VIT®

Vellore Institute of Technology

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

Internet and Web Programming

Oxygen Plant Funding Website Using Cryptocurrency

Final Report

TEAMMEMBERS:

Dhanush Kumaar J -19BCE0298

Girish Balaji – 19BCI0247

Dharun Karthick-19BCE2316

Logesh S-19BCE0839

Table of Contents

ABSTRACT

Our application is a crowdfunding website for setting up oxygen plants. We intend to use blockchain technology to create crypto tokens that investors can buy at any point from anywhere globally, and users would trade their Ethereum to invest in the seeker's project (seeker is a person/organization seeking investment). Anyone who has proper documents for building a plant can register to our site as an investment seeker, and the documents can be cross checked by the investor and then they can invest in their projectand get back their funds after the seeker gets the profit and returns it through the website.

1.INTRODUCTION

1.1 SYSTEM OVERVIEW

We have built a web application where users can donate Ethereum toinvestment seekers through and the smart contract will automatically send the ether to the investment seekers after the target is reached.

For front end we used react to render it andthe backend server is scripted in nodejs to handle the API and we use mongoDB to store all the user and project data

Smart contracts are stored in block chain and they are written using solidity language based on which transactions occur. A local test blockchain using Ganache is started, which provides 10 Ethereum wallets with 100 ethers each and the smart contract is deployed to that blockchain using truffle. Investors are allowed to buy ethers, withdraw the invested amount (if project is not sanctioned) and ether is returned to them at the end of the covid waveWe give them native

tokens instead of the Ethereum which they can withdraw as Ethereum after the project's seeker returns the money here 1 token = 0.001 ether

1.2 OBJECTIVE

Medical oxygen is the single most important intervention for moderate and severe cases of COVID-19. Hence our objective is to create a user-friendly crowdfunding website for oxygen plants so, we use a worldwide famous digital currency Ethereum so that anyone in the world who has ether can fund the oxygen plants (where 1 Ether is equivalent to Rs. **3,67,850**) and there are about 114.3 million Ether in circulation.

We know that the demand for oxygen has been increasing with the increase in the number of cases around the world and especially in India. Many people have died without the supply of oxygen at the right time. our project solves the major problem of COVID-19 ie, oxygen shortage for patients. Thus, our project has been implemented with a good motive to aid the COVID-19 affected patients through a modern way of Ethereum crowdfunding.

The crowdfunding works as follows: the seeker creates a project for funding before which they have to create and register as a seeker by filling in the necessary documents and information andafter this, investors can also register and create accounts in a similar way and after this, they can join a project started by any seeker and donate the money to meet the goal of the project. The investors will get back the money after the seeker gets their profit.

1.3 APPLICATIONS

The major application of this project is to aid the COVID-19 affected patients by helping them out with the increasing demand of oxygen supply. We know that the demand for oxygen has been increasing with the increase in the number of cases around the world and especially in India. Many people have died without the supply of oxygen at the right time. Thus it is really important to solve this problem and Ethereum crowdfunding has been used in our case to solve this.

This can also be extended to support the COVID-19 affected patients by helping to know the closest oxygen plants so that they can get the oxygen at the right time. In a nutshell our project's indirect application is to helps save livesduring these difficult times.

1.4 LIMITATIONS

The limitations of this project are:

1. The investor can only donate through Ethereum and not by any other means.
2. The return of the Ethereum to the investor depends on the trustworthiness of the seeker even though verifications are done.
3. If the seeker doesn't reach the funding target, any finance that has been pledged will usually be returned to your investors and you will receive nothing

2.SYSTEM ANALYSIS

2.1 Existing System

What makes direct crowdfunding less advantageous for unknown fundraisers (e.g. entrepreneurs) is the fact, that they would directly have to be able to address a large number of people via their own website.

The costs should be in proportion to the invested capital so that the transaction costs are not even higher. The strong control by the system and the transfer to the associated crowdfunding platforms is considered to be particularly negative.

Another problem is that some crowdsourcers and crowdworkers circumvented the platform after the initial contract and reached an extraordinary agreement.

On the crowdworking platforms, there were also some trust conflicts between the platform operators and crowdworkers, who tried to trick the system with automated work.

2.2 Proposed System

According The proposed system is a blockchain based crowdfunding platform. Blockchain technology is a decentralized ledger, more efficient, safe and tamper-proof system of nodes in connection. Introduction of blockchain in crowdfunding will make it more reliable, transparent, trusted, decentralized, cost-efficient and convenient.

2.2.1 Benefits of proposed System

- According to our research, we found that the major problem regarding the oxygen demand was poor distribution and logistics in the transportation of oxygen.
- Since transactions are made with cryptocurrency(ether) anyone

across the globe can contribute to the projects.

