Implementation of Ethereum Request for Comment (ERC20) Token

Khalid Husain Ansari[1], Umesh Kulkarni[2]

Department of Computer Engineering

Vidyalankar Institute of Technology

Mumbai, India

khalid.ansari@vit.edu.in[1], umesh.kulkarni@vit.edu.in[2]

***Abstract*—Blockchain technology has been an essential aspect for research due to its, peer to peer nature, distributed nature, and decentralization, some of its applications are Supply chain, Healthcare System, Real state, etc. Ethereum is a distributed public blockchain network that focuses on running the code of any decentralized application. A smart contract is a computer program having self-verifying, self-executing, tamper-resistant properties. Initial Coin Offering (ICO) is a process similar to crowdfunding, in which companies raise funds from investors, who, in exchange receive tokens or digital assets. This paper deals by explaining the proposed model, different functions of ERC20 standard, the different tasks performed by each function and implementing various functionalities of ERC20 Token in Solidity.**

***Index Terms*— Blockchain, Ethereum, Smart Contract, ICO, Crowdsale, ERC20.**

I. INTRODUCTION

Different digital currencies have developed over the past few years, where these currencies are a replica of money that can be easily transferred over the Internet via banks and regulating agencies acting as intermediaries. Due to lack of transparency and centralization traditional currencies have issues, to solve these problems Digital currencies, which are referred as cryptocurrencies, were created. These cryptocurrencies can be built by organizations as well as by various start-ups. It is challenging for new organizations or start-ups to raise funds at the beginning across the entire globe. People and other organizations lack confidence and have an assumption of failure of the start-up; based on this assumption, they don't invest or share funds to those start-ups. Other organizations believes working in collaborations with well known, accomplished, settled, multi-national organizations with the fact or assumption that their investment, time, business, friendships will be for longer term, and investments at safer side (wont bear huge losses). Many people don't make investments in an online portfolio with assumptions of it being a fraud. Due to these reasons, many organizations and start-ups end or shuts-down even before starting.

Ethereum is a platform that has allowed people to create their cryptocurrencies, tokens, and new forms of digital assets. Such programmable blockchains have resulted in the emergence of an Initial Coin Offering (ICO). ICO is a fund-raising mechanism in which new organizations sell their underlying crypto-assets, typically ERC20 tokens in Ethereum, in exchange for either Bitcoin or Ether.

In this work, we aim is to design a system that will help users, organizations and start-ups. Organizations and start-ups will be able to raise funds regardless of being famous or not. User's security will be assured, with no frauds. We will write our ERC20 token smart contract for building our tokens, write a test against the smart contract.

Specific objective:

1. Creating our token smart contract.
2. Implementing ERC20 functions.
3. Writing a test for our smart contract.
4. Performing test in command prompt.
5. It stores token information on personal Blockchain.

This paper is formulated in the following way. Section II describes our proposed system, while Section III provides details about different software used and the installation procedure. Section IV describes the details about the token functions and shows the implementation part of the Token, and Section V concludes the paper.

II. PROPOSED SYSTEM

We aim to design a system that is used to raise funds for an organizations or start-ups. The fund-raising will be based on several tokens being sold; tokens will have a fixed price. In this project, funds will be in terms of Ether(Eth).

In this paper,

1. We are creating our cryptocurrency and also an ICO.
2. We are creating our cryptocurrency using Ethereum and will sell it.
3. We are generating our ERC20 Token with Smart Contract.
4. We will create a token sale(Crowdsale) Smart Contract.

5. We are writing test against both the Smart Contract and lastly
6. We are creating a Token sell Website where we can hold an ICO and allow people to purchase the Token.
7. A form that will allow us to purchase Token, You might be able to select number of Tokens that you want/need. You can buy them.

Metamask confirmation would pop-up whenever you buy some Tokens through the website. We can also see how many Tokens exist and number of Tokens sold.
We can also see the account to which we are connected to the test network. We can see the price of the Token and the number of tokens we have purchased with our account.

