VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Digital Assignment – I

Topic	**:**	**Solidity in crowdfunding**
Name	**:**	**Rishabh Kumar Rai**
Regi No.	**:**	**20BKT0076**
Course Title	**:**	**Blockchain Ecosystem**
Couse Code	**:**	**BKT4001**

ABSTRACT:

Crowdfunding is a decentralized application based on Ethereum blockchain platform that allows users to invest money to the campaigns that interest them.

By using blockchain, we can make sure that the investors engage in low-risk support of new ventures and venture creators can gain more supporters globally making it easy for them to raise large amount of funds in minimal time. Especially in blockchain world at present, there are lot of projects created by individuals or small-distributed teams that want to raise funds by issuing tokens to the investors.

Crowdfunding platform simplifies the whole idea of raising capital with help of global public that might be interested in the campaign for an incentive that is profitable to the investor.

INTRODUCTION

A new business venture often requires funding to develop and market its products, CrowdFunding is the collection and use of funds from various individuals and organizations to finance the business venture.

Nowadays, crowdFunding is facilitated by social media and crowdfunding websites, in which investors can fund small amounts to ventures they are interested in. Some of the current popular platforms for crowdfunding include Kickstarter, GoFundMe and Indiegogo.

The investors in crowdfunding usually get a reward, which may be gifts or products or equity-based.

The major disadvantage of such platforms is that one has to abide by the rules set by that platform, which essentially verifies and validates every funding transaction, and also must pay the fees that is needed to be paid according to the platform.

The major issues with the established crowdfunding platforms are that they are centralized bodies controlled by a corporation charging high fees and influencing campaigns.

With the support of Blockchain :

- Decentralization is achieved, hence there is no more need for reliance on a single authority or platform, this also reduces fees that are involved in traditional crowdfunding platforms.
- Outreach for investors is increased, as anyone with an internet connection can invest.
- Tokenization of assets changes the way we own and use assets, and provides new ways of rewards to investors.

Blockchain crowdfunding is a purer form of crowdfunding as it removes any intermediaries between the backers and the start-up .