- Transactions are carried out throught blockchain so it is highly secure and transparent.
-

3.REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

1. Minimum:

- Processor: 1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set
- Memory: 2 gb RAM
- Display: Super VGA resolution 1024 x 768

2. Recommended:

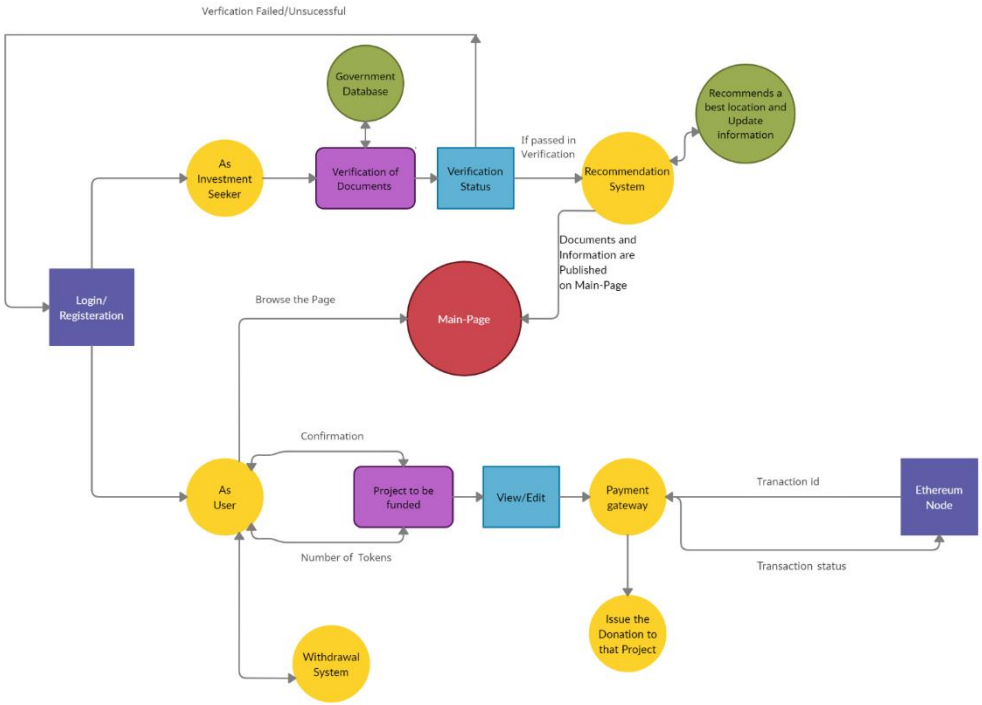
- Processor: 3.3 GHz or faster 64-bit dual core processor with SSE2 instruction set
- Memory: 4 gb RAM or more
- Display: Super VGA with resolution1024 x 768

3.2 SOFTWARE REQUIREMENTS

1. VScode (code editor)
2. Ganache (Test virtual blockchain for ethereum)
3. Node.js
4. Express.js
5. Reactjs
6. MongoDB
7. MongoDB Compass
8. Truffle.js
9. Solidity (Ethereum Solidity Language for VScode)

4.SYSTEM DESIGNSPECIFICATION

4.1SYSTEM DESIGNSPECIFICATION



4.2. DETAILED DESIGN

Website:
We used react to render our frontend, server is scripted in nodejs to handle the API and we use mongoDB to store all the user and project data
Blockchain:
A smart contract is written in Solidity language based on which transactions occur. A local test blockchain using Ganache is started, which provides 10 Ethereum wallets with 100 ethers each and the smart contract is deployed to that blockchain using truffle. Investors are allowed to buy ethers, withdraw the invested amount(if project is not sanctioned) and ether is returned to them along with some interest amount at the end of the covid wave.

4.3. DATABASE DESIGN

Tables:

1. Investors
    - Id
    - projectIds
    - userid
2. Projects
    - Id
    - Amount recieved
    - Investor count
    - Image
    - Seeker
    - Deposit
    - Address
    - Name
    - Description
    - Total required tokens
    - Ethereum address
3. Seekers
    - Project
    - User
    - Address

- IsVerified
- Stage
- Eth
- Verification doc 1
- Verification doc 2
- Verification doc 3
- Verification doc 4
- Verification doc 5

4. Users

- Id
- Name
- Email
- Password
- Session
- User type
- Ethereum address

5.SYSTEM IMPLEMENTATION

5.1 MODULE DESCRIPTION

1. Front-end: We use HTML, CSS and React-JS for the front end to implement the necessary functionalities to interact with the user like registration of seeker and investor, checking of documents, display of projects and interface to buy the tokens for donation.

2. Back-end: We use Node-JS and Express-JS to implement the backend to implement functionalities like storing the data of user details (both seeker and investor), details about the projects, the coins remaining and required for each project in the database

3. Blockchain module: We use solidity to implement the blockchain module of our project to serve functionalities such as facilitating the transactions between the users and the blockchain like ganache so that the transactions can be reflected in the blockchain and also the database of our project.