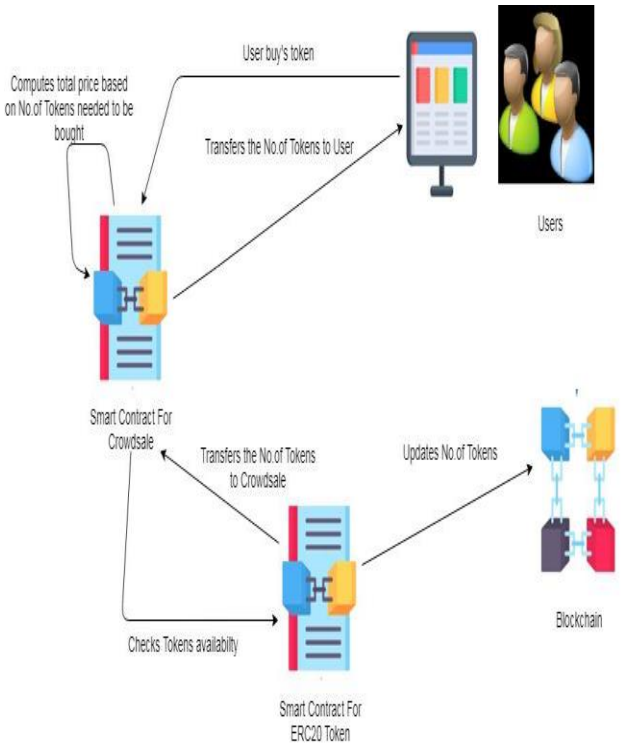


Figure 1. System Architecture

ERC20 Smart Contract has the following responsibility:

1. Describing Name, Symbol and Version of tokens.
2. Initialing total supply(number of tokens).
3. It Keeps track of balance of accounts that have some token with them by using blanaceOf.
4. Transferring tokens to addresses specified with value.
5. Triggers transfer event.
6. Approve function will allow someone to approve another account to spend tokens on their behalf.
7. Allowance function will store the allotted amount that we have approved to transferred in the approve function.

Crowdsale Smart Contract has the following responsibilities:

1. Provision tokens to token sale(Crowdsale) Contract.
2. Set token price in WEI.
3. Assigns an admin.
4. Buy token will allow users to buy tokens those tokens that we have built in ERC20 Smart Contract.
5. End sale.

User Interface(UI) has:

1. A form that will allow users to select the number of tokens that they want to buy, they have to purchase a minimum one token.
2. Buy tokens button in the form, will trigger the token purchase.
3. UI displays the account address of the users.
4. UI has a token price, also shows the number of tokens the logged user has with him.
5. UI will have a progress bar, which will show the number of tokens system has and the number of token sold.