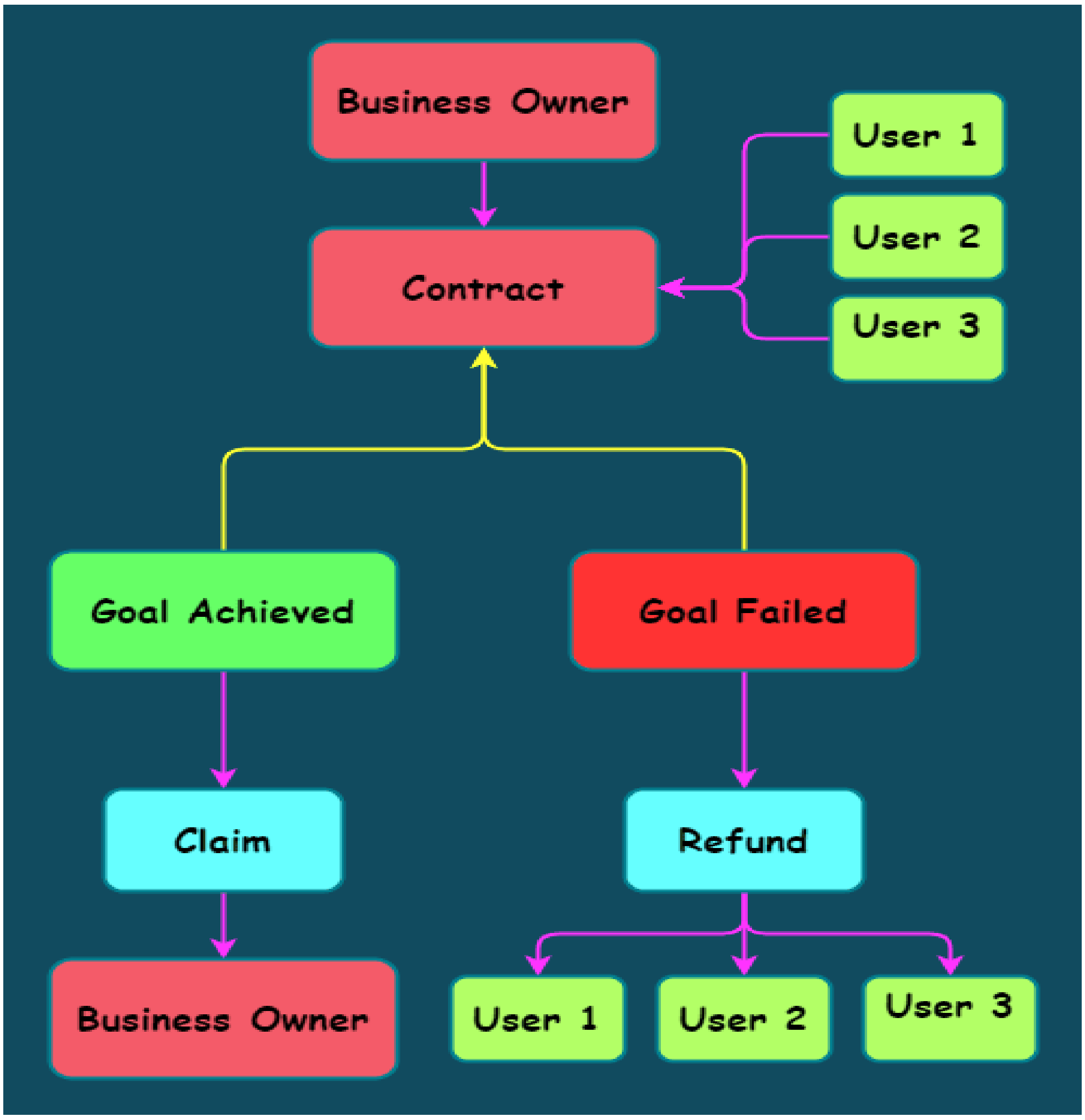
What is Crowdfund

Crowdfunding is the practice of using modest sums of money from a large number of people to finance a new business endeavor.

Let's say I want to open a modest tea shop, but I lack the resources or cash to do so. What then can I do?
Either I can get a loan from a friend, my family, or the bank, or I may ask someone to give me money and work with me on a project. Crowdfunding will be used if the project is too large for a single investor to invest in. I will solicit money from anyone in exchange for shares of my business, and each investment will possess a part in the business.

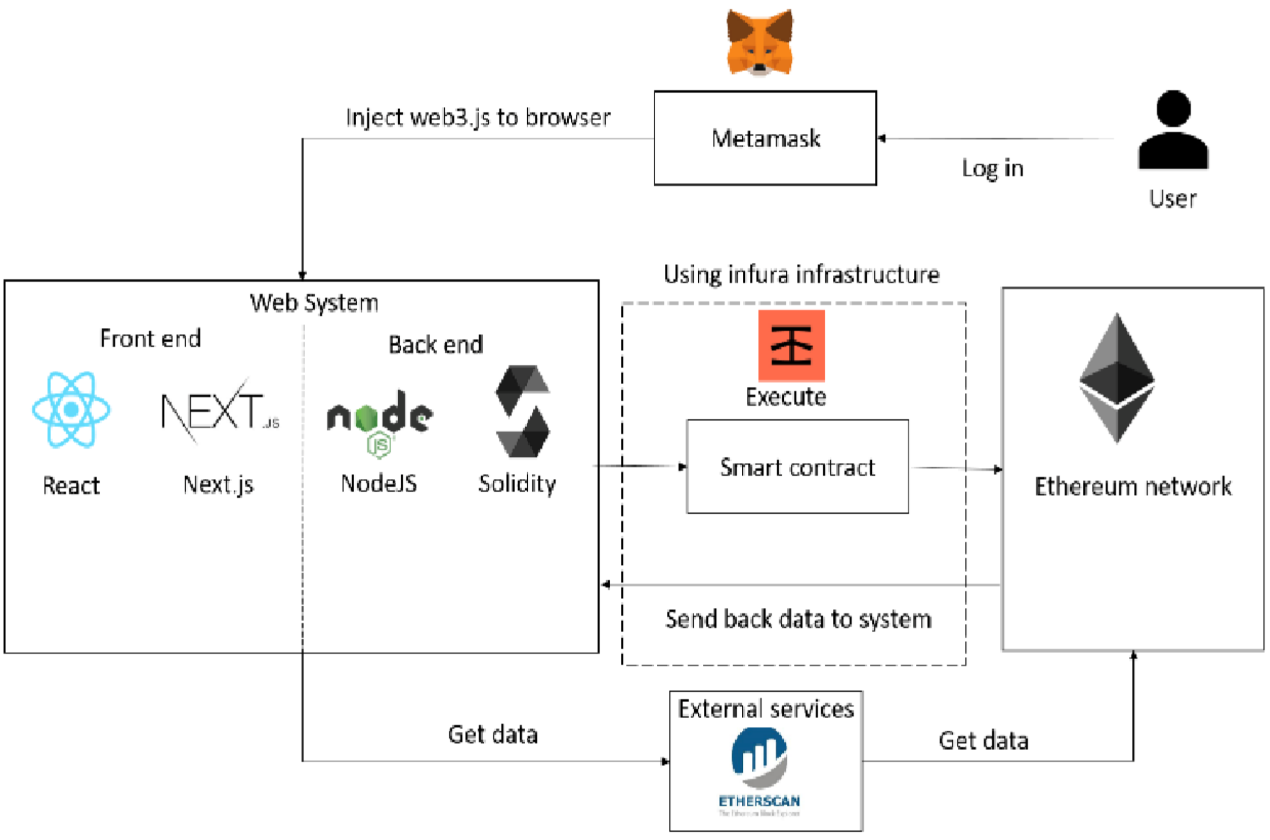
How does it work?