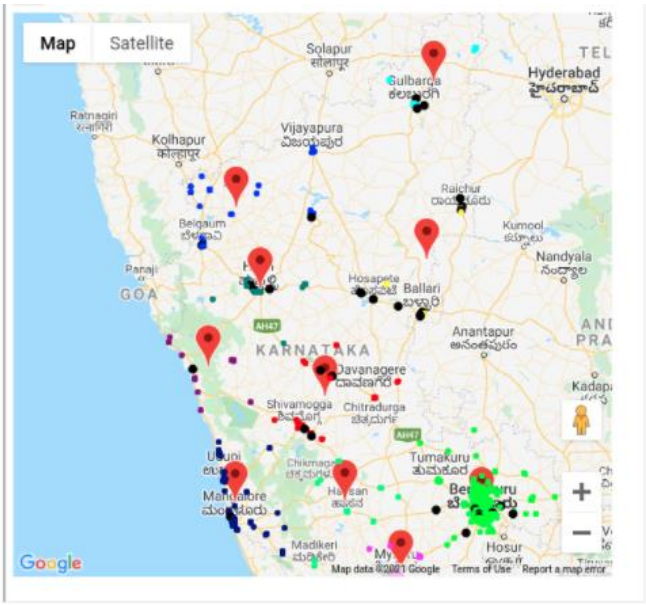
4. Investment module: This is basically the system on which the project functions, here the seeker requests and creates the project and the investor buys tokens to fulfil the needs of the seeker. This is the basic system on which our blockchain crowdfunding project works.

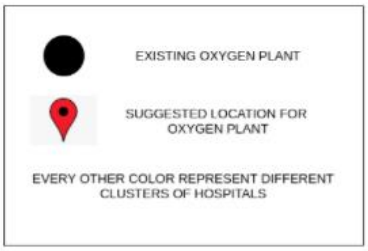
6.CONCLUSION AND FUTURE ENHANCEMENTS

We can see that this website is an alternative to investment with traditional currencies like dollars and rupees which are bound by barriers like high fees, rules and regulations which change according to the country, burden of marketing and advertising where you have to pay to the website to get featured in the top most visible spot and also copyright risks where the seekers IPs are left unprotected and with the current trend where cryptocurrency is becoming increasingly used and is also increasing in its value and quantity like Ethereum where 104 million coins are in circulation and as this token was created by keeping smart contracts in mind it is a good field to start crowdfunding projects where smart contracts are mainly used as they provide accountability.

Future Enhancements:
There is a lot of scope in this project as this project was built with the intention of being a basic one and one where we can add new features to it in several areas like including information of hotspots of covid-19 and etc

Map module:
This could revolutionize this app as we can add a map containing locations of existing oxygen plant and hospitals nearby which we can use as the input to a machine learning algorithm which recommends the suggested location of oxygen plant where potential project seekers can assess and ask a project