DFD

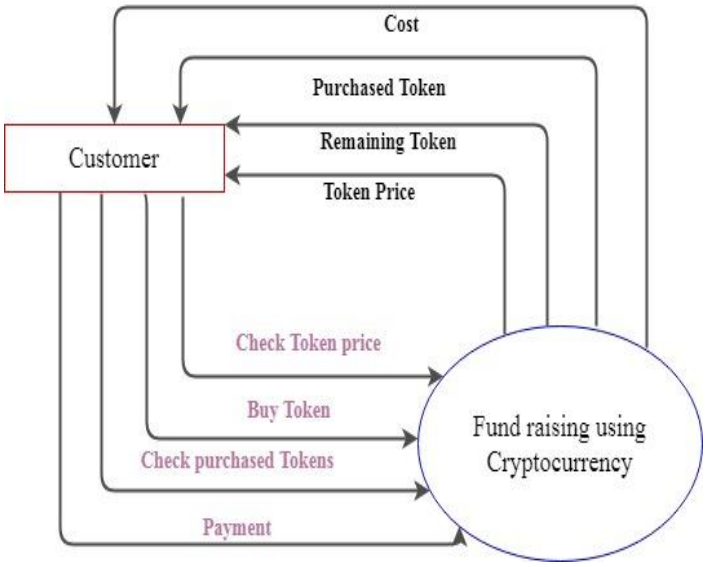


Figure 2. Level-0 DFD

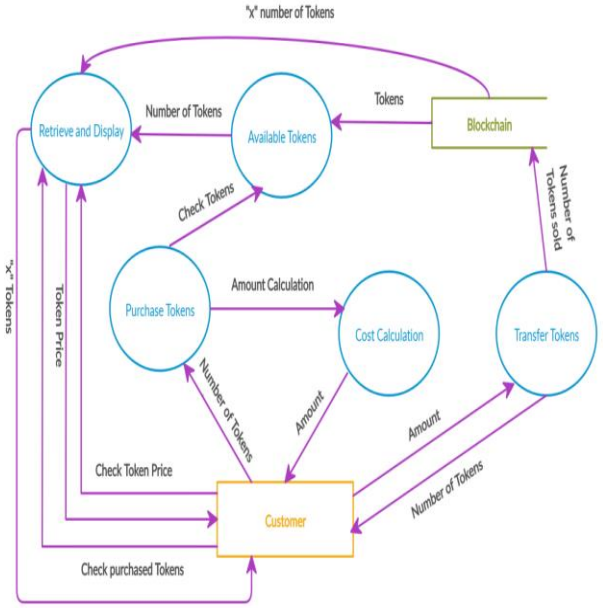


Figure 3. Level-1 DFD

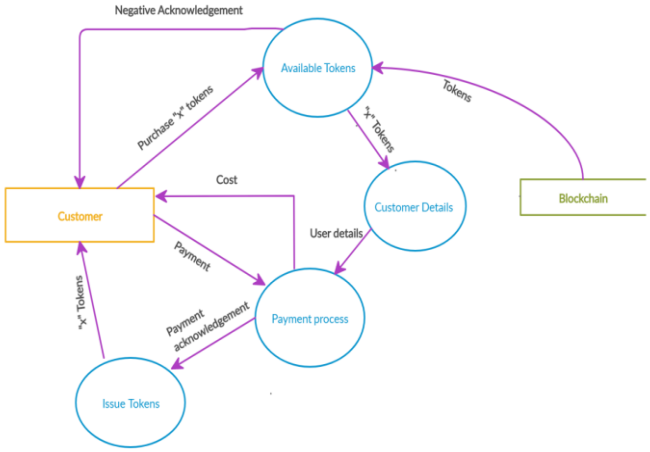


Figure 4. Level-2 DFD for purchase module

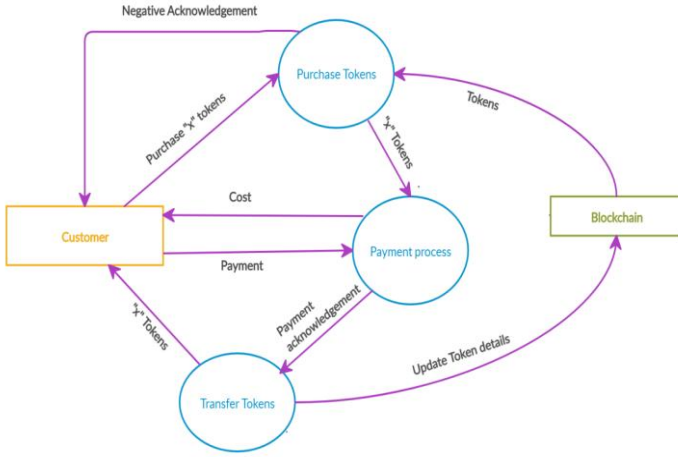


Figure 5. Level-2 DFD for transferring module

FLOWCHART

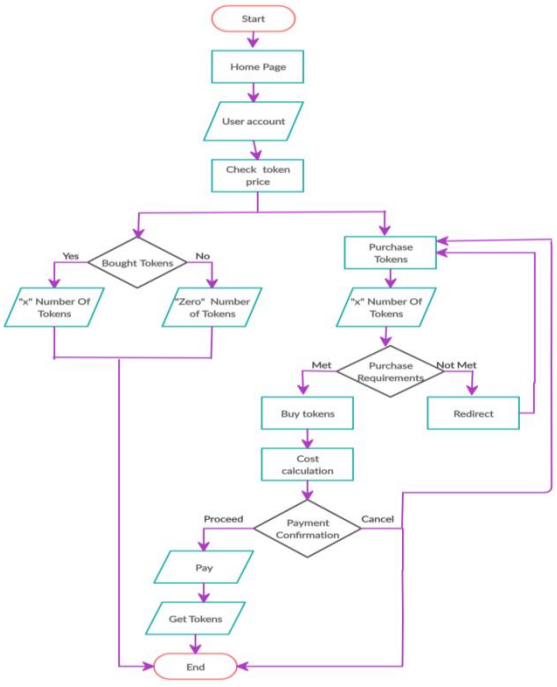


Figure 6. Flowchart

III. SOFTWARE USED

1. **Chocolatey:** Chocolatey is used for installing all of the system dependencies and it's a package manager for windows. The reason for using Chocolatey is because it's a seamless way to install new packages that are going to manage our environment.
2. **Nodejs:** Node.js is an open-source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules, which simplifies the development of web applications using Node.js to a great extent. Further, it can be used to install other packages by using Node Package Manager(NPM).
   Installation: You can directly download it from nodejs website or you can install it using powershell.
