A Project Report on

DECENTRALIZED CROWDFUNDING

Submitted in partial fulfillment of the

requirement for the award of the degree of

B-TECH-CSE

Under The Supervision of
Name of Supervisor **:**
V.Arul Sir

Submitted By

ABHISHEK RAY-18SCSE1010372
SANKET KUMAR PANDEY-18SCSE1010616

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER-2021

Abstract

In today's world, block chain-based systems are in demand across various industries, because of its secure, trusted, and decentralized network as well as for being more efficient than the traditional methods. However, the traditional ways these days are facing a lot of issues and challenges because of the complex and less secure network. Block chain network integration overcomes the problems faced by traditional methods across industries. The Block chain integrated network provides benefits such as increased security, increased transparency, increased efficiency and decreased chances of fraud. Although the block chain-based systems provide various benefits, due to lack of knowledge about this technology, the implementation rate is low. In this work, we have highlighted the distinction between the traditional crowd funding Platform as well as block chain network-based crowd funding platform and the benefits of implementing block chain network in other sectors. This work highlights the issues and challenges faced by the industries, as mentioned earlier, by using the traditional methods as well as the solutions to the problems provided by the block chain network-based systems to those industries. This work helps the people to understand the benefits of block chain network-based systems in their respective industries as well as execute it to improve the transparency, efficiency, and security of the system altogether.

Introduction

Crowdfunding could be a methodology of raising capital through the collective effort of friends, family, customers, and individual investors. This approach faucets into the collective efforts of an outsized pool of individuals— primarily on-line via social media and crowdfunding platforms—and leverages their networks for bigger reach and exposure. Crowdfunding is actually the other of the thought approaches to business finance. historically, if you wish to lift capital to begin a business or launch a brand new product, you'd have to be compelled to clean up your business arrange, marketing research, and prototypes, so look your plan around to a restricted pool or moneyed people or establishments. These funding sources enclosed banks, angel investors, and working capital corporations, very limiting your choices to a number of key players. you'll think about this fundraising approach as a funnel, with you and your pitch at the wide finish and your audience of investors at the closed finish. Fail to purpose that funnel at the proper capitalist or firm at the proper time, and that’s it slow and cash lost. Crowdfunding platforms, on the opposite hand, flip that funnel on-end. By supplying you with, the bourgeois, one platform to create, showcase, and share your pitch resources, this approach dramatically streamlines the normal model. historically, you’d pay months separation through your personal network, vetting potential investors, and defray your own time and cash to induce before them. With crowdfunding, it’s a lot easier for you to induce your chance before a lot of interested parties and provides them a lot of ways to assist grow your business, from finance thousands in exchange for equity to contributory $20 in exchange.

Literature Survey

The conventional method used by crowdfunding websites has a major drawback. It does not allow a contributor to have any control over the money they have contributed. This results in frauds and scams. The proposed method addresses this problem and provides contributors with control over the money they have contributed. Log of all the transactions happening in the network is called a ledger. Blockchain maintains a global ledger and each node in the network has a copy of this global ledger called the private ledger. Since every node has a copy of ledger so no node can perform malicious activity. Interaction of global and private ledger in Blockchain is depicted in file. Ethereum is an implementation of blockchain and extends its functionality using smart contracts. Smart contracts can be used to implement logic in blockchain secured environments. Thus using blockchain and smart contracts, a new system has been designed to solve the problem faced by existing crowdfunding websites. It is a decentralized network whereas the traditional method uses a centralized approach. Decentralized approach eliminated the chances of a single point of failure. Thus the proposed system is robust. In the convention The signed data file can then be verified using the public key of the sender which is easily available and thus the authenticity of the data file is maintained. The digital signature ensures that the data is being sent by that particular person only and the person also cannot deny that

Methodology

Traditional Crowdfunding Concept Most ancient business funding takes one in all 3 forms: self funding, bank funding, or working capital. The problem is that for many folks, self-funding is implausibly restricted. Bank funding needs having AN existing business with sensible revenues and income. And venture fund capital nearly invariably needs a product or service that has mass attractiveness. This makes ancient funding terribly restricted and laborious to induce for newer businesses. It will inhibit growth even for products and services with immense potential. Crowdfunding permits businesses with very nice product and repair ideas to lift funds from regular folks in tiny investment amounts. Once it works, it will very offer your business an enormous boost. firms like Kickstarter, Indiegogo, and Crowdfunder were among the earliest to create it well-liked. One drawback is that even with crowdfunding, the model remains very inefficient. In step with Kickstarter, seventy eight of campaigns that raise 2 hundredth of their goal ultimately become absolutely funded, whereas Martinmas of comes end having ne'er received any funding the least bit. This brings United States of America to however blockchain is dynamical the crowdfunding landscape

Problem Statement

- Centralized Croudfunding Has Many Downsides Although the concept of Croudfunding offers many opportunities on paper, one can't ignore the drawbacks either. The fast way of raising money and different take on pitching or business has helped many companies become successful . More importantly , crowd funding can help generate community feedback before launching products or new ideas. While all of those advantages are
- positive, one has to consider the drawbacks of traditional crowdfunding too. The first hurdle to overcome is raising money successfully.

Tools and Technology

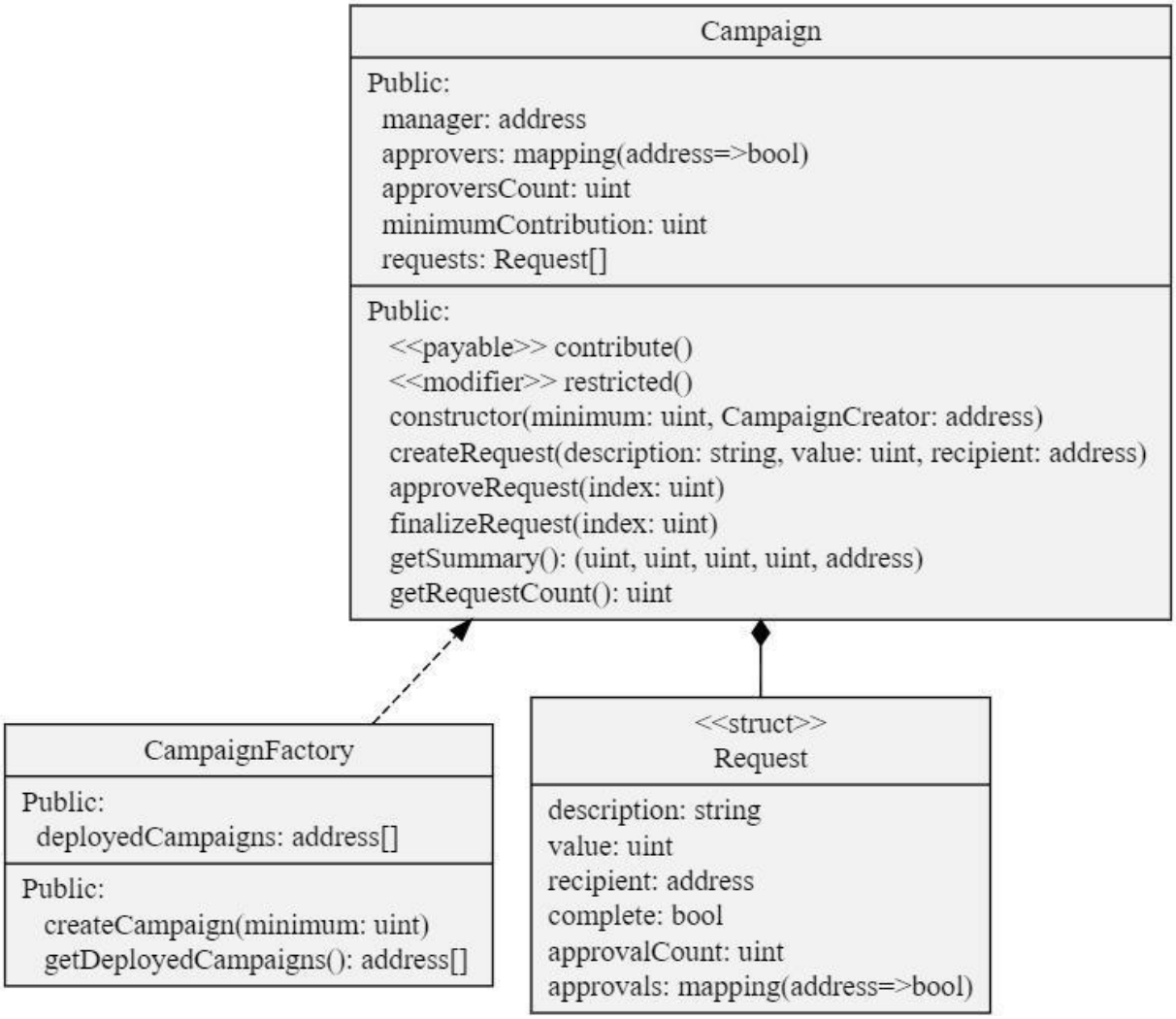
- **FRONT-END**

✓ React-js
✓ Javascript
✓ Sematic-UI(For css , color button ,card , navigation , etc)
✓ next-js for server side routing

- **BACK-END**

✓ SmartContract (On Ethereum e.g. solidity

UML Diagram


Campaign
Public:
manager: address
approvers: mapping(address=>bool)
approversCount: uint
minimumContribution: uint
requests: Request[]
Public:
<<payable>> contribute()
<<modifier>> restricted()
constructor(minimum: uint, CampaignCreator: address)
createRequest(description: string, value: uint, recipient: address)
approveRequest(index: uint)
finalizeRequest(index: uint)
getSummary(): (uint, uint, uint, uint, address)
getRequestCount(): uint
CampaignFactory
Public:
deployedCampaigns: address[]
Public:
createCampaign(minimum: uint)
getDeployedCampaigns(): address[]
<<struct>>
Request
description: string
value: uint
recipient: address
complete: bool
approvalCount: uint
approvals: mapping(address=>bool)