7.APPENDICES

7.1 APPENDIX 1 - SAMPLE SOURCE CODE

Github link: https://github.com/dhanushkumaar/crypto1.git

transaction_seeker.js

```
import React, { useState,useEffect} from 'react'
import Web3 from 'web3'
import $ from 'jquery'
import {useSelector} from 'react-redux'
import TruffleContract from '@truffle/contract'


import Snackbar from '@material-ui/core/Snackbar';
import MuiAlert from '@material-ui/lab/Alert';
import { makeStyles } from '@material-ui/core/styles';


// import { Button, Header, Image, Modal } from 'semantic-ui-react'
import Button from '@material-ui/core/Button';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogContentText from '@material-ui/core/DialogContentText';
import DialogTitle from '@material-ui/core/DialogTitle';


function Alert(props) {
  return <MuiAlert elevation={6} variant="filled" {...props} />;
}


const useStyles = makeStyles((theme) => ({
  root: {
    width: '100%',
    '&> * + *': {
      marginTop: theme.spacing(2),
    },
  },
}));
```

```
function SeekerPage(props) {
    const userState = useSelector(state => state.user);
    console.log(props.user)
    const seekerAcc = props.project.project.eth;//Change default value
    const projectNo = 0;//Change the Project number as variable!!!!!!!!!!!
    // const totalRequiredTokens = 10000;//Change this also
    let tokenPrice= 1000000000000000;
    const [open,setOpen] = useState(false)
    const [openMsg,setOpenMsg] = useState(false)
    const [type,setType] = useState('');
    const [message,setMessage] = useState('');


    const handleClose = (event, reason) => {
        if (reason === 'clickaway') {
          return;
        }
        setOpenMsg(false);
      };


    const sanctionedDate = props.project.project.sanctionedDate;//Change this


    const web3 = new Web3("http://localhost:7545")
    const loadbc = async (event)=>{
        window.accounts = await web3.eth.getAccounts();
        connectToContract();
        console.log(window.accounts[0])
    }
    const connectToContract = async()=>{
        $.getJSON('/Token.json',(token)=>{
            window.tokenInst = TruffleContract(token)
            console.log("Amaan ba")
            console.log(window.tokenInst)
            window.tokenInst.setProvider("http://localhost:7545")
            window.tokenInst.deployed().then(async(token)=>{
                window.TokenInstance = token
                console.log('Token address is:'+token.address)
                console.log("Seeker account: "+seekerAcc)

  window.totalRequiredTokens = await window.TokenInstance.totalrequired(seekerAcc
,projectNo)
                console.log("Initial call: "+window.totalRequiredTokens)
            })
        })
    }


    //Sets the required amount of tokens for project
    const setRequired = async()=>{
```

```
        window.tokenInst.deployed().then(async(token)=>{
            const required_tokens = $('#setRequired').val()
           await token.setRequired(seekerAcc,projectNo,required_tokens,{
               from: seekerAcc

           })
            const numbertest = await token.required(seekerAcc,projectNo);


            console.log("set number is: "+numbertest)

    })
}


    const returnMoney = async()=>{
        console.log("Token isnsyasgyas")
        console.log(window.tokenInst)
        window.tokenInst.deployed().then(async(token)=>{
            window.TokenInstance = token;


            //Logic to calculate token price at the end of return
            let dateNow = Date.now()
            console.log(typeof sanctionedDate)
            let timeElapsed = dateNow-sanctionedDate;//Change
            let diffDays = Math.ceil(timeElapsed / (1000 * 60 * 60 * 24));


            tokenPrice = tokenPrice + (diffDays*6881310000000); //1 rupee
equivalent
            console.log("tpp ttejoig")
            console.log(typeof diffDays)

            let amount_to_return = Number(window.totalRequiredTokens)
* tokenPrice
            console.log("amoutn ttejoig")
            console.log("Tokes is: "+window.totalRequiredTokens)
            token.payToSmartContract.sendTransaction({
                    from: seekerAcc,
                    to: token.address,
                    value: amount_to_return
                }).then(async()=>{
                    console.log("")
                    let length;
                    let investorsAcc = [];

  length = await window.TokenInstance.returnRegisterLength.call(seekerAcc,project
No);
                    for(let i=0;i<length;i++){
```

```
 let investorFromRegister = await window.TokenInstance.investorsRegister(seekerAc
c,projectNo,i);
                        investorsAcc.push(investorFromRegister);
                    }


                    //Filters and gives unique addresses
                    let uniqueInvestors = [...new Set(investorsAcc)];
                    console.log(uniqueInvestors)
                    for(let j=0;j<uniqueInvestors.length;j++){

  let valueContributed = await window.TokenInstance.purchased(seekerAcc,projectNo
,uniqueInvestors[j])

  await window.TokenInstance.payToSeeker(uniqueInvestors[j],Number(valueContribut
ed),{
                            from: uniqueInvestors[j]
                        })

  await window.TokenInstance.makeZero.call(uniqueInvestors[j],seekerAcc,projectNo
)
                    }
                })

        })
        setType('success')
        setMessage('Money returned back to the investors');
        setOpenMsg(true);
    }


    useEffect(loadbc,[])
    return(
        <div style={{float: 'right',marginTop:'10px'}}>

  <Button variant="contained" color="primary" onClick={() => setOpen(true)}>
            Seeker
            </Button>
                 <Dialog
                open={open}
                onClose={() => setOpen(false)}
                onOpen={() => setOpen(true)}

                >
            <DialogTitle>Seeker</DialogTitle>



            <DialogContent dividers={'paper'}>
            <p>
            <div>
```

```
  <img style={{height:"260px",width:"260px",float:"left",marginLeft:"140px"}} src
={"https://www.reuters.com/resizer/aNcj2Z2FOLjaOBlN8TCOLeF18Hs=/960x0/cloudfront-
us-east-2.images.arcpublishing.com/reuters/QFS5CLPP3BNR3HJ2YF3DZRLVRA.jpg"}/>

  <p style={{fontSize: '20px',float:"right"}}>Hi {props.user.user.name}, this is
your portal to return money back to your investors after the covid wave is over.
To return it back to investors along with a certain interest,
click <b>Return</b></p>


  <Button style={{fontSize: '14px',marginLeft: '230px',backgroundColor:'green',fl
oat:"left"}} variant="contained" color="secondary" id="returnMoney" onClick={retu
rnMoney}>Return</Button>
            </div>
            </p>
            </DialogContent>
            <DialogActions>
            <Button negative onClick={() => setOpen(false)}>
            Cancel
            </Button>
            </DialogActions>
            </Dialog>

  <Snackbar open={openMsg} autoHideDuration={4000} onClose={handleClose}>
              <Alert onClose={handleClose} severity={type}>
                  {message}
              </Alert>
            </Snackbar>
            </div>

    )
}


export default SeekerPage
```

