SUBJECT: MIDTERM REPORT

fundETH: Blockchain Based Crowdfunding System

April 7, 2022

1 GROUP MEMBERS

S.No.	Name	Enrollment No.	Email-Id
1	Naman Jain	19114056	naman_j@cs.iitr.ac.in
2	R Chinmay	19114067	r_c@cs.iitr.ac.in
3	Raghav Somani	19114068	raghav_s@cs.iitr.ac.in

2 PROJECT DESCRIPTION

We are building a Ethereum blockchain based crowdfunding website, where the users can open new fundraising campaigns or contribute to any already open fundraisers available on the website. All the transactions occur on the blockchain network, which ensures privacy and security. We have also built a simple user-friendly website so that users can perform these operations without understanding the intricacies of blockchain network and smart contracts deployed. The functioning of our website will be explained in later sections.

2.1 BASIC TERMINOLOGIES

1. **Blockchain**: A digital ledger that stores the transactions in a form of secure list of blocks and this is distributed across the entire network to make it secure and decentralized.

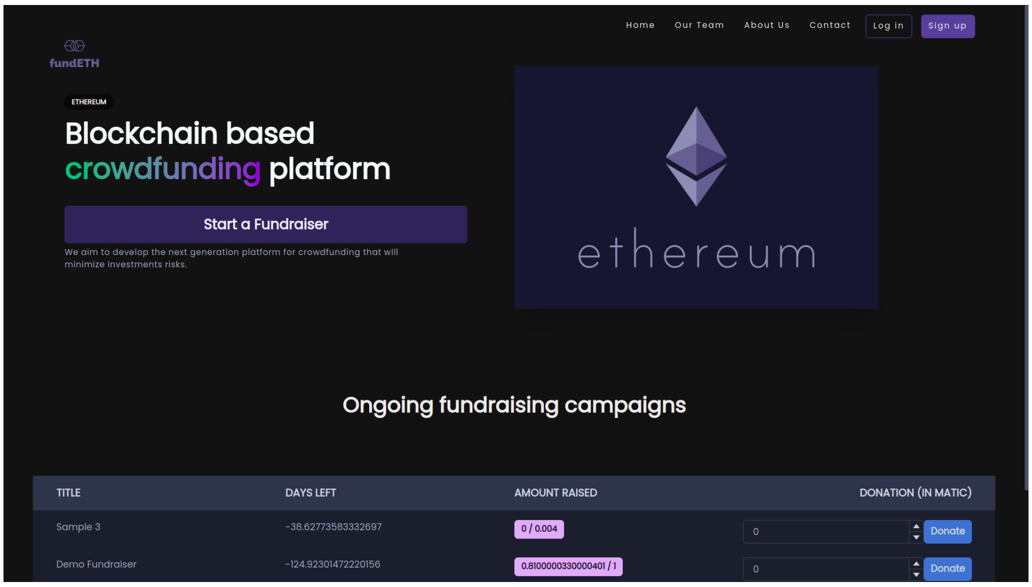
2. **Ethereum** [1]: It is open-source and one of the most popular blockchains presently, which supports smart contracts to be built on its network.

3. **Smart Contracts** [2]: They are programs stored on blockchain, which anyone can request to execute by making a transaction request.

3 Software Architecture

Note: For testing purposes our website currently operates on Polygon Mumbai TestNet using MATIC as its underlying Cryptocurrency but it can be easily ported to Ethereum Mainnet for deployment as our code is Ethereum Virtual Machine (EVM) compatible.

3.1 Front-End Website



Blockchain based crowdfunding platform

We aim to develop the next generation platform for crowdfunding that will minimize investments risks.

Ongoing fundraising campaigns

TITLE	DAYS LEFT	AMOUNT RAISED	DONATION (IN MATIC)
Sample 3	-38.62773583332697	0 / 0.004	0
Demo Fundraiser	-124.92301472220156	0.8100000330000401 / 1	0

We have created a website using React.js, NEXT.js, Chakra UI (for User Interface Components) and Firebase (for authentication). Our Website supports following actions:

1. Campaign Creators will have to first login into the website and then they can create a fundraising campaign by filling required details. Upon Validation their campaign will be added to list of ongoing campaigns.

2. We display a list of ongoing fundraising campaigns and their details on our website homepage.

3. Donor can select any ongoing campaign and donate CryptoCurrency to that campaign using his/her MetaMask wallet or any other Crypto wallet.

[1] https://ethereum.org/en/
[2] https://ethereum.org/en/developers/docs/smart-contracts/

3.2 WEB3.JS

For all the mentioned actions/tasks our website needs to communicate with BlockChain Network to get details regarding active campaigns, Creating a new campaign and Donating cryptocurrency to a active Campaign. Also there is a need for user to connect their crypto-wallets with our website for processing all these transactions. All these functionalities are provided using Web3 library of javascript in a file named Web3.js. Our Web3.js file performs the following tasks:

1.
   ```
   let provider = window.ethereum;

   if(typeof provider != "undefined"){
       web3 = new Web3(provider);
       ethereum.enable();
       web3.eth.getAccounts((accounts) => callback(accounts));
   }
   ```

   This code checks for any available Crypto wallets and connects with that wallet if it is available. The last line queries the wallet for available Crypto Accounts and returns a list of all available accounts of user.