3. **Truffle:** It allows us to create decentralized application(DAPP) on the Ethereum network, also provides tools which help us in writing our Smart Contract with the Solidity programming language. It also provides a framework for testing our Smart Contract and tools to deploy our Smart Contract to Blockchain; we can also develop client-side application inside Truffle.
4. **Ganache:** Ganache is a personal Blockchain that allows us to install a Blockchain on our computer and use it for the development purposes. It gives us some accounts so that we can use them.
   Installation: You can install Ganache from truffle website.

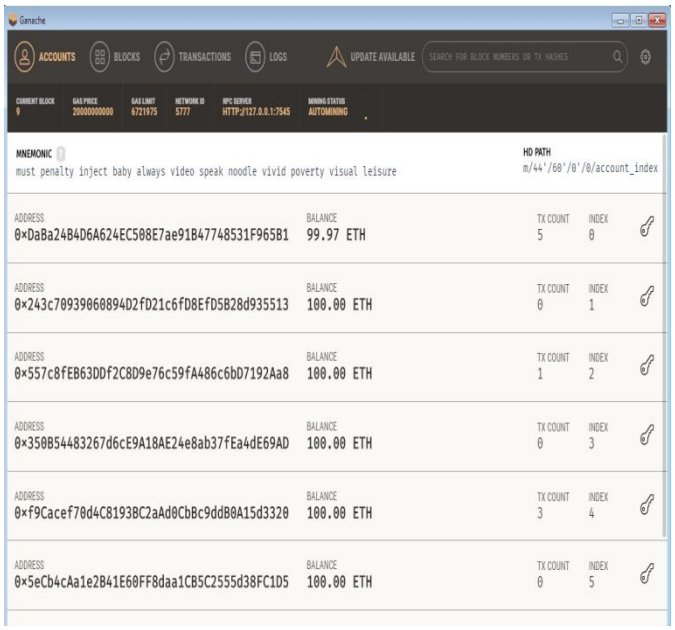


Figure 7. Ganache

5. **Metamask Extension:** To use to Blockchain, we must connect to the Blockchain and it can be done by using Metamask Google Chrome Extension. This extension allows us to connect to our local Ethereum network with our account and interact with a smart contract by using it.

Installation: To install metamask you need to go to the chrome web-store and search for metamask plug-in and then add it to chrome, once you have added metamask go to your chrome extension and ensure that metamask is enabled.

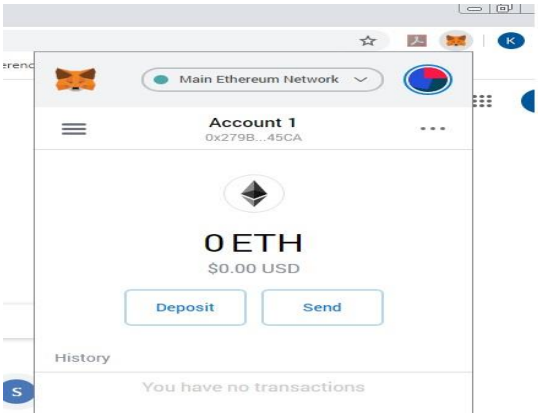


Figure 8. Metamask account

6. **Text Editor:** It is required for writing our source code.