Implementation of the project:-

```
pragma solidity ^0.4.25;

contract CampaignFactory{
address[] public deployedCampaigns;
function createCampaign(uint minimum)public{
address newCampaign=new Campaign(minimum,msg.sender);
deployedCampaigns.push(newCampaign);
}
function getDeployedCampaigns()public view returns(address[]){
return deployedCampaigns;
}
}


//not voting means no vote
contract Campaign{
address public manager;
mapping(address=>bool)public approvers;
uint public approversCount;
uint public minimumContribution;
modifier restricted(){
require(msg.sender==manager);
_;
}

struct Request{
string description;
uint value;
address recipient;
bool complete;
uint approvalCount;
//approvalCount only count number of yes votes
mapping(address=>bool) approvals;
}
```

```
Request [] public requests;

constructor(uint minimum,address CampaignCreator) public{
manager=CampaignCreator;
minimumContribution=minimum;
}
function contribute() public payable{
require (msg.value >= minimumContribution);
approvers[msg.sender]=true;
approversCount++;
}


function createRequest(string description,uint value,address recipient)
public restricted
{
Request memory newRequest=Request({
description:description,
value:value,
recipient:recipient,
complete:false,
approvalCount:0
});
//Request(description,value,recipient,false);
requests.push(newRequest);


}
function approveRequest(uint index) public
{
Request storage request=requests[index];
require(approvers[msg.sender]);
require(!request.approvals[msg.sender]);
request.approvals[msg.sender]=true;
request.approvalCount++;
}
function finalizeRequest(uint index) public restricted
```

```
{




Request storage request=requests[index];
require(request.approvalCount>(approversCount/2));
require(!request.complete);
request.complete=true;
//uint etherValue = request.value/(1 ether);
//uint ether_to_wei=request.value*(1 ether);
request.recipient.transfer(request.value);

}
function getSummary() public view returns(uint,uint,uint,uint,address){
return(
minimumContribution,
address(this).balance,
requests.length,
approversCount,
manager
);
}
function getRequestCount() public view returns(uint){
return requests.length;
}
}
```

For testing our project we have unit test as

Campaign.test.js

```
//here test may be related to ethereum side and  other might be related to web appplication side!!
const assert = require('assert');
const ganache = require('ganache-cli');
const Web3=require('web3');
const web3=new Web3(ganache.provider());

const compiledFactory=require('../ethereum/build/CampaignFactory.json');
const compiledCampaign=require('../ethereum/build/Campaign.json');

let accounts;
//factory=>  refrence to the deployed instance of the factory that we gonna make
let factory;
let campaign;
let campaignAddress;
beforeEach(async()=>{
//mostly we need instance of a Campaign
//so rather we use factory inside of every it() block to create seperate instance of a Campaign we use  Campaign
  accounts=await web3.eth.getAccounts();
  factory=await new web3.eth.Contract(JSON.parse(compiledFactory.interface))
  .deploy({data:compiledFactory.bytecode})
  .send({from:accounts[0],gas:1000000});
   await factory.methods.createCampaign(10).send({
    from:accounts[0],
    gas:1000000
  });
  //it is es2016 syntax
// const addresses=  await factory.methods.getDeployedCampaigns().call();
// campaignAddress=addresses[0];
//use es5 syntax
  [campaignAddress]=await factory.methods.getDeployedCampaigns().call();
//create actual  instance of campaign
//we use Web3 for instruct that will acess to address at  campaignAddress

campaign=await
new web3.eth.Contract(JSON.parse(compiledCampaign.interface),campaignAddress);
});
```

```
describe('Campaigns',()=>{

  it('deploys a factory and a campaign',()=>{
    assert.ok(factory.options.address);
    assert.ok(campaign.options.address);
  });

  //test campaign has manager of address mark as  account[0],because we
  //use account[0] to create instance of a campaignAddress
  //means manager os this campaign have account[0]
  //caller=> person who call createCampaign() method
  it('marks caller as the campaign manager',async()=>{
    const campaign_manager=await campaign.methods.manager().call();
    assert.equal(campaign_manager,accounts[0]);
  });


  it('Allows people to contribute money and mark them as approver',async()=>{

    await campaign.methods.contribute().send({value:200,from:accounts[1]});
    const is_Contributor=await campaign.methods.approvers(accounts[1]).call();
    //is_Contributor=false;
    //assert(is_Contributor);
    assert(is_Contributor);
});

  it('Requires a minimum contribution',async()=>{

try {
  await campaign.methods.contribute().send({value:100,from:accounts[1]});
//we use assert(false) because yeh wali line(campaign.methods.contribute()
//.send({value:10,from:accounts[1]});) chalne ke baad hum seedha catch block ke
//andar  aa jayenege and agar test fail hua it means try block ke andar hi the
  assert(false);
}
catch (error) {
  assert(error);
}
```

```
  });
it('only manager has the ability to make a payment request',async()=>{
  await
campaign.methods.createRequest("batteries",100,accounts[1]).send({from:accounts[0],gas:1000
000});
  const request=await campaign.methods.requests(0).call();
  assert.equal('batteries',request.description);

});

//now contribute,createRequest,approveRequest,finalizeRequest
it('process requests',async()=>{

  await campaign.methods.contribute().send({
    from:accounts[0],
    value:web3.utils.toWei('10','ether')
  });

  await campaign.methods
  .createRequest('A',5,accounts[1])
  .send({from:accounts[0],
       gas:1000000});

    await campaign.methods.approveRequest(0).send({from:accounts[0],gas:1000000});

    await campaign.methods.finalizeRequest(0).send({from:accounts[0],gas:1000000});

    //balance is a string that represent the ammounts of money that account has in wei
    let balance=await web3.eth.getBalance(accounts[1]);
    //now balance is in ether
    balance=web3.utils.fromWei(balance,'ether');
    //balance have sting value just in float type
    balance=parseFloat(balance);
    //in real all ganache accounts have 100 ether and we transfer 5 ether to finalize
    //so we use 99.+5>104
    assert(balance>104);
});
});
```

This is how we route our pages -
***server.js ***

```
//We have to tell next js to use that routes file
//so purpose of server.js file is to manaually start up our next application
//and specifically tell it to use the routes that we defined inside routes.js file
const {createServer}=require('http');
const next = require('next')
const routes = require('./routes')
//new instance of next application is app
//dev=> development
//here dev specifies whether we are running our server in a production or a development mode

const app = next({dev: process.env.NODE_ENV !== 'production'})
const handler = routes.getRequestHandler(app);
app.prepare().then(()=>{

  createServer(handler).listen(3000,(err)=>{
    if(err) throw err;
    console.log('Ready on localhost:3000');
  })
});
```

and here how we add routes :
***routes.js ***

```
const routes = require('next-routes')();
//it will return the function so we use parenthesis ()
// routes.add('...','...')
//we will set different dynamic routes that we nedd inside of our application
routes
.add('/campaigns/new','/campaigns/new')
.add('/campaigns/:address','/campaigns/show')
.add('/campaigns/:address/requests','/campaigns/requests/index')
.add('/campaigns/:address/requests/new','/campaigns/requests/new');
module.exports=routes;
```

This is how we deploy contract Factory -

```
const HdWalletProvider=require('truffle-hdwallet-provider');
const Web3=require('web3');
const compiledFactory=require('../ethereum/build/CampaignFactory.json');
const provider=new HdWalletProvider(
  'jeans toast bone embody tortoise trophy often amazing split into robust fortune',

'https://rinkeby.infura.io/v3/45662a3729fa43678d13b210e60dee48'
);
const web3=new Web3(provider);
const deploy=async()=>{
  const accounts=await web3.eth.getAccounts();
  console.log('Attempting to deploy from account',accounts[0]);
  const factory=await new web3.eth.Contract(JSON.parse(compiledFactory.interface))
  .deploy({data:'0x'+compiledFactory.bytecode})
  .send({from:accounts[0],
        gas:1000000
  });
  console.log("Contract deployed to "+factory.options.address);
}

deploy();
provider.engine.stop();
```

This is how we connect to the web3 :

```
import Web3 from "web3";
//our code 1st get executed on the next server
//window is global variable that is availabe only browser
//window is not availabe on node js (whisch is where our server is currently running)
//so we don't acess window right now and it show error that => window is not defined
//mostly peope don't use metamask ,so how to cope up with these currentProvider(metamask)
//we directly link nest js to ethereum network and do some initial calls and we goind to do some data fetching
//before we produce HTML Document to send user browser,it doesn't matter whether or not user installed metamask
//it doesn't matter or not they even have acess to ethereum network because we have already fetch that data and
//send them HTML document with all information already contained inside of it
//by using next js which directly connect to ethereum network so they have to don't care about metamask and rinkeby and any other network

// const provider=window.web3.currentProvider;
// const web3=new Web3(provider);

// export default web3;
//this file exexute two times => 1. one time on the next js server to initially render our application
//2. second time inside the browser

//now we remove this error

let web3;
//to check our code is inside our browser (and metamask is availabe)  or not use typeof window
//if yes then it(typeof window ) return object otherwise it return undefinded
//to check metamask installed(or injected web3 ) use => window.web3
if (typeof window !== "undefined" && typeof window.web3 !== "undefined") {
  //we are inside the browser and metamask is running
  web3 = new Web3(window.web3.currentProvider);
} else {
  // we are on the server OR  user is not running metamask
  // here we use infura api  provider to making our own provider that is aceesing the network through URL

  const provider = new Web3.providers.HttpProvider(
    "https://rinkeby.infura.io/v3/45662a3729fa43678d13b210e60dee48"
```

```
  );
  web3 = new Web3(provider);
}
export default web3;
```

For getting contract factory artifacts and bytecode we have to run this :

```
//we always recompile our code(as in lottery.sol file ) it takes time (4-7 sec)
//to save that time ,we compile our code at one time ,and write output to the new
//file and then acess that compile version
const path = require("path");
const solc = require("solc");
//fs-extra improved version of require('fs') module
const fs = require("fs-extra");
const buildPath = path.resolve(__dirname, "build");
//delete all files which is under build folder
fs.removeSync(buildPath);
const campaignPath = path.resolve(__dirname, "contracts", "Campaign.sol");
const source = fs.readFileSync(campaignPath, "utf-8");
const output = solc.compile(source, 1).contracts;
//here output object contain two seperate object => 1. campaign compile 2. campaign_factory
compile
fs.ensureDirSync(buildPath);
for (let contract in output) {
  fs.outputJsonSync(
    path.resolve(buildPath, contract.replace(":", "") + ".json"),
    output[contract]
  );
}
```

Here is our main front end component :