Look at this illustration of the structure we plan to build first.



- The business owner calls the contract to launch a campaign using several justifications, including the number of tokens that must be raised for the campaign, its start timestamp, and its end timestamp.
- As long as the campaign has not already begun, the business owner may cancel it at any time.
- By using the "pledge" function and entering the "id" of the campaign along with the number of tokens to be pledged, users can donate their tokens to a particular campaign.
- As long as the campaign is still active, they can also withdraw their previously pledged tokens.
- After the campaign is over, one of the two possible results is -
  - A campaign is considered successful when it raises the required number of tokens and meets the business owner's minimum requirements. In this scenario, the business owner can call the "claim" function to withdraw all the tokens.
  - If not enough tokens are pledged, which is the other scenario where a campaign fails, pledgers can withdraw their tokens from the contract by using the "withdraw" function.

ARCHITECTURE



Application Of Blockchain Technology in Crowdfunding

Crowdfunding is a critical utility particularly for small market enterprises as the new venture amidst a pervasive threat of employment crisis and insecurity. It is thus vital for governments to facilitate access to funds by small enterprises. The commonality of legislation across the European Union is one factor that enhances economic growth and the rise of new businesses. Notwithstanding the favorable environment of the EU, crowdfunding has not been very successful in the region. The tenacious growth of the sector further illustrates that the practice is a necessity that entrepreneurs expect to raise the capital to operate. Traditional crowdfunding in the EU has been thwarted by concerns of malpractices such as money laundering, information asymmetry, and fraud that prompts legislative restrictions on the fundraising activities. Nevertheless, blockchain technology is a tool that provides immense hope for a revival of crowdfunding across the world. The technology is a revolutionary and disruptive innovation targeting the reduction of bureaucracy and regulation without compromising legal provisions on business conduct. The blockchain technology provides a distributed public ledger that enhances transparency such that participants can conduct affairs without concerns of imposition over the internet. Most importantly, blockchain technology eliminates information asymmetry in its entirety thus suiting every stakeholder’s needs for proof of authenticity

Implementation of a Crowdfunding Decentralized Application on Ethereum

The goal of this paper was to analyze inefficiencies of the crowdfunding market and address them by combining smart contracts with smart property in a robust and trustless decentralized application. This has been achieved by developing an equity crowdfunding decentralized application that can be ran by any user willing to raise and manage equity financing. Its architecture is designed to be exible, so that users may ne-tune their preferences and use it as a base to attach additional functionalities suited to their needs. The developed decentralized crowdfunding application generates a secure crowd-sale campaign connected to a decentralized organization including tradable equity shares, voting rights, and transparency of transactions. This paper illustrates the convenience nature of Solidity's programming language and shows by example that nancial products have the potential to be decentralized with smart contracts. This small contribution to the world of information systems and microeconomics can be taken as a base for further research and developments of decentralized financial tools for equity fundraising.

Role of Blockchain Technology in Crowdfunding (International Banking and Finance)

Crowdfunding is a new and innovative method for funding various kinds of ventures, wherein individual founders of the ventures can request for funds. The ventures may be working for profit motive, cultural or social. The funds are usually given in return for future products or equity. It includes the use of internet social media platforms to connect investors with entrepreneurs in order to raise capital for various kinds of ventures in return for compensation. The Internet and social media became new platforms that emerged. Social media and the internet play a vital role in raising funds for entrepreneurs and other non-profit organizations. The paper shall first deal with the role of technology in crowdfunding, followed by the various crowdfunding platforms that have emerged in recent times. Blockchain is a unique, independent and a transparent system which keeps the transactions between parties transparent. Crowdfunding is based on the trust between the investors and stakeholders. The emergence of new technologies has great potential in crowdfunding organizations as well as individuals. Crowdfunding platforms using the blockchain technology increase the credibility of various projects and ventures and therefore attract huge funds from investors and donors.