IV. IMPLEMENTATION

STEP 1: Use the truffle init command in our implementation. Truffle init command will provide us with:

1. Contract folder with a migration.sol contract.
2. Migration folder with initial_migration details regarding the migration.sol file.
3. It provides an empty test folder where you can write your own test for the contract.
4. Truffle configuration file which is responsible for specifying details about network.

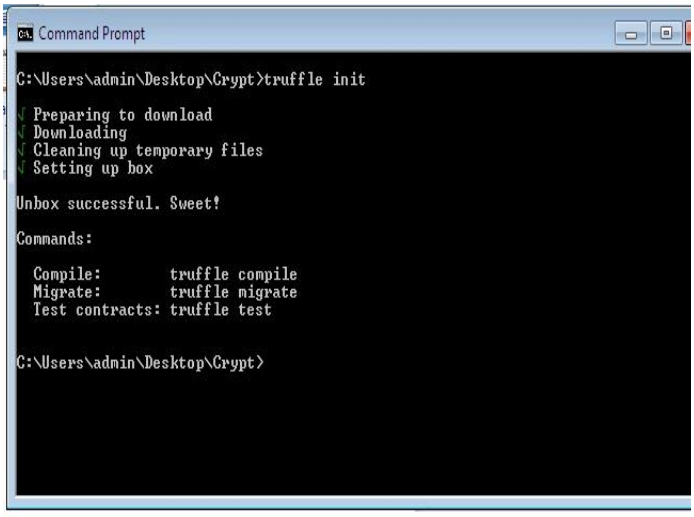


Figure 9. Truffle init command

STEP 2: Edit the truffle_config.js file, by specifying following details:

- Host
- Port
- Network_id

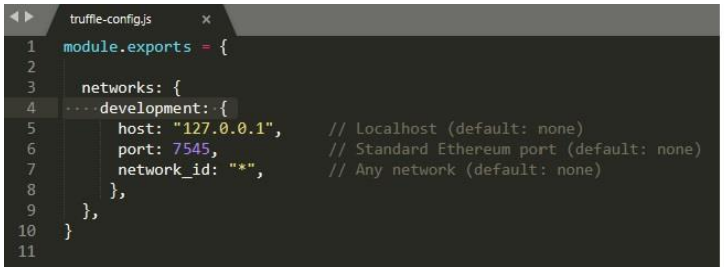


Figure 10. Truffle-config

STEP 3: Create our contract deployment file, which is used for interaction with truffle contract in truffle console or truffle test.

Figure 11. 2_deploy_contract

Artifacts allow us to create contract abstraction that Truffle can use to run in a javascript runtime environment. It has several different applications

- Allow us to interact with Smart Contract in any javascript runtime environment like our truffle console, or when we are writing test or when we are trying to interact with Smart Contract with our client-side application.

ERC20 Smart Contract

The different functionalities that will be used by us in developing our tokens in the smart contract are listed one after the other as follows:

1. **name():-** It returns the name of the token. E.g. “MyToken”.
   OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present [5].
   function name() public view returns (string) [5].

2. **symbol():-** It returns the symbol of the token. E.g. “EX”.
   OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present [5].
   function symbol() public view returns (string) [5].

3. **standard():-** Returns the standard of the token. E.g. “MyToken v.1.0”.
   OPTIONAL - This method can be used to improve usability, but interfaces and other contracts MUST NOT expect these values to be present [5].
   function standard() public view returns (string) [5].

4. **totalSupply():-** It returns the total token supply of the system. E.g. “100000”.

function totalSupply() public view returns (uint256) [5].

5. **balanceOf():-** Returns the account balance of another account with address _owner [5].
   It will perform following actions:
   - ✓ It will be responsible for knowing each token belongs.
   - ✓ Whenever a token is bought or sold or transferred, this mapping is going to be responsible for knowing each token belongs.

   function balanceOf(address _owner) public view returns (uint256 balance) [5].