2.
   ```
   const crowdfund = new web3.eth.Contract(contract_abi, address);

   crowdfund.methods.startProject(
     title, details, deadline,
     web3.utils.toWei(amount, "ether")
   )
   .send({
     from: accounts[0],
   });
   ```

   Here *'contract_abi'* is a javascript dictionary that defines the functions available in contract and their input output specifications. Here *'address'* is the address of our Smart Contract in hex (More on smart contracts in next section). This code starts a new crowdfunding project with the details provided by the first user account by default.

3.
   ```
   crowdfund.methods.returnAllProjects().call().then((projects) => {
     projects.forEach((projectAddr) => {
       const project = new web3.eth.Contract(project_abi, projectAddr);

       project.methods.getDetails().call().then((result) => {
           setCampaigns(active => [...active, projectData])
       });
     });
   });
   ```

Similarly, here we first get all active projects from the crowdfund Contract. Then we request all individual project contracts to get details about those projects and add it to our list of active Fundraising Campaigns.

4.
```
const project = new web3.eth.Contract(project_abi, token.projectAddr);

project.methods.contribute().send({
  from: accounts[0],
  value: web3.utils.toWei(donationAmount, "ether"),
});
```

   Finally, above code is used to call contribute function that is used to donate the money from donor's account to campaign creator's account. This request has to be accepted in user's crypto wallet and validated by blockchain to successfully occur.

3.3 Smart Contract:

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific address on the Ethereum blockchain. Smart Contracts can be written in developer-friendly languages. For e.g: Solidity, Vyper and Yul/Yul+ etc.

3.3.1 CrowdFunding Contract:

We have written two Smart Contracts in Solidity language, the 'Crowdfunding' contract is created once and it is used to create new instances of 'Project' contracts using startProject() which emits a ProjectStarted event. It also stores list of projects in projects array which can be returned using returnAllProjects() function. Given below is the skeleton code of Crowdfunding contract:

```
pragma solidity 0.8.0;
import "@openzeppelin/contracts/utils/math/SafeMath.sol";

contract CrowdFunding {
    using SafeMath for uint256;
    Project[] private projects;

    event ProjectStarted(address contractAddress, address projectStarter,
                         string projectTitle, string projectDesc,
                         uint256 deadline, uint256 goalAmount);

    function startProject(string calldata title, string calldata description,
        uint durationInDays, uint amountToRaise
    ) external {}
    function returnAllProjects() external view returns(Project[] memory){
        return projects;
    }
}
```

3.3.2 Project Contract:

The 'Project' contract stores state of individual fundraising projects and can be used to perform transactions on a project. Given below is outline of all major functions and state variables in Project contract:

```
contract Project {
    using SafeMath for uint256;

    enum State {Fundraising, Expired, Successful}
    address payable public creator;
    uint public amountGoal, completeAt, raiseBy;
    uint256 public currentBalance;
    string public title, description;
    State public state = State.Fundraising; // initialize on create
    mapping (address => uint) public contributions;

    event FundingReceived(address contributor, uint amount, uint currentTotal);
    event CreatorPaid(address recipient);

    constructor( // add parameters here){
        // Set states here
    }

    function contribute() external inState(State.Fundraising) payable {}
    function checkIfFundingCompleteOrExpired() public {}
    function payOut() internal inState(State.Successful) returns (bool) {}
    function getDetails() public view returns (// return values here) { }
}
```

We have called these functions from our Web3 code in website to get data or perform actions. Also, we have use **hardhat** to compile and deploy our contracts and **Alchemy** to track our smart contract usage statistics.

4 Future Work

Under normal conditions our website is working fine and all functionalities are active. But there are some exceptions and error conditions that need to be appropriately handled. We aim to address these exceptions and error conditions before the final submission, refactor the code and add some more functionalities to our website.

5 Acknowledgement

We thank Professor Sandeep Kumar of the Department of Computer Science and Engineering IIT Roorkee, for providing us with an opportunity to work on this project.

6 References

1. Ethereum Whitepaper - `https://ethereum.org/en/whitepaper/`

2. *Md Nazmus Saadat, Syed Abdul Halim, Husna Osman, Rasheed Mohammad Nassr, Megat F. Zuhairi.* **Blockchain based crowdfunding systems** in Indonesian Journal of Electrical Engineering and Computer Science, Vol 15, July 2019. `https://pdfs.semanticscholar.org/f9e0/70e981f30907e15556fc7959f5aeb7eeb969.pdf`

3. Ethereum docs - `https://ethereum.org/en/developers/docs/`

4. Smart contract tutorial - `https://medium.com/openberry/creating-a-simple-crowdfunding-dapp-with-ethereum-solidity-and-vue-js-69ddb8e132dd`