CrowdSale PupperCoin

Crowdsale is required in blockchain space to have faith while raising funds, unlike traditional revenue-raising methods. An initial coin offering (ICO), or digital token crowdsale, is a method of blockchain-based crowdfunding based on the exchange of a project's new and unique cryptocurrency tokens for established cryptocurrencies like ETH, EOS, etc. Now, Crowdsale smart contract defines the rules and makes sure that there is transparency while raising these funds.

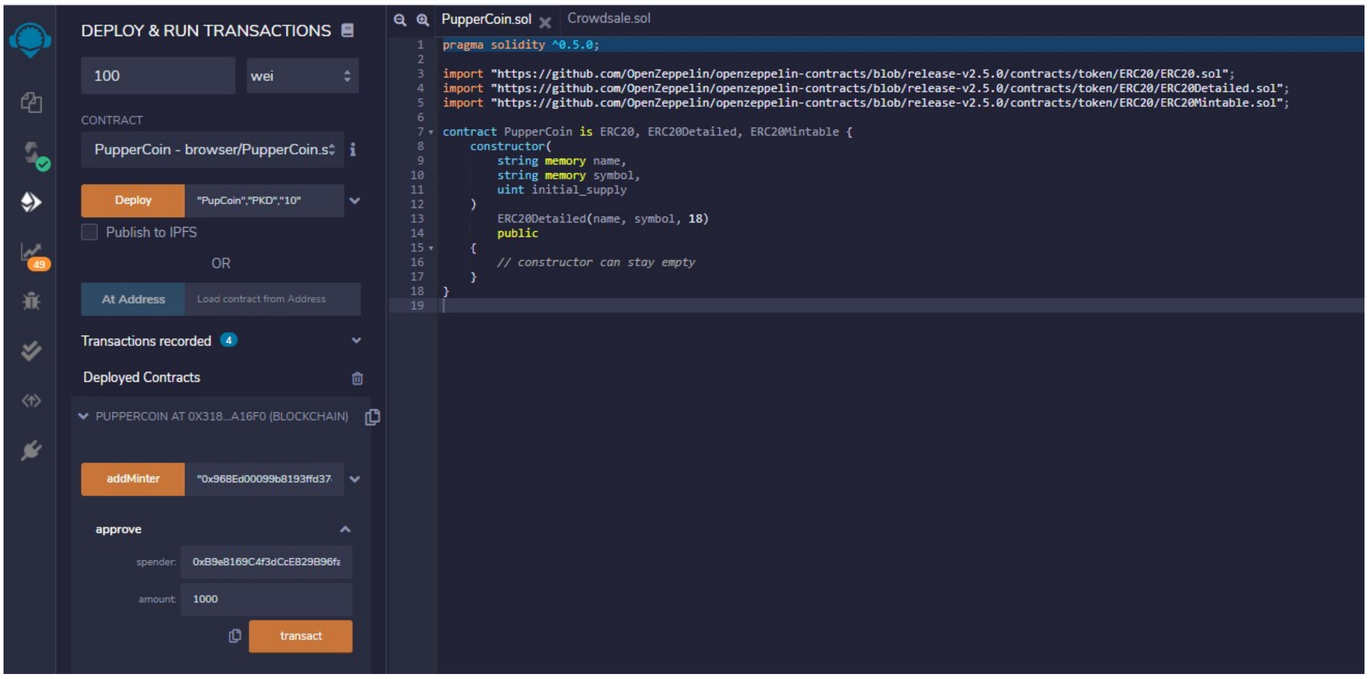
In this assignemnt, we are going to develop a crowdsale contract for PupeprCoin token to fund the network development.This network will be used to track the dog breeding activity across the globe in a decentralized way, and allow humans to track the genetic trail of their pets

Dependencies

1. In Remix open files PupeprCoin.sol and Crowdsale.sol
2. Open Ganache
3. Link metamask with Remix

Steps to deploy the contract

1. Compile PupperCoin.sol file



100 wei

PupperCoin - browser/PupperCoin.s

Deploy "PupCoin","PKD","10"

