Description:

- TestERC20Token: FT ERC20 token will be used to buy NFT from market place
- Artifact Token : NFT ERC721 token which will be sold on market place.
- BiddingProcess: Market place for bidding of NFT ERC721

All the addresses are kept for reference purpose for testing.

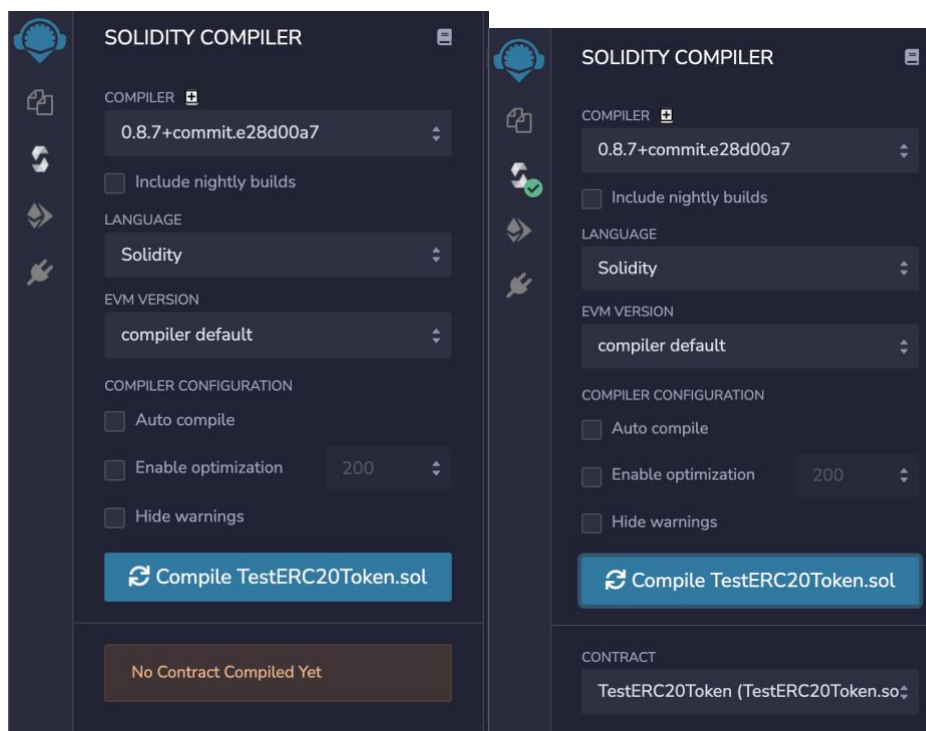
## Deployment of contracts:

- ERC20 token




The screenshot shows the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel displays the file structure of the project on 'localhost'. The 'contracts' folder is expanded, showing files like 'AddressUtils.sol', 'ArtifactToken.sol', 'BidProcess.sol', 'Migrations.sol', and 'TestERC20Token.sol'. The 'TestERC20Token.sol' file is selected. The main editor on the right shows the code for 'TestERC20Token.sol'.