1. header.js

```
import React from 'react';
import {Menu} from 'semantic-ui-react';
import {Link} from '../routes';
//if we use both Link tag and Menu.Item tag simultaneously then styles of these
//two components clashed with each other so rather usin <Menu.Item> tag we going
//to use only < Link> tag

//we know that Link tag will create automatic anchor tag(<a></a>)
//Link tag is a generic wrapper component that doesn't add any Html of it's own
//it's wraps its children with a click event handler ,so anyone clicks on any of it's
//children it automatically navigate the user around the page

export default () =>{
  return (
     <Menu style={{marginTop:'30px'}}>
       <Link route="/">
       <a className="item"><h3> RayCampaign</h3></a>
       </Link>

       <Menu.Menu position='right'>
       <Link route="/">
       <a className="item"><h3>All Campaigns</h3></a>
       </Link>
       <Link route="/campaigns/new">
       <a className="item"><h3> + </h3></a>
       </Link>


       </Menu.Menu>
     </Menu>

);
}
```

2. layout.js

```
import React from 'react';
```

```
import {Container} from 'semantic-ui-react';
import Header from './header.js';
import Head from 'next/head';
export default (props)=>{
return (
  <Container>
  <Head>
  <link rel="stylesheet" href="//cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.css"
/>
  </Head>
    <Header/>
    {props.children}
  </Container>
);
}
```

3. RequestRow.js

```
import React ,{Component} from 'react';
import {Table,Button} from 'semantic-ui-react';
import  web3 from '../ethereum/web3';
import Campaign from '../ethereum/Campaign';
import {Router} from '../routes';
class RequestRow extends Component{
  state={
    loading1:false,
    loading2:false
  }
  onApprove=async()=>{
    this.setState({loading1:true});
    const campaign=Campaign(this.props.address);
    const accounts=await web3.eth.getAccounts();
    try{
    await campaign.methods.approveRequest(this.props.id).send({from:accounts[0]});}
    catch(error){

  }
  this.setState({loading1:false});
  Router.replace(`/campaigns/${this.props.address}/requests`);
  }
  onFinalize=async()=>{
```

```
   this.setState({loading2:true});
   const campaign=Campaign(this.props.address);
   const accounts=await web3.eth.getAccounts();
   try{
   await campaign.methods.finalizeRequest(this.props.id).send({from:accounts[0]});
  }
   catch(error){

  }
   this.setState({loading2:false});
Router.replace(`/campaigns/${this.props.address}/requests`);
  }
  render(){
   const{Row,Cell}=Table;
   const{id,request,approversCount}=this.props;
   const readyTOFinalize=request.approvalCount>approversCount/2;
   return (
     <Row disabled={request.complete} positive={readyTOFinalize && !request.complete}>
       <Cell>{id+1}</Cell>
       <Cell>{request.description}</Cell>
       <Cell>{web3.utils.fromWei(request.value,'ether')} </Cell>
       <Cell>{request.recipient}</Cell>
       <Cell>{request.approvalCount}/{approversCount}</Cell>
       <Cell><Button disabled={request.complete}  loading={this.state.loading1}  color="green"
basic onClick={this.onApprove}>Approve</Button></Cell>
       <Cell><Button disabled={request.complete}  loading={this.state.loading2} color="teal"
basic onClick={this.onFinalize}>Finalize</Button></Cell>
     </Row>);
  }
}
export default RequestRow;
```

4. *ContributeForm.js *

```
import React,{Component} from 'react';
import {Form,Input,Message,Button,Icon} from 'semantic-ui-react';
import Campaign from '../ethereum/Campaign';
import web3 from '../ethereum/web3';
import {Router} from '../routes';
class ContributeForm extends Component {

  state={
    value:'',
    loading:false,
    errorMessage:'',
    transaction_status_message_for_notched:'',
    message:''
  }

  onSubmit= async(event)=>{
    event.preventDefault();
    if(this.state.value<web3.utils.fromWei(this.props.minimumContribution,'ether')){
      window.alert("Please give  minimum contribution as given");
      return ;
    }
    const campaign=Campaign(this.props.address);
    this.setState({loading:true,transaction_status_message_for_notched:'Waiting on transaction
success...',errorMessage:''})
    try {
      const accounts=await web3.eth.getAccounts();
      console.log(web3.eth.getBalance(accounts[0]));
      await
campaign.methods.contribute().send({from:accounts[0],value:web3.utils.toWei(this.state.value,'
ether')});
      this.setState({message:'You suceessfully contibute to this  campaign'});
      //To Refresh the page we use Router.replaceRoute() method
      Router.replaceRoute(`/campaigns/${this.props.address}`);

    }
    catch (error) {
      this.setState({message:'Transaction failed',errorMessage:'Hey your metamask is not allow to
complete the transaction !'+error.message});
}
```

```
this.setState({loading:false,value:'',message:''});
  }
  render(){
     return(
     <Form onSubmit={this.onSubmit} error={!!this.state.errorMessage}>
       <Form.Input
         min="0"
         step="any"
         type="number"
         pattern="[0-9]"
         label=<h3>Ammount to Contibute</h3>
         style={{width:"150px"}}
         placeholder="Ammount in ether"
         value={this.state.value}
         onChange={(event)=>this.setState({value:event.target.value})}
       />
       <Button loading={this.state.loading} primary>Contribute</Button>
       <h2>{this.state.message}</h2>
       <Message error style={{width:'520px'}}  header="Oops!"
content={this.state.errorMessage}/>
       <Message icon hidden={!this.state.loading} positive>
       <Icon name='circle notched' loading />

<Message.Content>{this.state.transaction_status_message_for_notched}</Message.Content>
       </Message>
     </Form>
    );
  }

}

export default ContributeForm;
```

For pages:

pages/index.js

```
import React, {Component} from 'react';
import { Card ,Button} from 'semantic-ui-react';
import factory from '../ethereum/factory';
import Layout from '../components/layout';
import {Link} from '../routes';
//import 'semantic-ui-css/semantic.min.css';
//but  next does not support css module ,so we don't use this above module
class CampaignIndex extends Component{
  //in any traditional react app or ptoject async componentDidMount() it is 100% appropiate but we using next js
  //next js introduce one little extra requirement around data loading
  //The server side rendering attempt to render our Component on the server and take that all Html and send it to the browser
  //and we getting data in server side by this line (const campaigns=await factory.methods.getDeployedCampaigns().call())
  //next doesn't execute componentDidMount method ,so when our application is beign rendered by next on server
  //async componentDidMount() not run so we use getInitialProps()
  static async getInitialProps(){
     const campaigns=await factory.methods.getDeployedCampaigns().call();
     // return {campaigns:campaigns};
     return {campaigns};
  }
  //getInitialProps return an object and that object is going to be provided to our component as props
  //and use that in our component this.props.campaigns

  renderCampaigns(){
    //function(address) == (addres)=>  ==(or  we have single argument so) address=>
  // const items=this.props.campaigns.map(address=>{
  //   return{
  //     header:address,
  //     description:<a href="show">View Campaign</a>,
  //     fluid:true
  //   };
  // });
```

```
//            OR

  const items=this.props.campaigns.map(address=>{
    return{
      header:address,
      description:(
                //<Link route={'/campaigns/'+address}>
                <Link route={`/campaigns/${address}`}>
                   <a>View Campaign</a>
                </Link>
               ),
      fluid:true
    };
  });

  return <Card.Group items={items}/>;
  }

  //we have to write km se km chota jsx otherwise it will some error (in render() method)
  render(){
    return (
    <Layout>
    <div>
    <h2>Open Campaigns</h2>
    <Link route="/campaigns/new">
      <a>
        <Button
          floated='right'
          content="Create Campaign"
          icon="plus circle"
          primary
          labelPosition='right'
          />
        </a>
    </Link>
    {this.renderCampaigns()}

    </div>
    </Layout>
  );
```

```
  }
}

//we use<a></a> to create or show a new tab only

//page always except that file exports a react Component
export default CampaignIndex;


//component is reder both on the server and once everything looads up it's executed on the client side as well
//to test server side rendering by disabling javascript execution inside our browser
pages/campaigns/new.js

import React,{Component} from 'react';
import Layout from '../../components/layout';
import {Button,Form,Input,Message,Icon} from 'semantic-ui-react';
import factory from '../../ethereum/factory';
import web3 from '../../ethereum/web3';
import {Router} from '../../routes';
//Link object is a react components that allow us to render <a></a> into our react
// components and navigate arround the application

//Router object allow us to programatically redirect people from one page to another
CampaignNew
//inside our application
class CampaignNew extends Component{
  state={
    minimumContribution:'',
    errorMessage:'',
    loading:false,

    realMessage:''
  };
  //whenever we do form submittal in the browser ,the browser is going to
  //attempt automatically submit the form to back end server and this thing we do not want to
have
  //to stop this we use = preventDefault()
   onSubmit=async (event)=>{
     this.setState({loading:true,errorMessage:'',realMessage:''});
```

```
    event.preventDefault();
    try
    {
      const accounts=await web3.eth.getAccounts();
      await factory.methods.createCampaign(this.state.minimumContribution)
      .send({
        from:accounts[0]
      });
      this.setState({realMessage:'You suceessfully created the campaign'});
      Router.pushRoute('/');
   }
   catch(error)
   {
     this.setState({errorMessage:'Hey your metamask is not allow to complete the transaction'});

  }
  //After succeesfully exit this function we have to stop that loading
  this.setState({loading:false});
}
  render(){
    return (
      <Layout>
        <Message
        info
        header='Hint'
        content='Hey!!This contract is developed on the ethereum blockchain so ,it only support
decimal value'
        style={{width:'620px'}}
        >
        </Message >
        <Form onSubmit={this.onSubmit} error={!!this.state.errorMessage}>
        <h2>Create a Campaign</h2>
        <Form.Input
          min="1"
          type="number"
          pattern="[0-9]"
          floating='true'
          label='Minimum Contribution'
          style={{width:"150px"}}
          placeholder="Ammount in wei"
```

```
        value={this.state.minimumContribution}
        onChange={(event)=>this.setState({minimumContribution:event.target.value})}
        />
        <Button loading={this.state.loading} primary type='submit' >Create  </Button>
        <h2>{this.state.realMessage}</h2>
        <Message style={{width:"500px"}}error header="Oops !" content
={this.state.errorMessage}/>

      </Form>
    </Layout>

    );
}
}

export default CampaignNew;



pages/campaigns/show.js
import React,{Component} from 'react';
import Layout from '../../components/layout';
import Campaign from '../../ethereum/Campaign';
import {Card,Grid,Button} from 'semantic-ui-react';
import web3 from '../../ethereum/web3';
import ContributeForm from '../../components/ContributeForm';
import {Link} from '../../routes';
class CampaignShow extends Component{
 //props.query.something     ! Here smething is routes that we add in route.js file
 static async getInitialProps(props){
   //console.log("address of campaign where it is deploy"+props.query.address);
   const campaign=Campaign(props.query.address);
   const summary=await campaign.methods.getSummary().call();
   //summary is Result object

   return {
        address:props.query.address,
        minimumContribution:summary['0'],
        campaignBalance:summary['1'],
```

```
          requestsCount:summary['2'],
          approversCount:summary['3'],
          manager:summary['4']
        };
  }

  renderCards(){
    const
{campaignBalance,manager,minimumContribution,approversCount,requestsCount}=this.props
;
    const items=[
      {
        header:manager,
        meta:'Address of manager ',
        description:'This manager created this campaign and able to create and finalize request to
withdraw money',
        style:{overflowWrap:'break-word'}
      },
      {
        header:minimumContribution,
        meta:'Minimum Contribution (in Wei) for this Campaign ',
        description:'Pledge to give minimum '+minimumContribution+' wei to make this
campaign successfull and become approver',
      },
      {
        header:requestsCount,
        meta:'Number of request',
        description:'A request tries to withdraw money from the contract .  Request must be
approved by approvers',
      },
      {
        header:web3.utils.fromWei(campaignBalance,'ether'),
        meta:'Campaign balance(in ether)',
        description:'this campaign has funded  '+web3.utils.fromWei(campaignBalance,'ether') +'
ether till now and manager allow to spent all money left ',
      },
      {
        header:approversCount,
        meta:'Number of approvers ',
        description:'Number of people who have already donated to this campaign'
      },
```

```
    ];
    return <Card.Group items={items}/>;
  }

  render(){
   return (
    <Layout>
     <h2>Campaign Details</h2>
     <Grid>
       <Grid.Row>
         <Grid.Column width={10}>
           {this.renderCards()}
         </Grid.Column>
         <Grid.Column width={6}>
            <ContributeForm address={this.props.address}
minimumContribution={this.props.minimumContribution}/>
         </Grid.Column>
        </Grid.Row>

       <Grid.Row>
         <Grid.Column>
           <Link route={`/campaigns/${this.props.address}/requests`}>
              <a>
                <Button primary floated="left" >View Requests</Button>
              </a>
           </Link>
          </Grid.Column>
       </Grid.Row>
      </Grid>
    </Layout>
   );
  }
}
```

```
export default CampaignShow;
```