Security Enhanced Crowdfunding Using Blockchain and Lattice Based Cryptosystem

Blockchain-based crowdfunding alters the usual approach to company finance. Generally, when people need to acquire funds to start a firm, they must first develop a strategy, statistical surveys, and models, and then offer their ideas to attract people or organisations. Banks, individual investors, and venture capital firms were among the sources of funding. The modern crowdfunding concept is based on three types of on-screen characters: the task initiator who presents the idea or venture to be financed, individuals or investors who invest in the idea, and a platform that connects these two characters to make the venture successful. It can be used to fund a wide range of start-ups and new concepts, such as inventive activities, medical improvements, travel, and social commercial enterprise projects.This work presents a practical implementation of a crowdfunding application that is secured by a lattice based cryptosystem for encryption of user data and zero-knowledge proof for the identification of application users. Additionally, machine learning has been used for prediction of campaign success for the benefit of fund contributors.

Smart Contract and Blockchain for Crowdfunding Platform

Smart Contracts based on Blockchain technology are very suitable to be implemented in 3 dominant crowdfunding process schemes. Beside increasing trust due to blockchain, smart contracts also shorten the main process in fundraising. Implementation of Smart Contract based on Blockchain technology, requires high-cost if the organization takes the initiative to implement this technology using their own resources. In general, smart contract service providers use cryptocurrency, where not all governments legally recognize the use of these currencies. In the situation of the Covid-19 plague, many organizations are raising funds to help local governments to obtain additional sources of funds that will be distributed to those in need. Trust is an important factor for the parties involved in raising these funds, both in terms of the funder, the service provider of fundraising platform and even the fundraiser. On this occasion, the author tries to analyze how to implement blockchain technology and smart contracts in the dominant schemes of crowdfunding process. the results of this study indicate that blockchain-based smart contracts can be applied to the dominant schemes of crowdfunding process.

Smart Contract

First, we specify our license and the Solidity version we intend to use. The version we are working on is indicated by pragma solidity; take note that our compiler version must match for there to be no error signs.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.7;
```

To help the contract call the transfer and transferFrom functions from the ERC-20 Token Contract, we will now construct an Interface for our ERC-20 Tokens.

```
interface IERC20 {
    function transfer(address, uint) external returns (bool);

    function transferFrom(
        address,
        address,
        uint
    ) external returns (bool);
}
```

We now declare some state variables to begin our smart contract.

- **Campaign** - A struct for storing data on the campaign's creator, aim, start time, end time, and the pledged tokens.
- **token** - to refer to the ERC-20 Token contract used for token transfers.
- **count** - to keep track of campaigns.
- **maxDuration** - Specify the maximum duration a campaign can be hosted.
- **campaigns** - a mapping that connects campaigns to their ID
- **pledgedAmount** - a mapping to link the user's address and the number of tokens they pledged, and another mapping to link the campaign id.

```
struct Campaign {
        address creator;
        uint goal;
        uint pledged;
        uint startAt;
        uint endAt;
        bool claimed;
    }

    IERC20 public immutable token;
    uint public count;
    uint public maxDuration;
    mapping(uint => Campaign) public campaigns;
    mapping(uint => mapping(address => uint)) public pledgedAmount;
```

Next, we designate some actions as events that will occur if a campaign is started, a token is pledged or unpledged, a campaign is cancelled, or a token is claimed or withdrawn.

```
event Launch(
        uint id,
        address indexed creator,
        uint goal,
        uint32 startAt,
        uint32 endAt
    );
    event Cancel(uint id);
    event Pledge(uint indexed id, address indexed caller, uint amount);
    event Unpledge(uint indexed id, address indexed caller, uint amount);
    event Claim(uint id);
    event Refund(uint id, address indexed caller, uint amount);
```

Next, we set up our contract's constructor, which requires the address of the ERC-20 token and the maximum time a campaign can be hosted, as its two inputs.