transaction_provider.js

```
import React, { useState,useEffect} from 'react'
import ApiService from '../../api.service'
import Web3 from 'web3'
import $ from 'jquery'
import {useSelector} from 'react-redux'
import styles from './transaction.module.css'
import Button from '@material-ui/core/Button';
import Dialog from '@material-ui/core/Dialog';
import DialogActions from '@material-ui/core/DialogActions';
import DialogContent from '@material-ui/core/DialogContent';
import DialogContentText from '@material-ui/core/DialogContentText';
import DialogTitle from '@material-ui/core/DialogTitle';
import axios from 'axios'
import config from '../../config'


import Snackbar from '@material-ui/core/Snackbar';
import MuiAlert from '@material-ui/lab/Alert';
import { makeStyles } from '@material-ui/core/styles';


import TruffleContract from '@truffle/contract';


function Alert(props) {
  return <MuiAlert elevation={6} variant="filled" {...props} />;
}


const useStyles = makeStyles((theme) => ({
  root: {
    width: '100%',
    '&> * + *': {
      marginTop: theme.spacing(2),
    },
  },
}));
// import { Button, Header, Image, Modal } from 'semantic-ui-react'


// import { Dialog } from '@material-ui/core'
```

```
function Investor(props) {


    const classes = useStyles();


    const handleClick = () => {
        setOpen(true);
    };



    const userState = useSelector(state => state.user);
    const [providerAcc,setProviderAcc] = useState('');
    const [isBuyAllowed,setBuy] = useState(true);
    const [isWithdrawAllowed,setWithdraw] = useState(true);
    const [tokensPurchased,setTP] = useState('');
    const [tokensRequired,setTRfunc] = useState('')
    const [open,setOpen] = useState(false)
    const [openMsg,setOpenMsg] = useState(false)
    const [type,setType] = useState('');
    const [message,setMessage] = useState('');


    const web3 = new Web3("http://localhost:7545")
    const seekerAcc = props.project.project.eth;
    console.log("Seeker acc:"+seekerAcc)
    const investorAcc = props.user.user.eth;
    const projectNo = 0;


    const tokenPrice= 1000000000000000;


    const handleClose = (event, reason) => {
        if (reason === 'clickaway') {
          return;
        }
        setOpenMsg(false);
      };

    const setBuyFunc = async()=>{

  const requiredTokens = await window.TokenInstance.required(seekerAcc,projectNo)

        if(requiredTokens==0){
            setBuy(false);
        } else {
            setBuy(true);
```

```
        }
        setTRfunc(Number(requiredTokens));
        setWithdrawFunc();
    }


    const setWithdrawFunc = async()=>{

  const purchased = await window.TokenInstance.purchased(seekerAcc,projectNo,inve
storAcc);
        if(purchased==0){
            setWithdraw(false);
        } else {

  const requiredTokens = await window.TokenInstance.required(seekerAcc,projectNo)

            if(requiredTokens==0){
                setWithdraw(false);
            } else {
                setWithdraw(true);
            }
        }
        setTP(Number(purchased))
    }

    const loadbc = async (event)=>{
        window.accounts = await web3.eth.getAccounts();
        await connectToContract();

        console.log(window.accounts[0])
    }

    const connectToContract = async()=>{
        $.getJSON('/Token.json',(token1)=>{
           window.tokenInst =TruffleContract(token1);
           console.log("inga paaru")
           console.log(window.tokenInst)
           window.tokenInst.setProvider("http://localhost:7545")
            window.tokenInst.deployed().then(async(token)=>{
                window.TokenInstance = token

  window.totalRequiredTokens = await window.TokenInstance.totalrequired(seekerAcc
,projectNo)
                setBuyFunc();
            })
        })

    }
    useEffect(loadbc,[])


    const buy = async ()=>{
```

```
        let noOfTokens = $("#amount").val();
        let inWei = String($("#amount").val() * tokenPrice);

  window.required_tokens = await window.TokenInstance.required(seekerAcc,projectN
o);
        window.required_tokens = parseInt(window.required_tokens.toString())
         //Change this for sureeeeeee
        if(noOfTokens<=window.required_tokens){

            await window.TokenInstance.payToSmartContract.sendTransaction({
                from: investorAcc,
                to: window.TokenInstance.address,
                value: inWei
            })

  await window.TokenInstance.changeVariables(investorAcc,seekerAcc,projectNo,noOf
Tokens,{
                from: investorAcc
            });
            // const addTransaction =
await ApiService.addTransaction(seekerAcc,props.user.user.id)
            setMessage(`${noOfTokens} tokens bought successfully!`);
            setType("success")
            setOpenMsg(true);
            console.log(window.required_tokens)
        if(noOfTokens==(window.required_tokens)){
            console.log("damnnnnn it works")

  await window.TokenInstance.payToSeeker(seekerAcc,(window.totalRequiredTokens),{

                from: seekerAcc
            })
            axios({
                method: 'POST',
                url: config.BASE_URL+'project/sanction',
                headers: {"Content-Type" : "application/json"},
                data: {
                    project: props.project.project._id
                }
            })
            .then((res)=>{
                setMessage("Transfered ether to seeker account from smart
contract");
                setType("success")
                setOpenMsg(true);
                // console.log("Transfered ether to seeker account from smart
contract")
                window.required_tokens-=noOfTokens;
            })
            .catch((e)=>{
                console.log(e)
            })
```