pages/campaigns/requests/index.js

```
import React, { Component } from "react";
import Layout from "../../../components/layout";
import { Button, Table, Divider, Message } from "semantic-ui-react";
import { Link } from "../../../routes";
import Campaign from "../../../ethereum/Campaign";
import RequestRow from "../../../components/RequestRow";
//In solidity we do not return array of struct
class RequestIndex extends Component {
  static async getInitialProps(props) {
    const { address } = props.query;
    const campaign = Campaign(address);
    const requestCount = await campaign.methods.getRequestCount().call();
    const approversCount = await campaign.methods.approversCount().call();
    //we will iterate or loop up to requestCount in All promise
    //not one by one

    //getRequestsCount returns a number inside a string, but the Array  constructor
    // expects to be passed a number, not a string. To fix this, we can use the parseInt
    const requests = await Promise.all(
      Array(parseInt(requestCount))
        .fill()
        .map((element, index) => {
          return campaign.methods.requests(index).call();
        })
    );
    return { address, requests, requestCount, approversCount };
  }
  renderRow() {
    return this.props.requests.map((request, index) => {
      return (
        <RequestRow
          id={index}
```

```
        key={index}
        request={request}
        address={this.props.address}
        approversCount={this.props.approversCount}
      />
    );
  });
}
render() {
  const { Header, Row, HeaderCell, Body } = Table;
  return (
    <Layout>
      <Message
        info
        header="Information alert"
        content="You only finalize the request when you get more than 50% vote "
        style={{ width: "620px" }}
      />
      <h2>Pending Requests</h2>
      <Link route={`/campaigns/${this.props.address}/requests/new`}>
        <a>
          <Button floated="right" primary style={{ marginBottom: 30 }}>
            {" "}
            Add Request
          </Button>
        </a>
      </Link>
      <Table>
        <Header>
          <Row>
            <HeaderCell>ID</HeaderCell>
            <HeaderCell>Description</HeaderCell>
            <HeaderCell>Ammount(in ether)</HeaderCell>
            <HeaderCell>Recipient</HeaderCell>
            <HeaderCell>Approval</HeaderCell>
            <HeaderCell>Approve</HeaderCell>
            <HeaderCell>Finalize</HeaderCell>
          </Row>
        </Header>
        <Body>{this.renderRow()}</Body>
```

```
        </Table>
        <h3>Found {this.props.requestCount} request .</h3>
      </Layout>
    );
  }
}

export default RequestIndex;
```

pages/campaigns/requests/new.js

```
import React,{Component} from 'react';
import Layout from '../../../components/layout';
import {Form,Button,Message,Input,TextArea,Icon} from 'semantic-ui-react';
import Campaign from '../../../ethereum/Campaign';
import web3 from '../../../ethereum/web3';
import {Link,Router} from '../../../routes';
class RequestNew extends Component{
  state={
        description:'',
        ammount_transfer:'',
        recipient:'',
        loading:false,
        errorMessage:'',
        message:''
}
 onSubmit=async(event)=>{
   event.preventDefault();
  this.setState({loading:true,errorMessage:'',message:''});

  const campaign=Campaign(this.props.address);
  const {description,ammount_transfer,recipient}=this.state;
```

```
  try {
    const accounts=await web3.eth.getAccounts();
    await
campaign.methods.createRequest(description,web3.utils.toWei(ammount_transfer,'ether'),recip
ient).send({from:accounts[0]});
    this.setState({message:'You sucessfully create the request'});
    Router.pushRoute(`/campaigns/${this.props.address}/requests/`);
}
  catch (error) {

     this.setState({errorMessage:'Hey your metamask is not allow to complete the transaction !
'+error.message})
  }
this.setState({loading:false});
}
  static async getInitialProps(props){
    const {address}=props.query;
    return {address};
  }
  render(){
    return (
      <Layout>
          <Form onSubmit={this.onSubmit} error={!!this.state.errorMessage}>
            <Link route={`/campaigns/${this.props.address}/requests`}>
            <a><h3>Back</h3></a>
            </Link>
            <h1>Create a request</h1>
            <Form.TextArea

            label='Description'
            placeholder="what this request for ?"
            value={this.state.description}
            onChange={event=>this.setState({description:event.target.value})}
            />

            <Form.Input

            label="Recipient Account"
            placeholder="Address of recipient who got money"
            value={this.state.recipient}
```

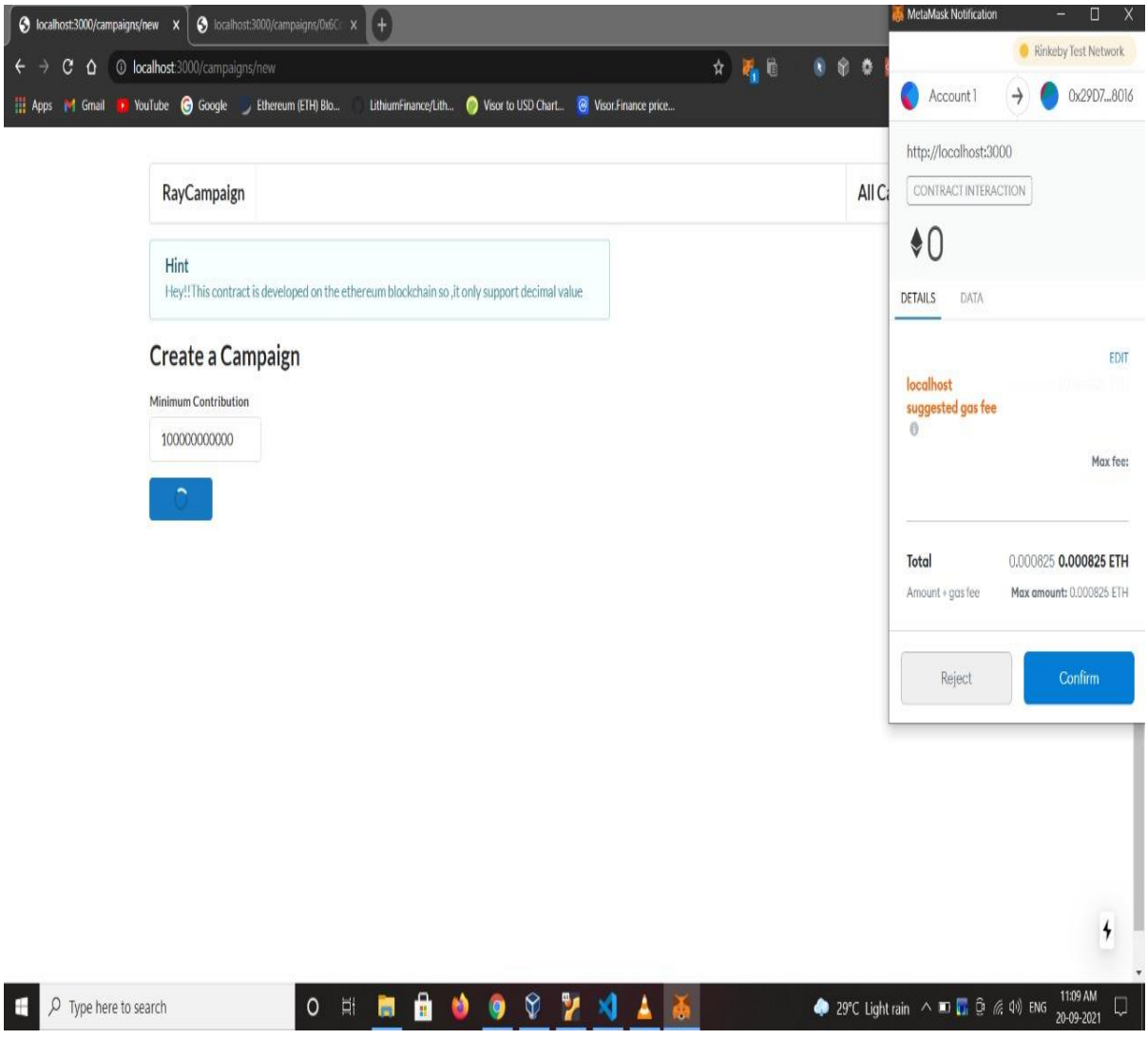
```
            onChange={event=>this.setState({recipient:event.target.value})}
            />
            <Form.Input

              pattern="[0-9]"
              type="number"
              step="any"
              label="Ammount transfer (in ether)"
              style={{width:"250px"}}
              placeholder="Ammount in ether "
              value={this.state.ammount_transfer}
              onChange={event=>this.setState({ammount_transfer:event.target.value})}
             />
            <Button loading={this.state.loading} floated="left" primary>Add request</Button>
            <br/>
            <h2>{this.state.message}</h2>
            <Message error header="Oops! "content={this.state.errorMessage} />

            </Form>
      </Layout>
    );
  }
}
export default RequestNew;

//here test may be related to ethereum side and  other might be related to web appplication side!!
const assert = require('assert');
const ganache = require('ganache-cli');
const Web3=require('web3');
const web3=new Web3(ganache.provider());

const compiledFactory=require('../ethereum/build/CampaignFactory.json');
const compiledCampaign=require('../ethereum/build/Campaign.json');

let accounts;
//factory=>  refrence to the deployed instance of the factory that we gonna make
let factory;
let campaign;
let campaignAddress;
```