```
constructor(address _token, uint _maxDuration) {
        token = IERC20(_token);
        maxDuration = _maxDuration;
    }
```

Our first function, "launch," which accepts the campaign's goal, the start timestamp, and the end timestamp, is now defined.

Before starting the campaign, we first perform some checks.

1. We determine whether the commencement time exceeds the current time.
2. We make sure the end time is later than the beginning time.
3. Finally, we make sure the campaign does not go beyond its maximum time.

After that, we raise our count variable. The campaign information is then stored in the campaigns mapping, with the count variable serving as the key and a struct as the value.

And finally emit the Launch event

```
function launch(uint _goal, uint32 _startAt, uint32 _endAt) external {
        require(_startAt >= block.timestamp,"Start time is less than current Block Timestamp");
        require(_endAt > _startAt,"End time is less than Start time");
        require(_endAt <= block.timestamp + maxDuration, "End time exceeds the maximum
Duration");

        count += 1;
        campaigns[count] = Campaign({
            creator: msg.sender,
            goal: _goal,
            pledged: 0,
            startAt: _startAt,
            endAt: _endAt,
            claimed: false
        });

        emit Launch(count,msg.sender,_goal,_startAt,_endAt);
    }
```

Next, we define a function called cancel, which allows the campaign's creator to end the campaign provided that they are the campaign's creator and that the campaign has not yet begun.

```
function cancel(uint _id) external {
        Campaign memory campaign = campaigns[_id];
        require(campaign.creator == msg.sender, "You did not create this Campaign");
        require(block.timestamp < campaign.startAt, "Campaign has already started");

        delete campaigns[_id];
        emit Cancel(_id);
    }
```

We have now developed our pledge feature, which asks for the campaign id and the number of tokens that need to be pledged.

In order to transfer tokens from the user to the smart contract, we first run basic tests, such as determining whether the campaign has begun or concluded. Next, we use the token variable, which corresponds to the IERC interface and call the transferFrom function to transfer tokens from the user to the smart contract.

By increasing the amount of tokens offered by the campaign and storing the number of tokens pledged by the user, we modify the state variables of the contract.

And then we emit the Pledge event.

```
function pledge(uint _id, uint _amount) external {
        Campaign storage campaign = campaigns[_id];
        require(block.timestamp >= campaign.startAt, "Campaign has not Started yet");
        require(block.timestamp <= campaign.endAt, "Campaign has already ended");
        campaign.pledged += _amount;
        pledgedAmount[_id][msg.sender] += _amount;
        token.transferFrom(msg.sender, address(this), _amount);

        emit Pledge(_id, msg.sender, _amount);
    }
```

We provide a function called unpledge that removes the tokens that a user has pledged, just as the pledge function.

```
function unPledge(uint _id,uint _amount) external {
        Campaign storage campaign = campaigns[_id];
        require(block.timestamp >= campaign.startAt, "Campaign has not Started yet");
        require(block.timestamp <= campaign.endAt, "Campaign has already ended");
        require(pledgedAmount[_id][msg.sender] >= _amount,"You do not have enough tokens
Pledged to withraw");

        campaign.pledged -= _amount;
        pledgedAmount[_id][msg.sender] -= _amount;
        token.transfer(msg.sender, _amount);

        emit Unpledge(_id, msg.sender, _amount);
    }
```

The creator can claim all of the tokens raised for the campaign with the help of a claim function that we define next if the following criteria are met.

1. the campaign's originator is the one who called the function.
2. The campaign has come to a close.
3. The objective has been surpassed by the quantity of tokens raised (campaign succeded)
4. The tokens have not yet been redeemed.

```
function claim(uint _id) external {
        Campaign storage campaign = campaigns[_id];
        require(campaign.creator == msg.sender, "You did not create this Campaign");
        require(block.timestamp > campaign.endAt, "Campaign has not ended");
        require(campaign.pledged >= campaign.goal, "Campaign did not succed");
        require(!campaign.claimed, "claimed");

        campaign.claimed = true;
        token.transfer(campaign.creator, campaign.pledged);

        emit Claim(_id);
    }
```

In the event that the campaign is unsuccessful, we create a refund function that allows users to withdraw their tokens from the contract.

```
function refund(uint _id) external {
        Campaign memory campaign = campaigns[_id];
        require(block.timestamp > campaign.endAt, "not ended");
        require(campaign.pledged < campaign.goal, "You cannot Withdraw, Campaign has
succeeded");

        uint bal = pledgedAmount[_id][msg.sender];
        pledgedAmount[_id][msg.sender] = 0;
        token.transfer(msg.sender, bal);

        emit Refund(_id, msg.sender, bal);
    }
```

This is how a crowd funding contract is created, allowing users to establish campaigns and donate tokens to various projects.