6. **transfer():-** We want to add the ability to pay with tokens or send tokens or move them from one place to another. It [5] transfers `_value` amount of tokens to address `_to`, and MUST fire the `Transfer` event. The function SHOULD `throw` if the message caller's account balance does not have enough tokens to spend.
   Note [5] Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.
   Transfer function will perform following actions:
   - ✓ It will throw exceptions if the account doesn't have enough tokens with them.
   - ✓ It checks the balanceOf of the account which is transferring the tokens.
   - ✓ If the account transferring token has enough tokens then the transfer function will transfer/send the balance.
   - ✓ Transfer function will be responsible for triggering transfer event.
   - ✓ It returns Boolean.

   function transfer(address _to, uint256 _value) public returns (bool success) [5].

   Transfer Event:
   MUST trigger when tokens are transferred, including zero value transfers. A token contract which creates new tokens SHOULD trigger a Transfer event with the _from address set to 0x0 when tokens are created.
   event Transfer(address indexed _from, address indexed _to, uint256 _value) [5]

7. **approve():-** Implementing functions that handles transfers where the account didn't initially initiate a transfer and this can be referred to as delegated transfer, it will be a two step process one function is going to allow our account to approve a transfer and another function is going to handle the delegated transfer. It will allow someone to approve another account to spend tokens on their behalf.
   It will perform following actions:
   - ✓ Accepts an address of Spender amount to that we want to approve to send on our behalf .
   - ✓ Handle the allowance.
   - ✓ Handle approval event.
   - ✓ It returns Boolean.

   function approve(address _spender, uint256 _value) public returns (bool success) [5].

   Approval Event:
   MUST [5] trigger on any successful call to approve function.
   event Approval(address indexed _owner, address indexed _spender, uint256 _value) [5].

8. **allowance():-** Allowance is basically the allotted amount that we have approved to transfer.
   Returns [5] the amount which _spender is still allowed to withdraw from _owner.
   function allowance(address _owner, address _spender) public view returns (uint256 remaining) [5].

9. **transferFrom():-**The transferFrom function will handle the transfer once you have approved certain amount the third party address can actual execute that transfer. It [5] transfers `_value` amount of tokens from address _from to address `_to`, and MUST fire the `Transfer` event. This can be used for example to allow a contract to transfer tokens on our behalf and/or to change fees in sub-currencies. The function SHOULD `throw` unless the _from account has deliberately authorized the sender of the message via some mechanism.
   Note [5] Transfers of 0 values MUST be treated as normal transfers and fire the `Transfer` event.
   TransferFrom function will perform following actions:
   - ✓ It requires that _from has enough tokens.
   - ✓ Requires allowance to be big enough.
   - ✓ Whenever a transfer happens it will always trigger a transfer event.
   - ✓ Changes the balance.
   - ✓ Updates the allowance.
   - ✓ Returns Boolean.

   function transferFrom(address _from, address _to, uint256 _value) public returns (bool success) [5].