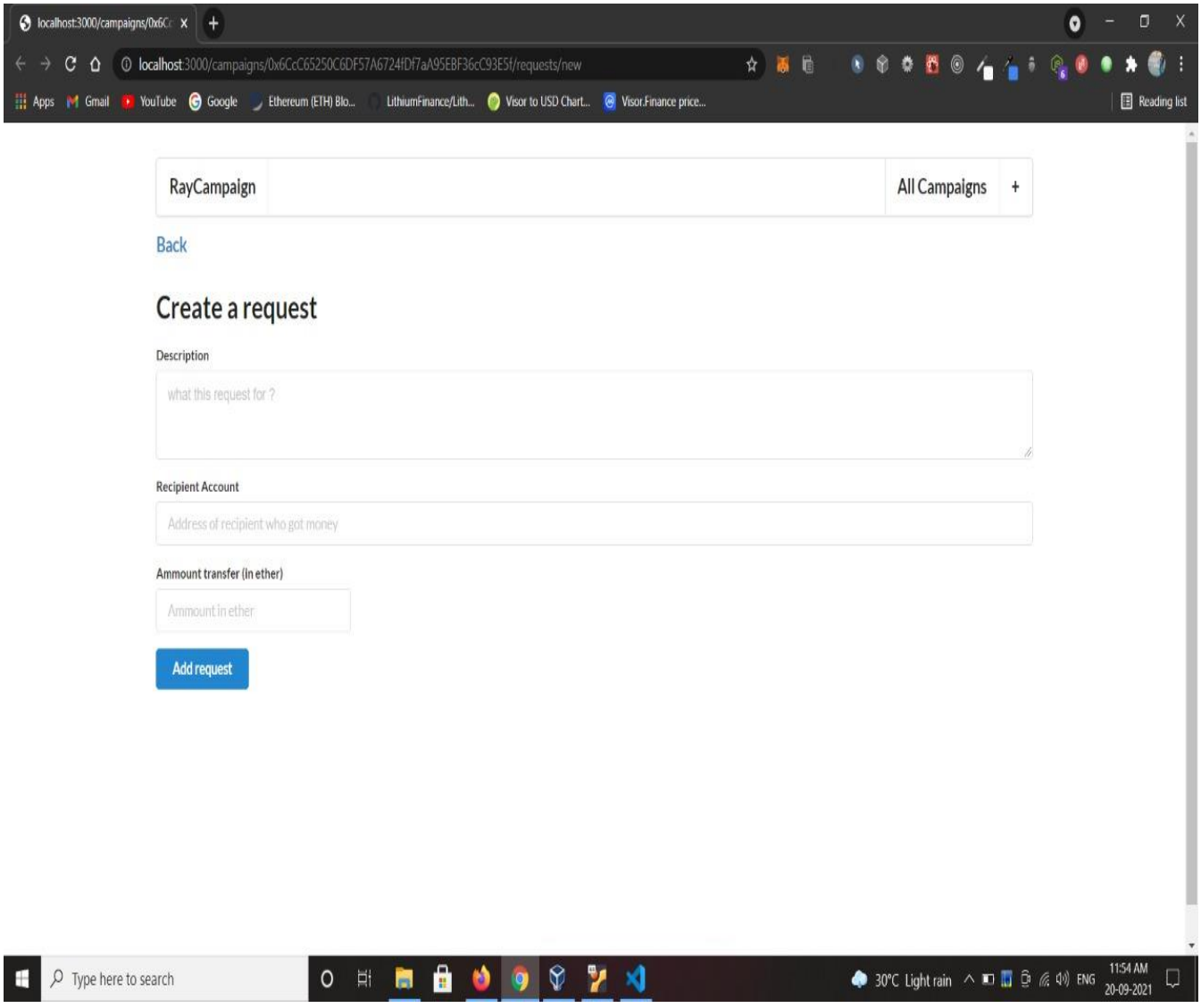
```
beforeEach(async()=>{
//mostly we need instance of a Campaign
//so rather we use factory inside of every it() block to create seperate instance of a Campaign we
use  Campaign
  accounts=await web3.eth.getAccounts();
  factory=await new web3.eth.Contract(JSON.parse(compiledFactory.interface))
  .deploy({data:compiledFactory.bytecode})
  .send({from:accounts[0],gas:1000000});
   await factory.methods.createCampaign(10).send({
    from:accounts[0],
    gas:1000000
  });
  //it is es2016 syntax
// const addresses=  await factory.methods.getDeployedCampaigns().call();
// campaignAddress=addresses[0];
//use es5 syntax
  [campaignAddress]=await factory.methods.getDeployedCampaigns().call();
//create actual  instance of campaign
//we use Web3 for instruct that will acess to address at  campaignAddress

campaign=await                                                                    new
web3.eth.Contract(JSON.parse(compiledCampaign.interface),campaignAddress);
});



describe('Campaigns',()=>{

  it('deploys a factory and a campaign',()=>{
    assert.ok(factory.options.address);
    assert.ok(campaign.options.address);
  });

  //test campaign has manager of address mark as  account[0],because we
  //use account[0] to create instance of a campaignAddress
  //means manager os this campaign have account[0]
  //caller=> person who call createCampaign() method
  it('marks caller as the campaign manager',async()=>{
    const campaign_manager=await campaign.methods.manager().call();
    assert.equal(campaign_manager,accounts[0]);
  });
```

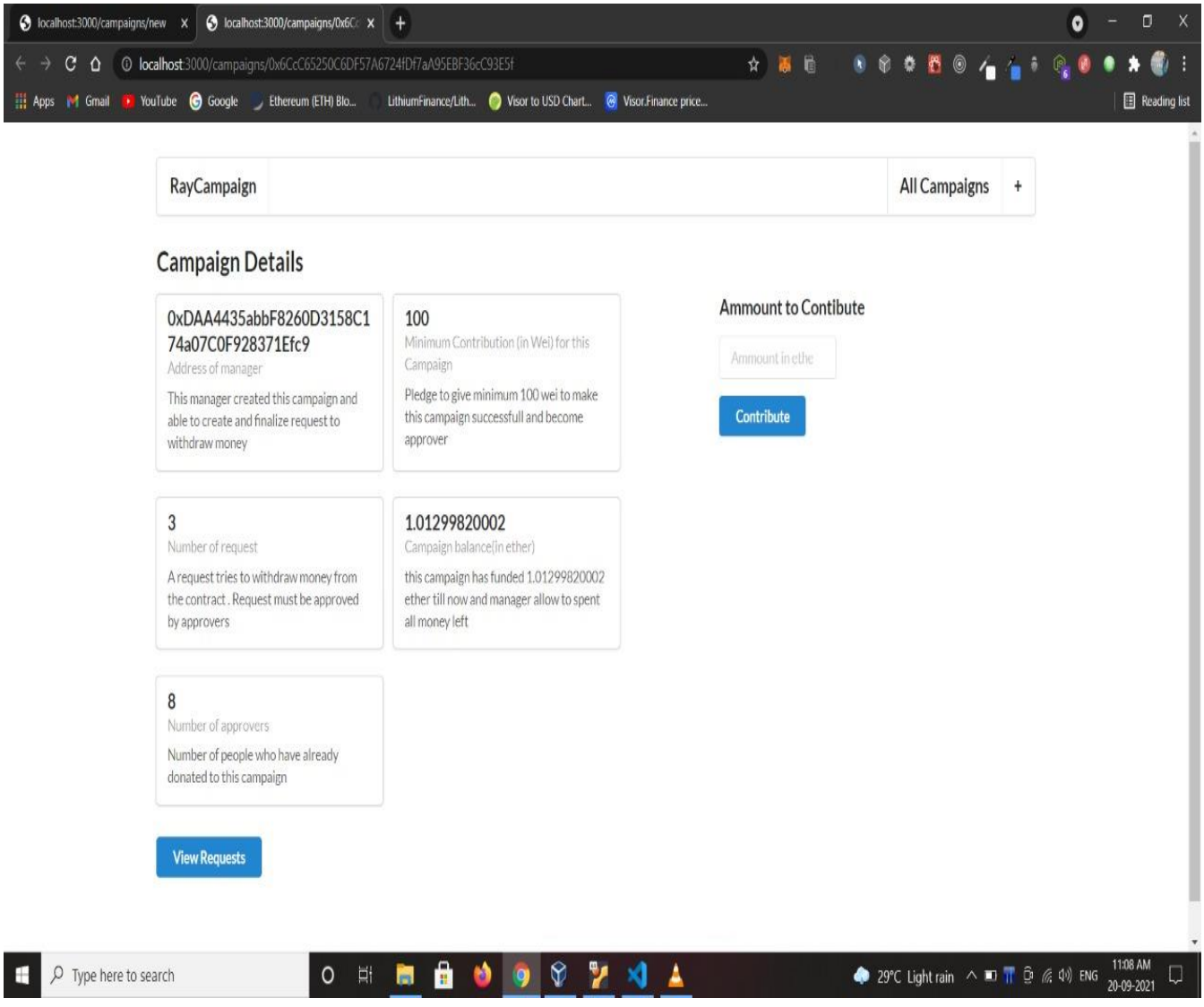
```
  it('Allows people to contribute money and mark them as approver',async()=>{

    await campaign.methods.contribute().send({value:200,from:accounts[1]});
    const is_Contributor=await campaign.methods.approvers(accounts[1]).call();
    //is_Contributor=false;
    //assert(is_Contributor);
    assert(is_Contributor);
});

  it('Requires a minimum contribution',async()=>{

try {
  await campaign.methods.contribute().send({value:100,from:accounts[1]});
//we use assert(false) because yeh wali line(campaign.methods.contribute()
//.send({value:10,from:accounts[1]});) chalne ke baad hum seedha catch block ke
//andar  aa jayenege and agar test fail hua it means try block ke andar hi the
  assert(false);
}
catch (error) {
  assert(error);
}

  });
it('only manager has the ability to make a payment request',async()=>{
  await
campaign.methods.createRequest("batteries",100,accounts[1]).send({from:accounts[0],gas:1000
000});
  const request=await campaign.methods.requests(0).call();
  assert.equal('batteries',request.description);

});

//now contribute,createRequest,approveRequest,finalizeRequest
it('process requests',async()=>{

  await campaign.methods.contribute().send({
    from:accounts[0],
    value:web3.utils.toWei('10','ether')
```