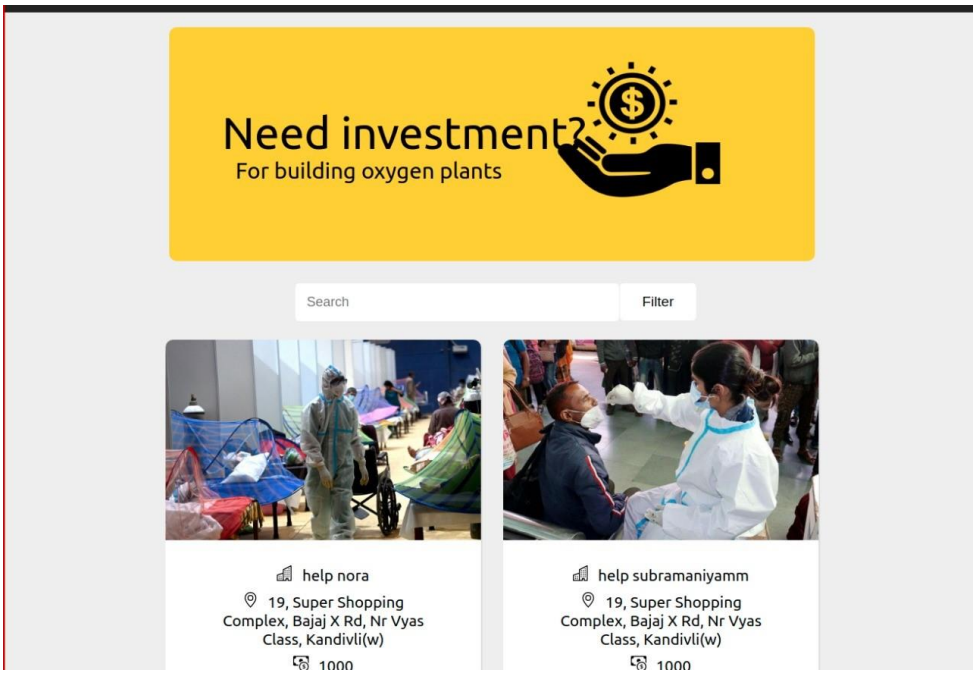
```
        }

        } else{
            setMessage("Enter lesser value");
            setType("error")
            setOpenMsg(true);
        }

  let pur1 = await window.TokenInstance.purchased(seekerAcc,projectNo,investorAcc
);
        setTP(Number(pur1))


        let mr1 = await window.TokenInstance.required(seekerAcc,projectNo)
        setTRfunc(Number(mr1))
    }


    const withdraw = async()=>{

  const moreRequired = await window.TokenInstance.required(seekerAcc,projectNo)
        console.log(Number(moreRequired))
        if(moreRequired==0){
            setMessage("Sorry cannot withdraw, the project has been
sanctioned");
            setType("error")
            setOpenMsg(true);
            // console.log("Sorry cannot withdraw, the project has been
sanctioned")
        } else{

  let purchased = await window.TokenInstance.purchased(seekerAcc,projectNo,invest
orAcc);
            console.log(Number(purchased))
            if(purchased!=0){
                purchased=purchased

  await window.TokenInstance.payToSeeker(investorAcc,Number(purchased),{
                    from:investorAcc
                })

  await window.TokenInstance.withdraw(investorAcc,seekerAcc,projectNo,{
                    from:investorAcc
                });
            }
            else{
                setMessage("Sorry you cannot withdraw ether as investment=0");
                setType("error")
                setOpenMsg(true);
            }

        }
```

```
  let pur = await window.TokenInstance.purchased(seekerAcc,projectNo,investorAcc)
;
        setTP(Number(pur))


        let mr = await window.TokenInstance.required(seekerAcc,projectNo)
        setTRfunc(Number(mr))
    }



    return(
        <div style={{float: 'right',marginTop:'10px'}}>
            {/*
<Snackbar open={open} autoHideDuration={6000} onClose={handleClose}>
                <Alert onClose={handleClose} severity="success">
                This is a success message!
                </Alert>
            </Snackbar> */}

  <Button variant="contained" color="primary" onClick={() => setOpen(true)}>Inves
tor</Button>
            <Dialog
           open={open}
           onClose={() => setOpen(false)}
           onOpen={() => setOpen(true)}
           >
       <DialogTitle>Investor</DialogTitle>



       <DialogContent dividers={'paper'}>
       <p>
       <div>

 <img style={{height:"260px",width:"260px",marginLeft:"140px"}} src={"https://www
.reuters.com/resizer/aNcj2Z2FOLjaOBlN8TCOLeF18Hs=/960x0/cloudfront-us-east-
2.images.arcpublishing.com/reuters/QFS5CLPP3BNR3HJ2YF3DZRLVRA.jpg"}/>


            <ul>
                <li style={{fontSize: '20px'}}>Tokens owned by
you: {tokensPurchased}</li>
            </ul>
            <ul>
                <li style={{fontSize: '20px'}}>Remaining tokens
required: {tokensRequired}</li>
            </ul>
```