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity >=0.4.22 <0.9.0;
3
4 import '../node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol';
5
6 contract TestERC20Token is ERC20{
7
8     constructor() ERC20("Artifact ERC20","AT"){
9         super._mint(msg.sender,1000000000000000 * (10**18));
10     }
11 }
```



The screenshot shows the 'SOLIDITY COMPILER' panel in the Remix IDE. The compiler is set to '0.8.7+commit.e28d00a7'. The language is 'Solidity' and the EVM version is 'compiler default'. The compiler configuration includes 'Auto compile' (unchecked), 'Enable optimization' (unchecked, set to 200), and 'Hide warnings' (unchecked). A blue button labeled 'Compile TestERC20Token.sol' is visible. Below the compiler settings, a message states 'No Contract Compiled Yet'.

COMPILER 

0.8.7+commit.e28d00a7

☐ Include nightly builds

LANGUAGE

Solidity

EVM VERSION

compiler default

COMPILER CONFIGURATION

☐ Auto compile

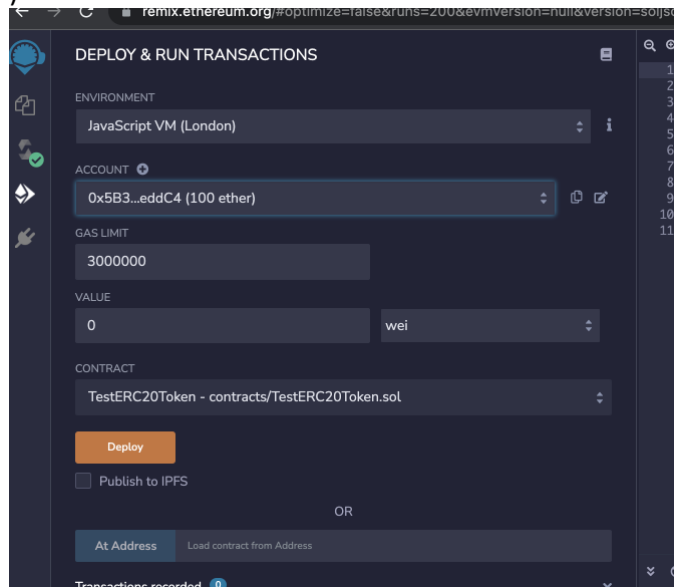
☐ Enable optimization 200

☐ Hide warnings

Compile TestERC20Token.sol

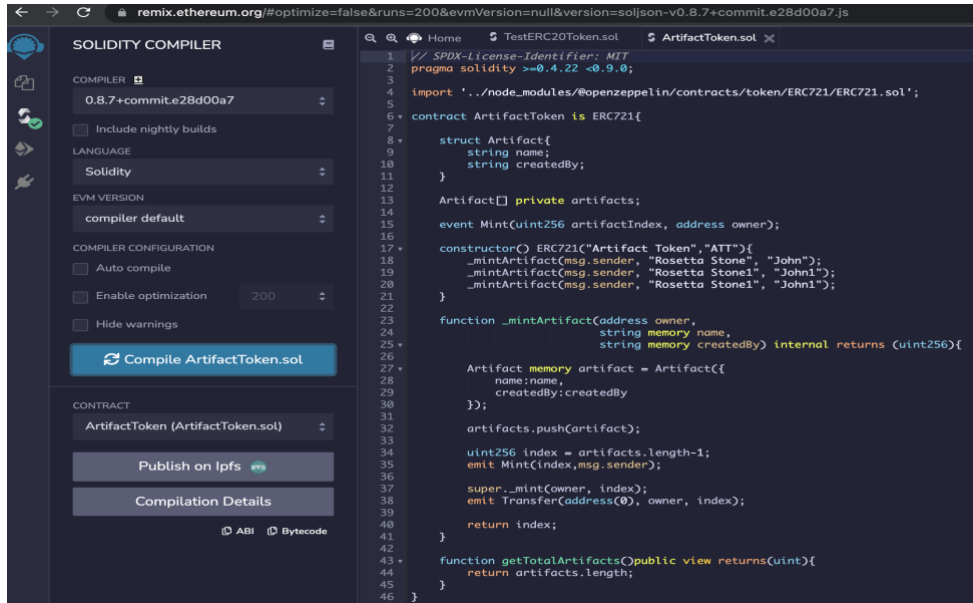
No Contract Compiled Yet

First account is selected for deployment(`0x5B38Da6a701c568545dCfcB03FcB875f56beddC4`)



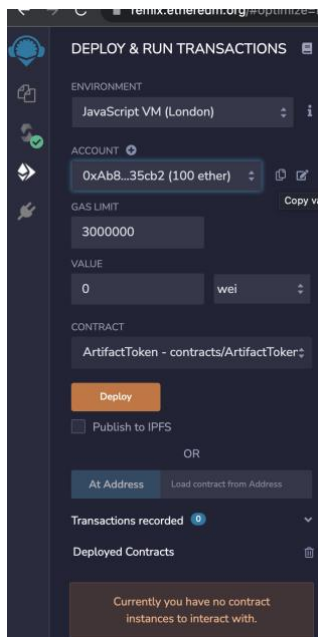
Contract Address : `0xaE036c65C649172b43ef7156b009c6221B596B8b`

- NFT Token



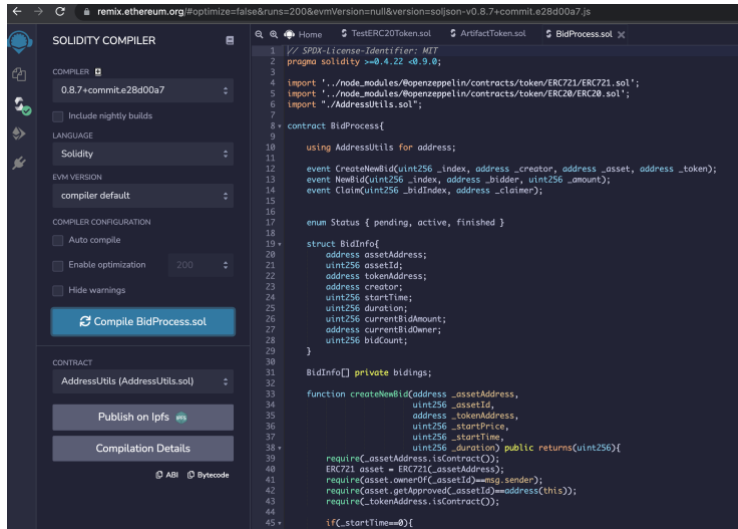
Deployed at second address from account

list(**0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2**)

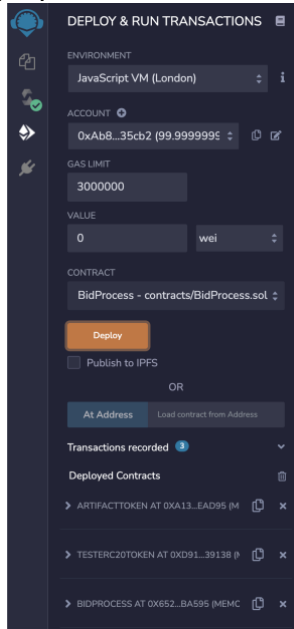


Contract Address : **0x9C9fF5DE0968dF850905E74bAA6a17FED1Ba042a**

- Market Place



Deployed at second address from list(0xA8b8483F64d9C6d1EcF9b849Ae677dD3315835cb2)



Contract Address: 0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8

Before starting creating any bidding, marketplace(contract address) access have to approved by NFT owner.

Input : Contract address of marketplace, id number of artifacts.

The screenshot shows a transaction interface titled "ARTIFACTTOKEN AT 0X9C9...A042A (MEMORY)". The transaction type is "approve". The "to:" field contains the address "0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8". The "tokenId:" field contains the value "0". A "transact" button is visible at the bottom right.

Start creating listing for new bid:

Input

assetAddress is NFT contract address

assetId is the artifacts id

tokenAddress is ERC20 contract address

Startprice is bidding start price

Start time is when bidding starts (0 for current time)

Duration for the bid to be active (in seconds)

The screenshot shows a transaction interface titled "BIDPROCESS AT 0X9DA...CBBB8 (MEMORY)". It lists three functions: "bid" (parameters: uint256 \_bidIndex, uint256 \_amount), "claimAsset" (parameter: uint256 \_bidIndex), and "claimTokens" (parameter: uint256 \_bidIndex). The "createNewBid" function is expanded, showing the following inputs: "\_assetAddress:" (0x9C9f5DE0968dF850905E74bAA6a17FED1Ba042a), "\_assetId:" (0), "\_tokenAddress:" (0xE036c65C649172b43ef7156b009c6221B596B8b), "\_startPrice:" (1000), "\_startTime:" (0), and "\_duration:" (800). A "transact" button is at the bottom right.