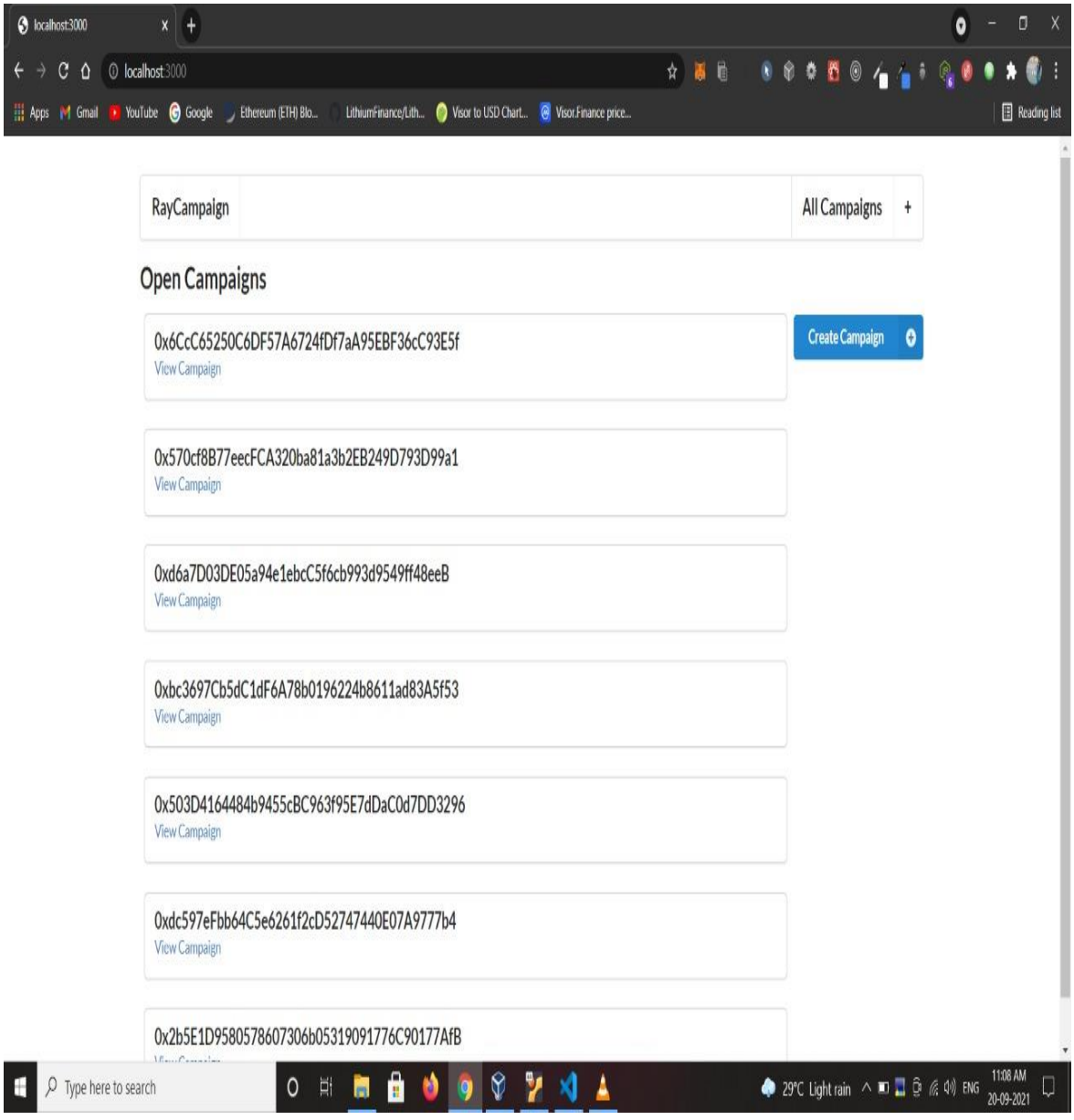
```
  });

  await campaign.methods
  .createRequest('A',5,accounts[1])
  .send({from:accounts[0],
       gas:1000000});

   await campaign.methods.approveRequest(0).send({from:accounts[0],gas:1000000});

   await campaign.methods.finalizeRequest(0).send({from:accounts[0],gas:1000000});

   //balance is a string that represent the ammounts of money that account has in wei
   let balance=await web3.eth.getBalance(accounts[1]);
   //now balance is in ether
   balance=web3.utils.fromWei(balance,'ether');
   //balance have sting value just in float type
   balance=parseFloat(balance);
   //in real all ganache accounts have 100 ether and we transfer 5 ether to finalize
   //so we use 99.+5>104
   assert(balance>104);
});
});
```