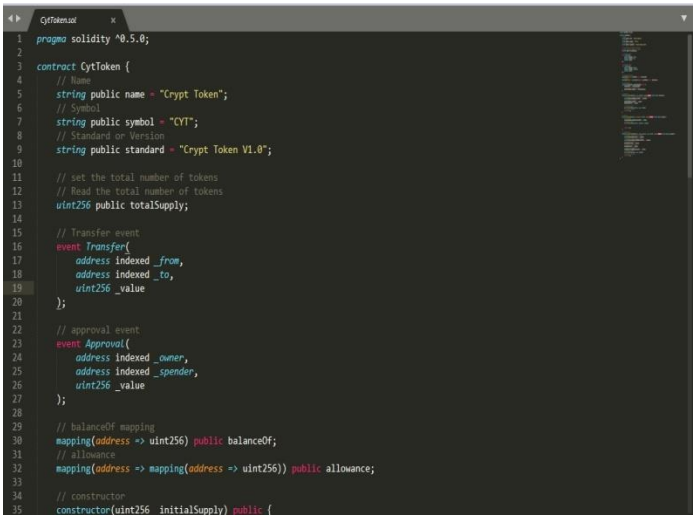
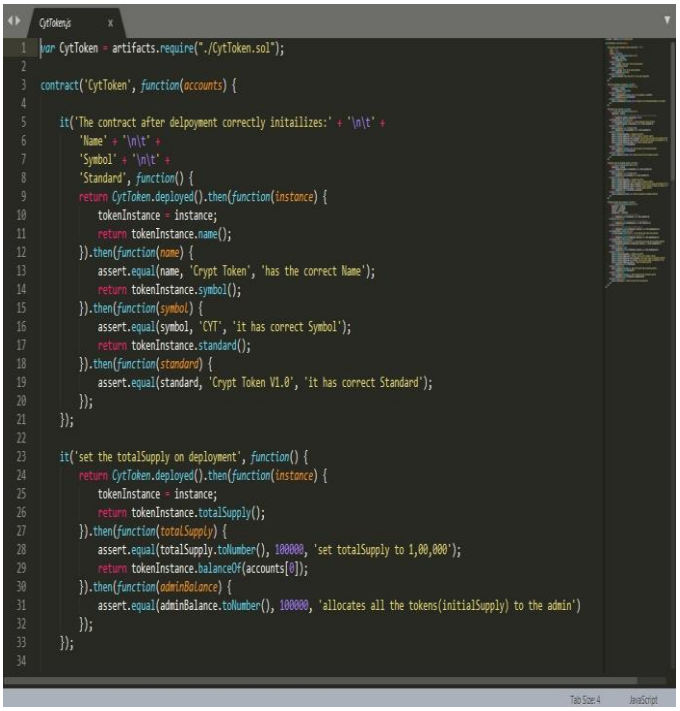


Figure 12. ERC20 token Smart Contract



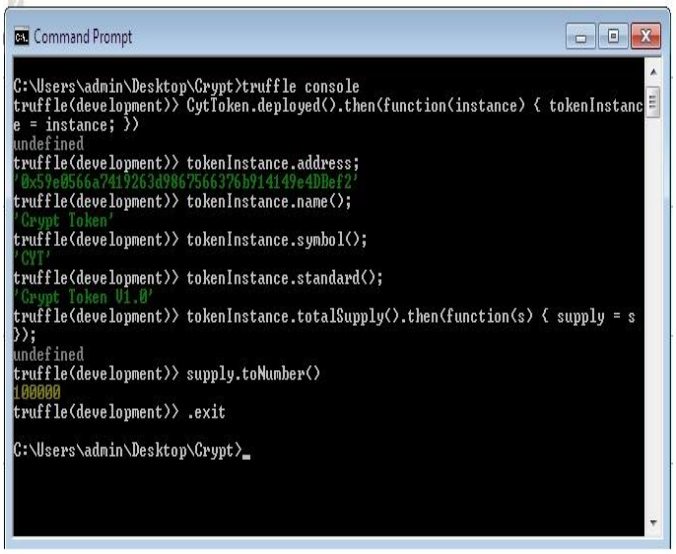
```
var CytToken = artifacts.require("./CytToken.sol");

contract('CytToken', function(accounts) {

    it('The contract after delpoyment correctly initailizes:' + '\n\t' +
        'Name' + '\n\t' +
        'Symbol' + '\n\t' +
        'Standard', function() {
        return CytToken.deployed().then(function(instance) {
            tokenInstance = instance;
            return tokenInstance.name();
        }).then(function(name) {
            assert.equal(name, 'Crypt Token', 'has the correct Name');
            return tokenInstance.symbol();
        }).then(function(symbol) {
            assert.equal(symbol, 'CYT', 'it has correct Symbol');
            return tokenInstance.standard();
        }).then(function(standard) {
            assert.equal(standard, 'Crypt Token V1.0', 'it has correct Standard');
        });
    });

    it('set the totalSupply on deployment', function() {
        return CytToken.deployed().then(function(instance) {
            tokenInstance = instance;
            return tokenInstance.totalSupply();
        }).then(function(totalSupply) {
            assert.equal(totalSupply.toNumber(), 100000, 'set totalSupply to 1,00,000');
            return tokenInstance.balanceOf(accounts[0]);
        }).then(function(adminBalance) {
            assert.equal(adminBalance.toNumber(), 100000, 'allocates all the tokens(initialSupply) to the admin')
        });
    });
```