Few sample operations regarding the new listing done using the artifact id(assetid).

getStatus	0
0: uint8: 1	
getTotalBids	
0: uint256: 1	
getWinner	0
isActive	0
0: bool: true	
isFinished	0
0: bool: false	



Let's start bidding:

Change the address to the first in list (owner which deployed the ERC20 smart contract)

Inside ERC20 approve the token for bidding

Input spender is MarketPlace address and amount is bidding amount.

DEPLOY & RUN TRANSACTIONS

0x5B3...eddC4 (99.999999999999533736 ether)

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

BidProcess - contracts/BidProcess.sol

Deploy

☐ Publish to IPFS

OR

At Address Load contract from Address

Transactions recorded 9

Deployed Contracts

TESTERC20TOKEN AT 0XAE0...96B8B (MEMORY)

approve

spender: 0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8

amount: 1050

transact

Few getters:

getCurrentBidAmount 0

0: uint256: 1050

getCurrentBidOwner 0

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

getStatus 0

0: uint8: 1

getTotalBids

0: uint256: 1

getWinner uint256\_index

isActive 0

0: bool: true

isFinished 0

0: bool: false

Try to bid from 3<sup>rd</sup> account (0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c)  
Transfer 2000 ERC token to 3<sup>rd</sup> address from the first address.

TESTERC20TOKEN AT 0xAE0...96B8B (MEMORY)

approve "0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8", "1050"

decreaseAllowance address spender, uint256 subtractedValue

increaseAllowance address spender, uint256 addedValue

transfer

recipient: 0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c

amount: 2000

transact

And bid the 1900 token for the nft using 3<sup>rd</sup> account

TESTERC20TOKEN AT 0xAE0...96B8B (MEMORY)

approve

spender: "0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8"

amount: 1900

transact

Finish the bidding in marketplace

BIDPROCESS AT 0x9DA...CBBB8 (MEMORY)

bid

\_bidIndex: 0

\_amount: 1900

transact

Few getters to see the updated bid info

getCurrentBidAmount	0
0: uint256: 1900	
getCurrentBidOwner	0
0: address: 0xCA35b7d915458EF540	
getStatus	0
0: uint8: 1	
getTotalBids	
0: uint256: 1	
getWinner	0
isActive	0
0: bool: true	
isFinished	0
0: bool: false	

Bidding is finished

getCurrentBidAmount	0
0: uint256: 2100	
getCurrentBidOwner	0
0: address: 0x5B38Da6a701c568545c	
getStatus	0
0: uint8: 2	
getTotalBids	
0: uint256: 1	
getWinner	0
0: uint256: 0	
isActive	0
0: bool: false	
isFinished	0
0: bool: true	

Claim asset with the winner address and claim token with market place address  
Input is the asset id(artificat id).

claimAsset	0
claimTokens	0

Token balance and ownership can be verified using the getter methods.

balanceOf

account: 0x9da9df2Fe440fA9E05B620a05990d7c644aCBBB8

call

0: uint256: 1500

ownerOf

0

0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4