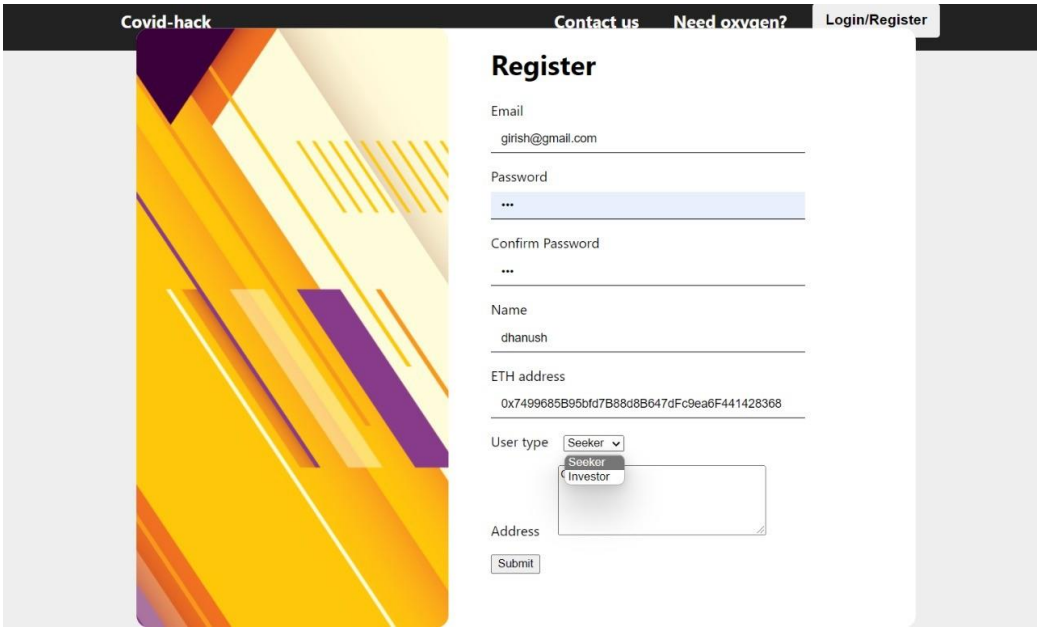
```
            <div style={{display:isBuyAllowed?'block':'none'}}>
            <p style={{fontSize: '18px'}}>Please enter the amount of tokens you
wish to purchase</p>
            <span style={{fontSize: '20px'}}>Amount:
 </span><input style={{marginTop: '10px',fontSize:'20px'}}  className={styles.fie
ldIn} id="amount"></input>
            <div style={{float:'right', marginRight: '300px'}}>

  <Button variant="contained" onClick={buy} style={{backgroundColor:'green',color
:'whitesmoke'}}>Buy</Button>
            </div>
            </div>

            <p style={{fontSize: '18px',marginTop:'20px'}}>Would you like to
withdraw the investment you have made? If yes click withdraw</p>
            <div>

  <Button variant="contained"  style={{marginLeft:'210px',backgroundColor: "orang
e",color:'whitesmoke'}} onClick={withdraw}>Withdraw</Button>
            </div>

        </div>

       </p>
       </DialogContent>


       <DialogActions>
       <Button negative onClick={() => setOpen(false)}>
       Cancel
       </Button>
       </DialogActions>
       </Dialog>
        <Snackbar open={openMsg} autoHideDuration={4000} onClose={handleClose}>
            <Alert onClose={handleClose} severity={type}>
                {message}
            </Alert>
        </Snackbar>
       </div>
    )
}


export default Investor
```