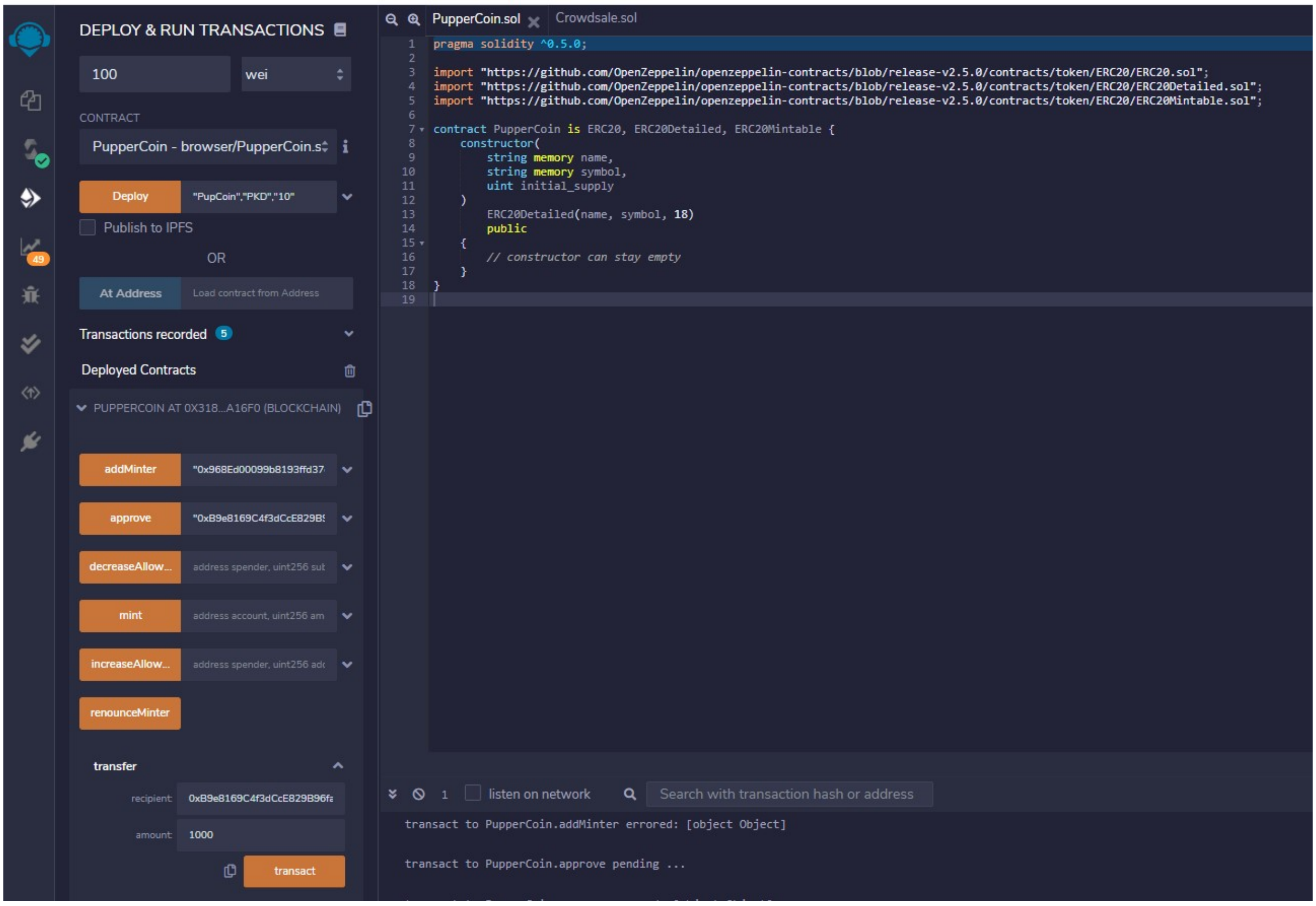
PUPPERCOIN AT 0X318...A16F0 (BLOCKCHAIN)

addMinter "0x968Ed00099b8193ffd37

approve
spender: 0xB9e8169C4f3dCcE829B96fa
amount: 1000

```
pragma solidity ^0.5.0;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Mintable.sol";

contract PupperCoin is ERC20, ERC20Detailed, ERC20Mintable {
    constructor(
        string memory name,
        string memory symbol,
        uint initial_supply
    )
        ERC20Detailed(name, symbol, 18)
        public
    {
        // constructor can stay empty
    }
}
```