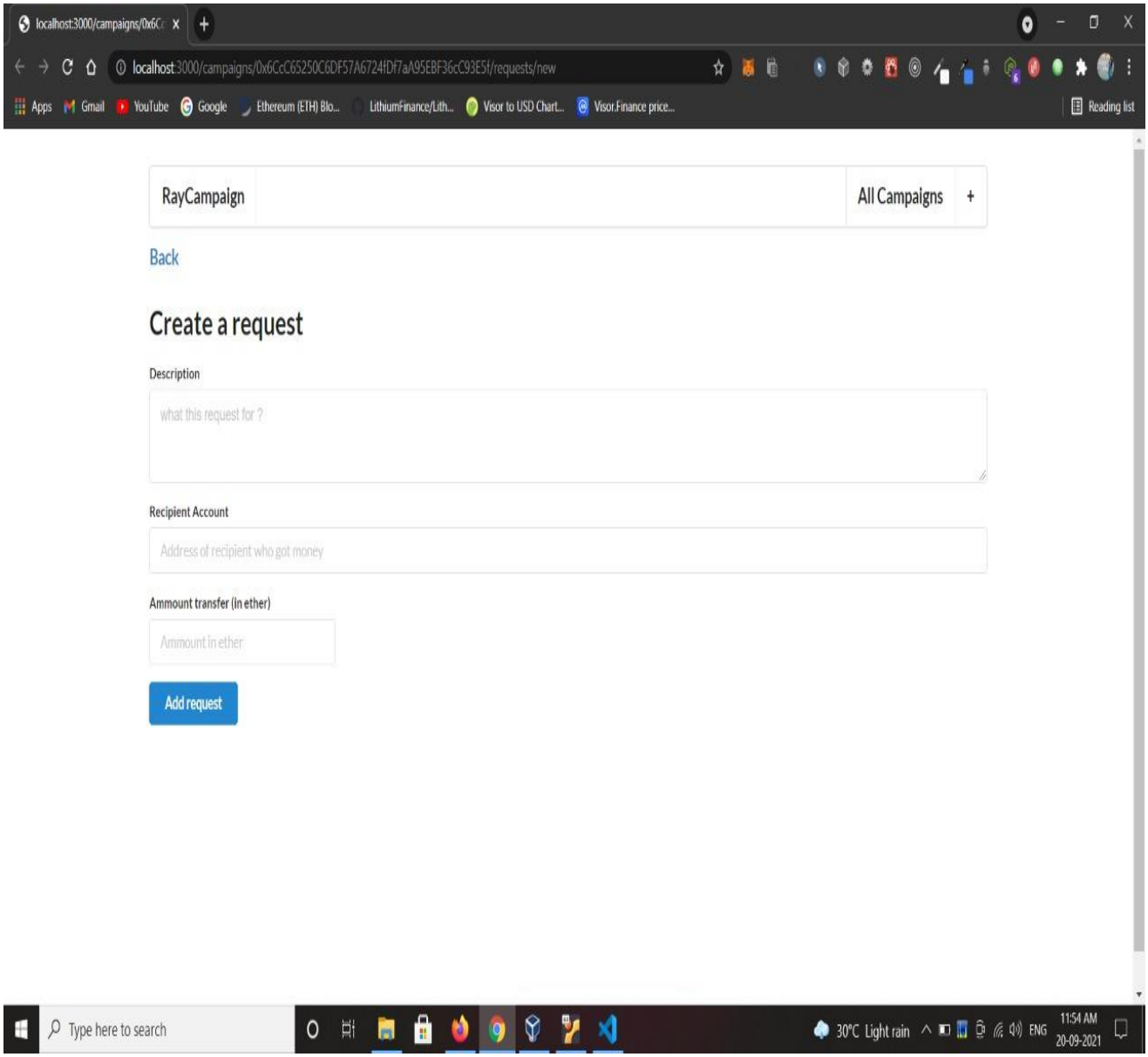
Screenshots

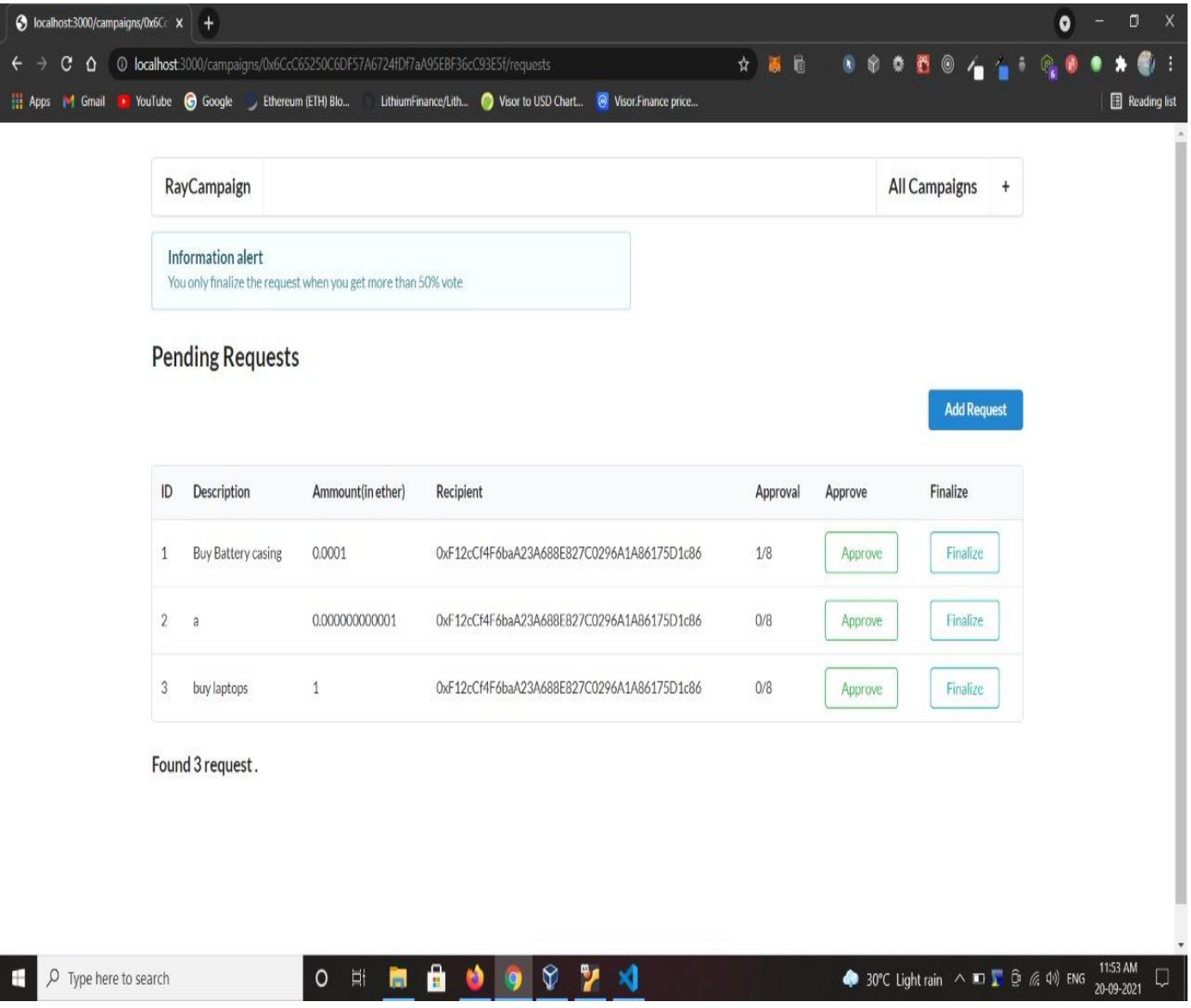


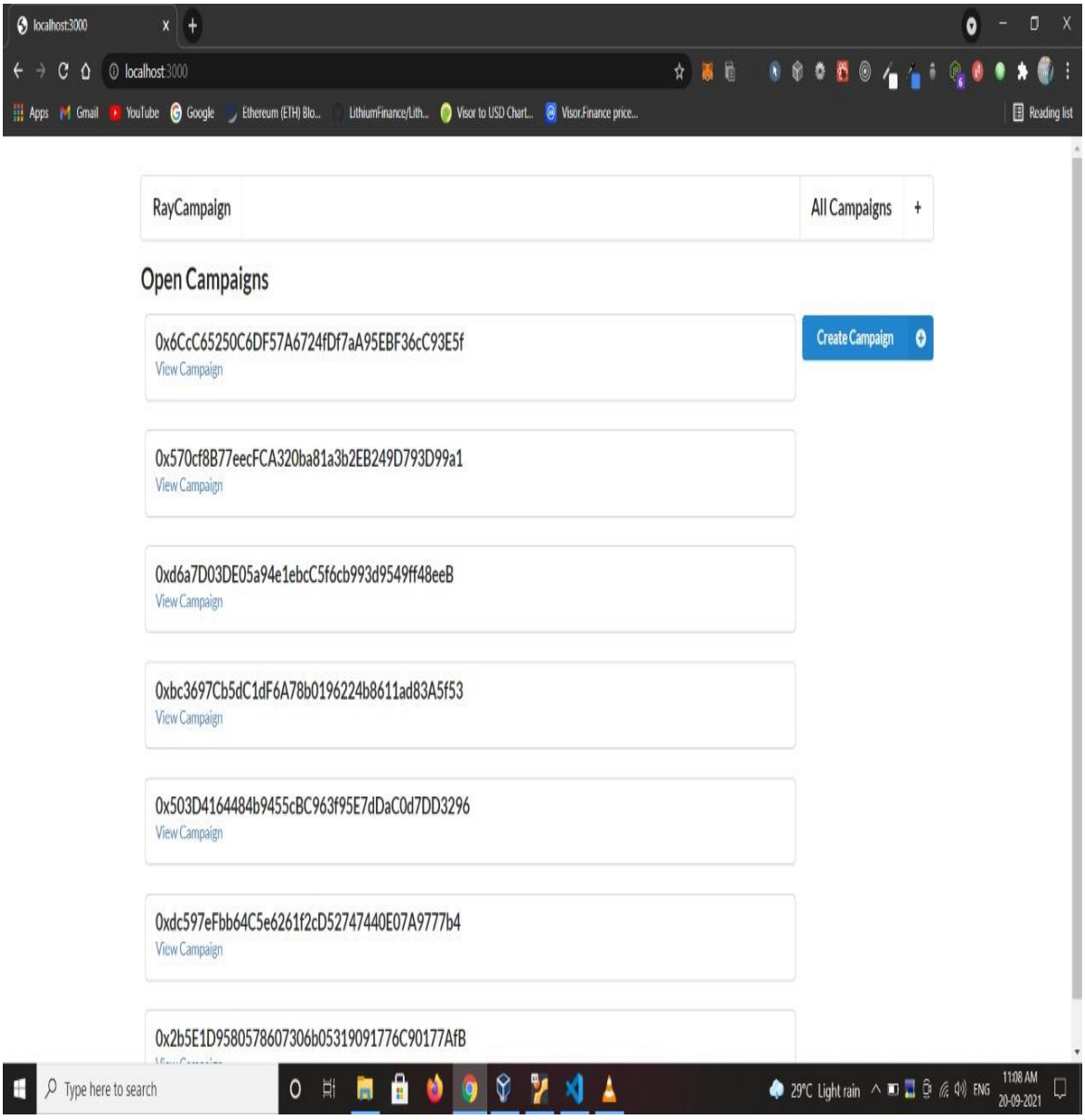

localhost:3000/campaigns/0x6Cc
localhost:3000/campaigns/0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f/requests/new
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Back
Create a request
Description
what this request for ?
Recipient Account
Address of recipient who got money
Ammount transfer (in ether)
Ammount in ether
Add request
Type here to search
30°C Light rain
ENG
11:54 AM
20-09-2021