Figure 13. Test file for ERC20 Smart Contract

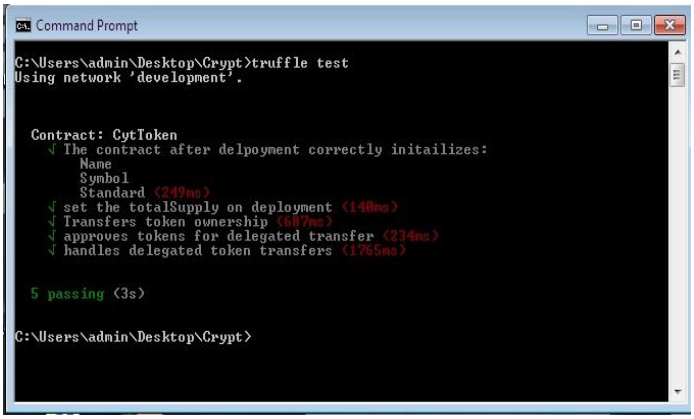


```
Command Prompt

C:\Users\admin\Desktop\Crypt>truffle console
truffle(development)> CytToken.deployed().then(function(instance) { tokenInstanc
e = instance; })
undefined
truffle(development)> tokenInstance.address;
'0x59e0566a7419263d9867566376b914149e4DBef2'
truffle(development)> tokenInstance.name();
'Crypt Token'
truffle(development)> tokenInstance.symbol();
'CYT'
truffle(development)> tokenInstance.standard();
'Crypt Token V1.0'
truffle(development)> tokenInstance.totalSupply().then(function(s) { supply = s
});
undefined
truffle(development)> supply.toNumber()
100000
truffle(development)> .exit

C:\Users\admin\Desktop\Crypt>_
```

Figure 14.Truffle console



```
Command Prompt

C:\Users\admin\Desktop\Crypt>truffle test
Using network 'development'.


  Contract: CytToken
    √ The contract after delpoyment correctly initailizes:
        Name
        Symbol
        Standard (249ms)
    √ set the totalSupply on deployment (140ms)
    √ Transfers token ownership (687ms)
    √ approves tokens for delegated transfer (234ms)
    √ handles delegated token transfers (1765ms)


  5 passing (3s)

C:\Users\admin\Desktop\Crypt>
```

Figure 15. Test

V. CONCLUSION

The proposed system will create its cryptocurrency named Crypt**(CYT)**. The system developed has two smart contracts one each for ERC20 Token and Crowdsale. We have developed the smart contract for token portion of our system. We have described various function used in creating our tokens and different event and task performed by each of them. Successfully conducted test against our token smart contract which checks various aspects of tokens and depicted various pictures of our implementation.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008. Available: https://bitcoin.org/bitcoin.pdf

[2] Bhabendu Kumar, Soumyashree S Panda, Debasish Jena,"An Overview of Smart Contract and Use cases in Blockchain Technology", 9th International Conference on Computing, Communication and Networking Technologies (2018)

[3] Bayu Adhi Tama, Bruno Joachim Kweka, Youngho Park, Kyung-Hyune Rhee," A Critical Review of Blockchain and Its Current Applications", International Conference on Electrical Engineering and Computer Science (ICECOS) 2017

[4] Saman Adhami, Giancarlo Giudici, Stefano Martinazzi," Why do businesses go crypto? An empirical analysis of Initial Coin Offerings", Journal of Economics and Business

[5] https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md

[6]https://coinsutra.com/what-is-erc20-token/

[7] https://tokenmint.io/blog/erc-20-tokens-and-how-to-create-yours.html

[8] https://solidity.readthedocs.io/en/v0.5.3/introduction-to-smart-Scontracts.html