2. Deploy the PupperCoin.sol file



VALUE: 100 wei

CONTRACT: PupperCoin - browser/PupperCoin.sol

Deploy: "PupCoin","PKD","10"

PUPPERCOIN AT 0X318...A16F0 (BLOCKCHAIN)

addMinter: "0x968Ed00099b8193ffd37
approve: "0xB9e8169C4f3dCcE829B5
decreaseAllow...: address spender, uint256 sub
mint: address account, uint256 am
increaseAllow...: address spender, uint256 adc

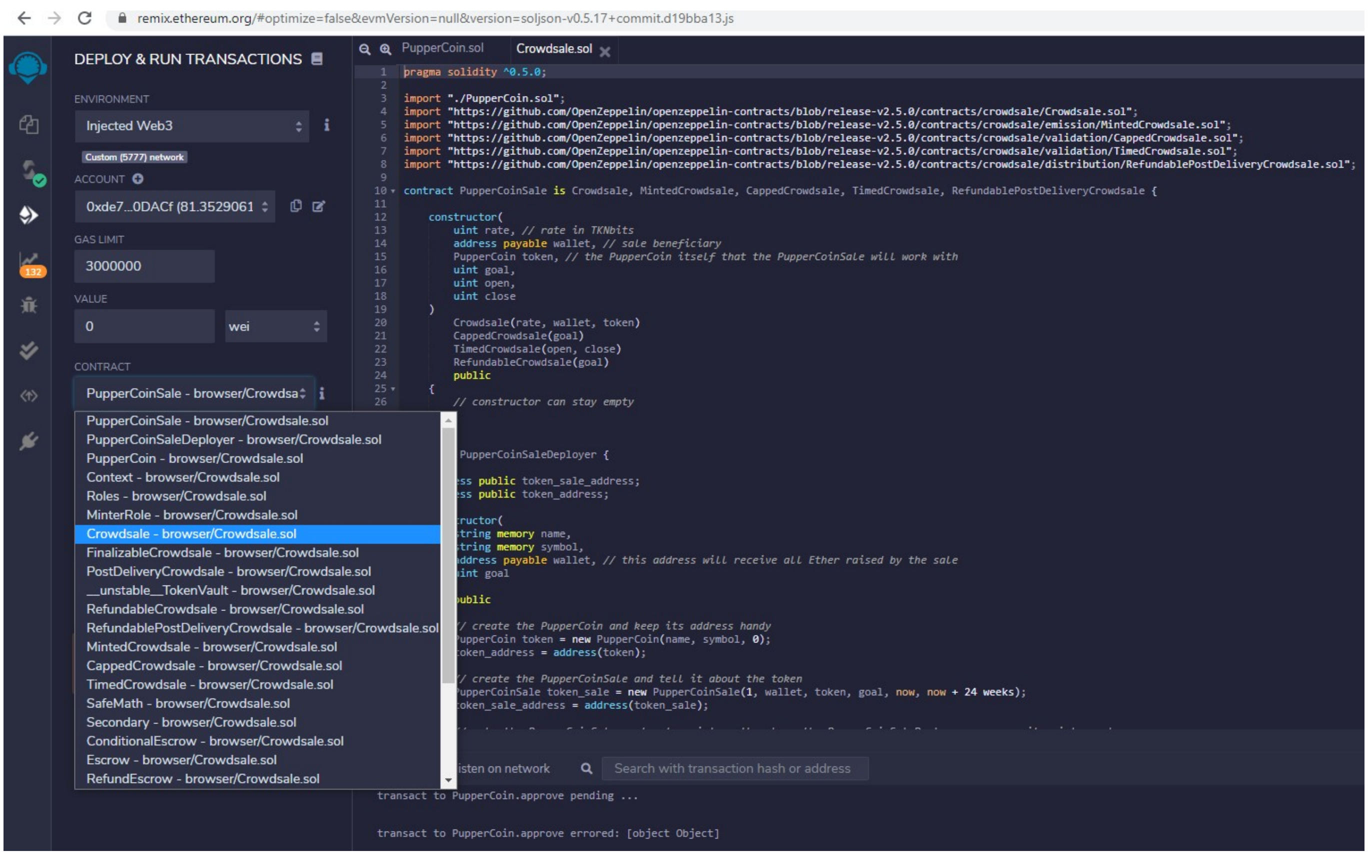
transfer — recipient: 0xB9e8169C4f3dCcE829B96fe; amount: 1000