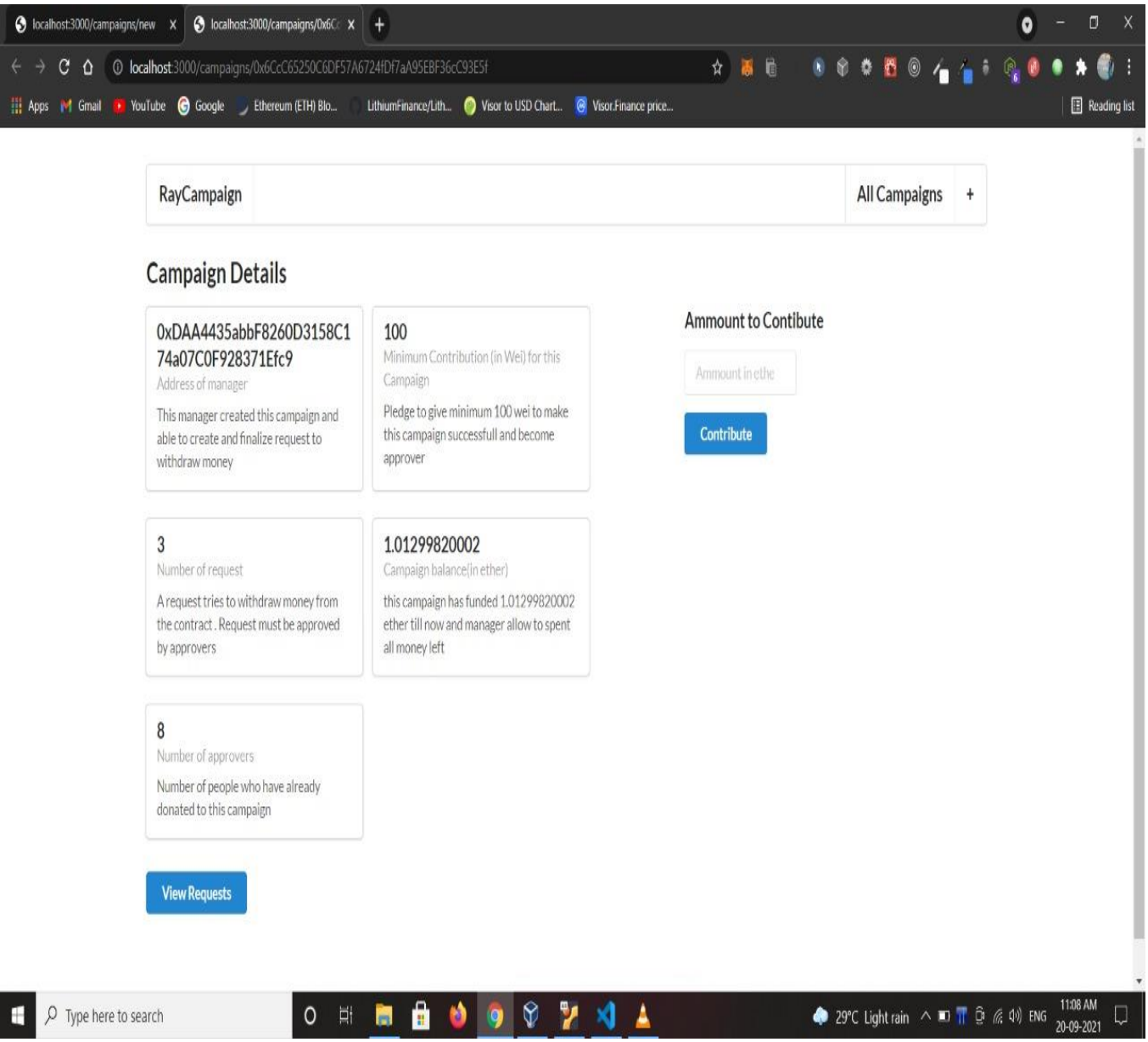

localhost:3000/campaigns/new
localhost:3000/campaigns/0x6C
localhost:3000/campaigns/0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Campaign Details
0xDAA4435abbF8260D3158C174a07C0F928371Efc9
Address of manager
This manager created this campaign and able to create and finalize request to withdraw money
100
Minimum Contribution (in Wei) for this Campaign
Pledge to give minimum 100 wei to make this campaign successfull and become approver
Ammount to Contibute
Ammount in ethe
Contribute
3
Number of request
A request tries to withdraw money from the contract . Request must be approved by approvers
1.01299820002
Campaign balance(in ether)
this campaign has funded 1.01299820002 ether till now and manager allow to spent all money left
8
Number of approvers
Number of people who have already donated to this campaign
View Requests
Type here to search
29°C Light rain
ENG
11:08 AM
20-09-2021

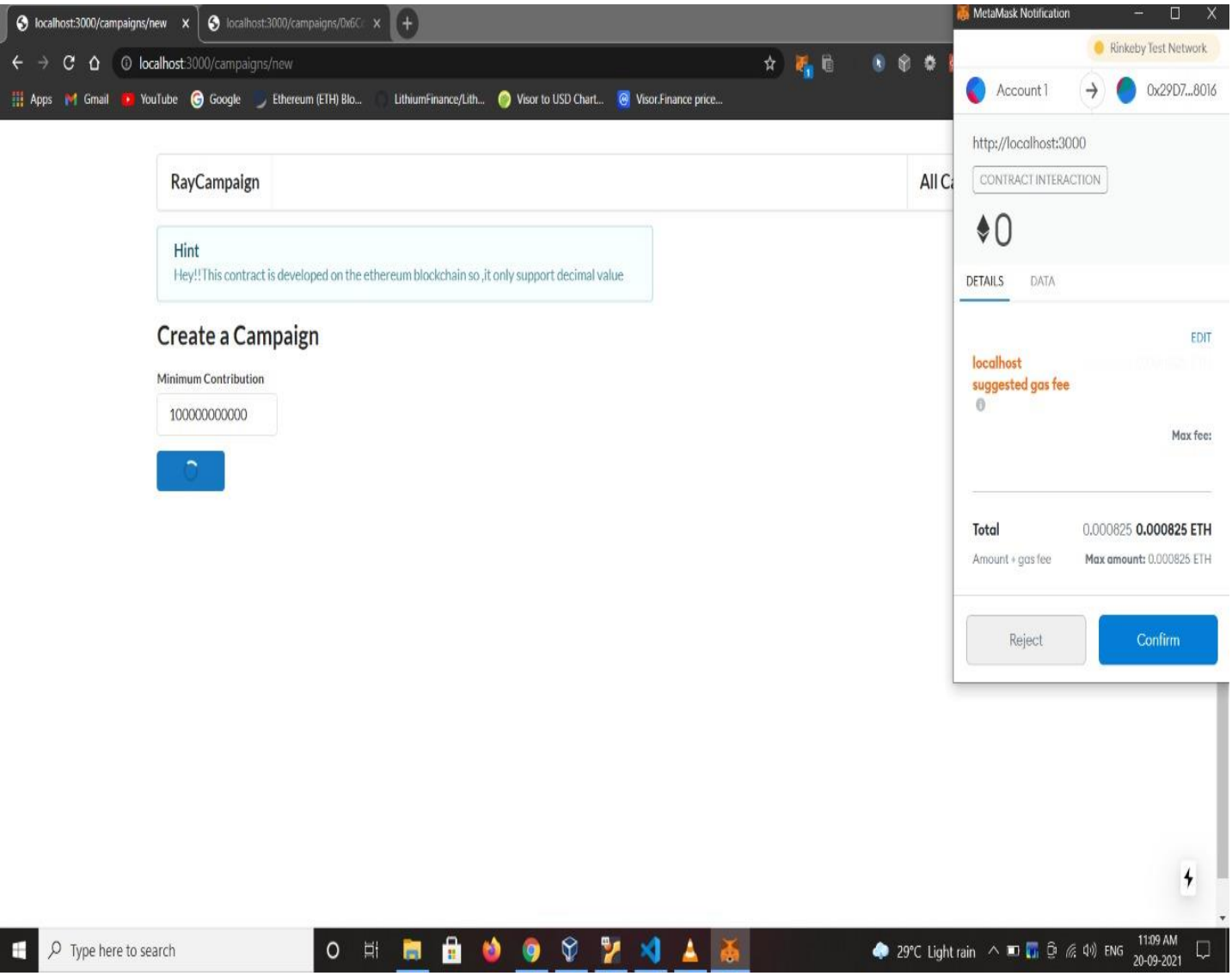

localhost:3000
localhost:3000
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Open Campaigns
0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f
View Campaign
Create Campaign
0x570cf8B77eecFCA320ba81a3b2EB249D793D99a1
View Campaign
0xd6a7D03DE05a94e1ebcC5f6cb993d9549ff48eeB
View Campaign
0xbc3697Cb5dC1dF6A78b0196224b8611ad83A5f53
View Campaign
0x503D4164484b9455cBC963f95E7dDaC0d7DD3296
View Campaign
0xdc597eFbb64C5e6261f2cD52747440E07A9777b4
View Campaign
0x2b5E1D9580578607306b05319091776C90177AfB
Type here to search
29°C Light rain
ENG
11:08 AM
20-09-2021


localhost:3000/campaigns/0x6Cc
localhost:3000/campaigns/0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f/requests/new
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Back
Create a request
Description
what this request for ?
Recipient Account
Address of recipient who got money
Ammount transfer (In ether)
Ammount in ether
Add request
Type here to search
30°C Light rain
ENG
11:54 AM
20-09-2021


localhost:3000/campaigns/0x6C
localhost:3000/campaigns/0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f/requests
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Information alert
You only finalize the request when you get more than 50% vote
Pending Requests
Add Request
ID
Description
Ammount(in ether)
Recipient
Approval
Approve
Finalize
1
Buy Battery casing
0.0001
0xF12cCf4F6baA23A688E827C0296A1A86175D1c86
1/8
Approve
Finalize
2
a
0.000000000001
0xF12cCf4F6baA23A688E827C0296A1A86175D1c86
0/8
Approve
Finalize
3
buy laptops
1
0xF12cCf4F6baA23A688E827C0296A1A86175D1c86
0/8
Approve
Finalize
Found 3 request .
Type here to search
30°C Light rain
ENG
11:53 AM
20-09-2021


localhost:3000
localhost:3000
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Open Campaigns
0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f
View Campaign
Create Campaign
0x570cf8B77eecFCA320ba81a3b2EB249D793D99a1
View Campaign
0xd6a7D03DE05a94e1ebcC5f6cb993d9549ff48eeB
View Campaign
0xbc3697Cb5dC1dF6A78b0196224b8611ad83A5f53
View Campaign
0x503D4164484b9455cBC963f95E7dDaC0d7DD3296
View Campaign
0xdc597eFbb64C5e6261f2cD52747440E07A9777b4
View Campaign
0x2b5E1D9580578607306b05319091776C90177AfB
Type here to search
29°C Light rain
ENG
11:08 AM
20-09-2021


localhost:3000/campaigns/new
localhost:3000/campaigns/0x6C
localhost:3000/campaigns/0x6CcC65250C6DF57A6724fDf7aA95EBF36cC93E5f
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
Reading list
RayCampaign
All Campaigns
+
Campaign Details
0xDAA4435abbF8260D3158C174a07C0F928371Efc9
Address of manager
This manager created this campaign and able to create and finalize request to withdraw money
100
Minimum Contribution (in Wei) for this Campaign
Pledge to give minimum 100 wei to make this campaign successfull and become approver
Ammount to Contibute
Ammount in ethe
Contribute
3
Number of request
A request tries to withdraw money from the contract. Request must be approved by approvers
1.01299820002
Campaign balance(in ether)
this campaign has funded 1.01299820002 ether till now and manager allow to spent all money left
8
Number of approvers
Number of people who have already donated to this campaign
View Requests
Type here to search
29°C Light rain
ENG
11:08 AM
20-09-2021


localhost:3000/campaigns/new
localhost:3000/campaigns/new
Apps
Gmail
YouTube
Google
Ethereum (ETH) Blo...
LithiumFinance/Lith...
Visor to USD Chart...
Visor.Finance price...
RayCampaign
Hint
Hey!! This contract is developed on the ethereum blockchain so ,it only support decimal value
Create a Campaign
Minimum Contribution
100000000000
MetaMask Notification
Rinkeby Test Network
Account 1
0x29D7...8016
http://localhost:3000
CONTRACT INTERACTION
0
DETAILS
DATA
EDIT
localhost
suggested gas fee
Max fee:
Total
0.000825 0.000825 ETH
Amount + gas fee
Max amount: 0.000825 ETH
Reject
Confirm
Type here to search
29°C Light rain
ENG
11:09 AM
20-09-2021

Conclusion

- We can conclude that the implementation of blockchain can improve many drawbacks that area unit gift within the ancient crowdfunding platforms. This includes, increased security, increased potency, fraud protection

- In gist, we are able to say we've got achieved the subsequent things:

- Decentralization

- Fraud interference victimisation e-Voting