PupperCoin.sol

```
pragma solidity ^0.5.0;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Mintable.sol";

contract PupperCoin is ERC20, ERC20Detailed, ERC20Mintable {
    constructor(
        string memory name,
        string memory symbol,
        uint initial_supply
    )
        ERC20Detailed(name, symbol, 18)
        public
    {
        // constructor can stay empty
    }
}
```

```
transact to PupperCoin.addMinter errored: [object Object]

transact to PupperCoin.approve pending ...
```

3. Compile Crowdsale.sol file


DEPLOY & RUN TRANSACTIONS
listen on network
Search with transaction hash or address

remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.17+commit.d19bba13.js

ENVIRONMENT: Injected Web3 (Custom (5777) network)

ACCOUNT: 0xde7...0DACf (81.3529061

GAS LIMIT: 3000000

VALUE: 0 wei

CONTRACT: PupperCoinSale - browser/Crowdsale.sol

PupperCoinSale - browser/Crowdsale.sol
PupperCoinSaleDeployer - browser/Crowdsale.sol
PupperCoin - browser/Crowdsale.sol
Context - browser/Crowdsale.sol
Roles - browser/Crowdsale.sol
MinterRole - browser/Crowdsale.sol
Crowdsale - browser/Crowdsale.sol
FinalizableCrowdsale - browser/Crowdsale.sol
PostDeliveryCrowdsale - browser/Crowdsale.sol
__unstable__TokenVault - browser/Crowdsale.sol
RefundableCrowdsale - browser/Crowdsale.sol
RefundablePostDeliveryCrowdsale - browser/Crowdsale.sol
MintedCrowdsale - browser/Crowdsale.sol
CappedCrowdsale - browser/Crowdsale.sol
TimedCrowdsale - browser/Crowdsale.sol
SafeMath - browser/Crowdsale.sol
Secondary - browser/Crowdsale.sol
ConditionalEscrow - browser/Crowdsale.sol
Escrow - browser/Crowdsale.sol
RefundEscrow - browser/Crowdsale.sol

Crowdsale.sol

```
pragma solidity ^0.5.0;

import "./PupperCoin.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/validation/CappedCrowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/validation/TimedCrowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/distribution/RefundablePostDeliveryCrowdsale.sol";

contract PupperCoinSale is Crowdsale, MintedCrowdsale, CappedCrowdsale, TimedCrowdsale, RefundablePostDeliveryCrowdsale {

    constructor(
        uint rate, // rate in TKNbits
        address payable wallet, // sale beneficiary
        PupperCoin token, // the PupperCoin itself that the PupperCoinSale will work with
        uint goal,
        uint open,
        uint close
    )
        Crowdsale(rate, wallet, token)
        CappedCrowdsale(goal)
        TimedCrowdsale(open, close)
        RefundableCrowdsale(goal)
        public
    {
        // constructor can stay empty



 PupperCoinSaleDeployer {

ss public token_sale_address;
ss public token_address;

ructor(
tring memory name,
tring memory symbol,
ddress payable wallet, // this address will receive all Ether raised by the sale
int goal

ublic

/ create the PupperCoin and keep its address handy
upperCoin token = new PupperCoin(name, symbol, 0);
oken_address = address(token);

/ create the PupperCoinSale and tell it about the token
upperCoinSale token_sale = new PupperCoinSale(1, wallet, token, goal, now, now + 24 weeks);
oken_sale_address = address(token_sale);
```

```
transact to PupperCoin.approve pending ...

transact to PupperCoin.approve errored: [object Object]
```

4. Deploy the Crowdsale.sol file