7.2 APPENDIX 2 - SCREEN SHOTS /OUTPUTs

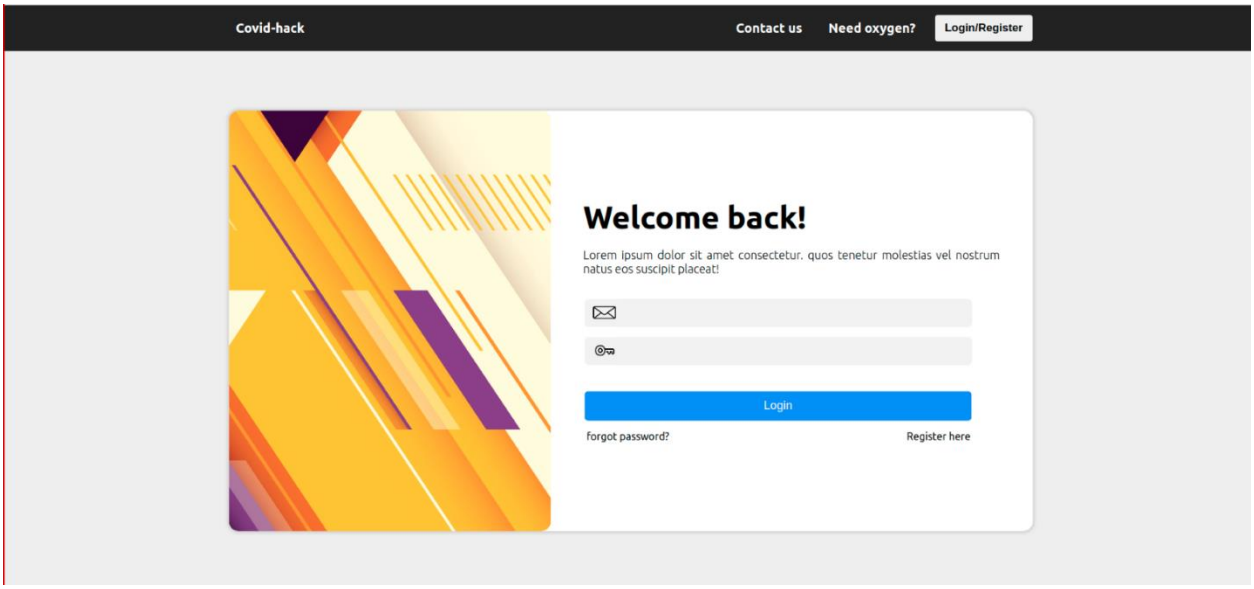
Home page



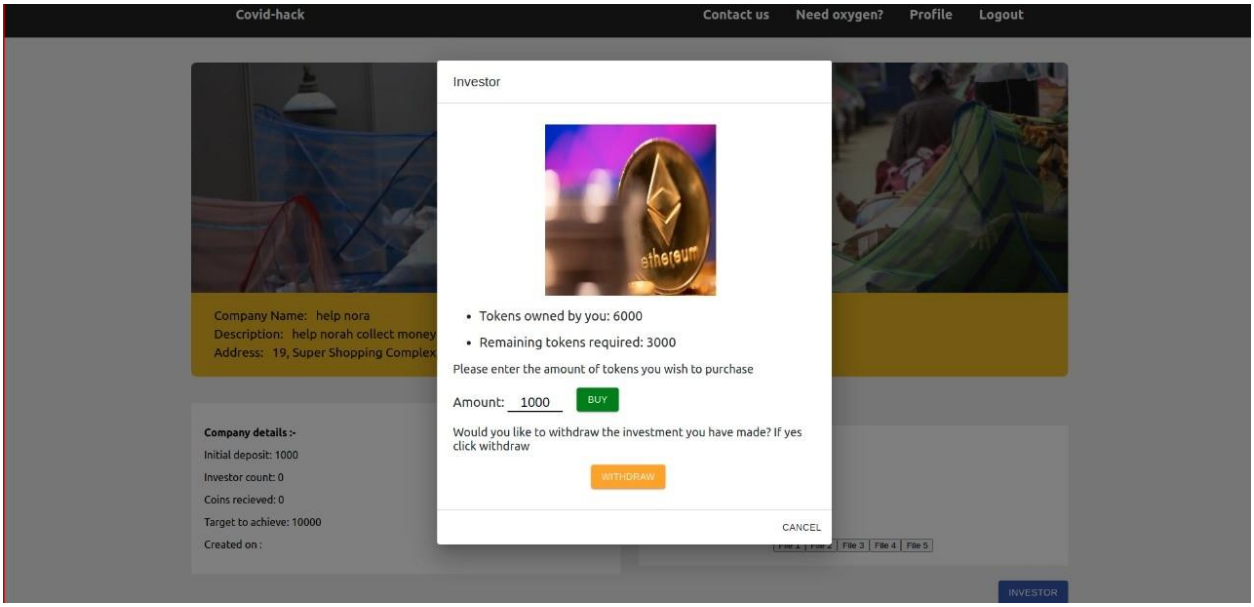
Registration page



Login page



Project page



Profile page to add government verifid documents for proof

Licensing/ Registration

All the major necessary licenses and permits are to be applied. A list of the same is provided below.

Description	Status
Village NOC	From Gaon Panchayat
Trade License	From Local Authority
Consent to Establish (Pollution NOC)	State Pollution Control Board
Factory License	Approach to Chief Inspection of Factories, State Govt
Power Load	Approach to Power Distribution Company

View all files: Stage 1 Stage 2 Stage 3 Stage 4 Stage 5

Click here to open file 1

Licensing/ Registration

All the major necessary licenses and permits are to be applied. A list of the same is provided below.

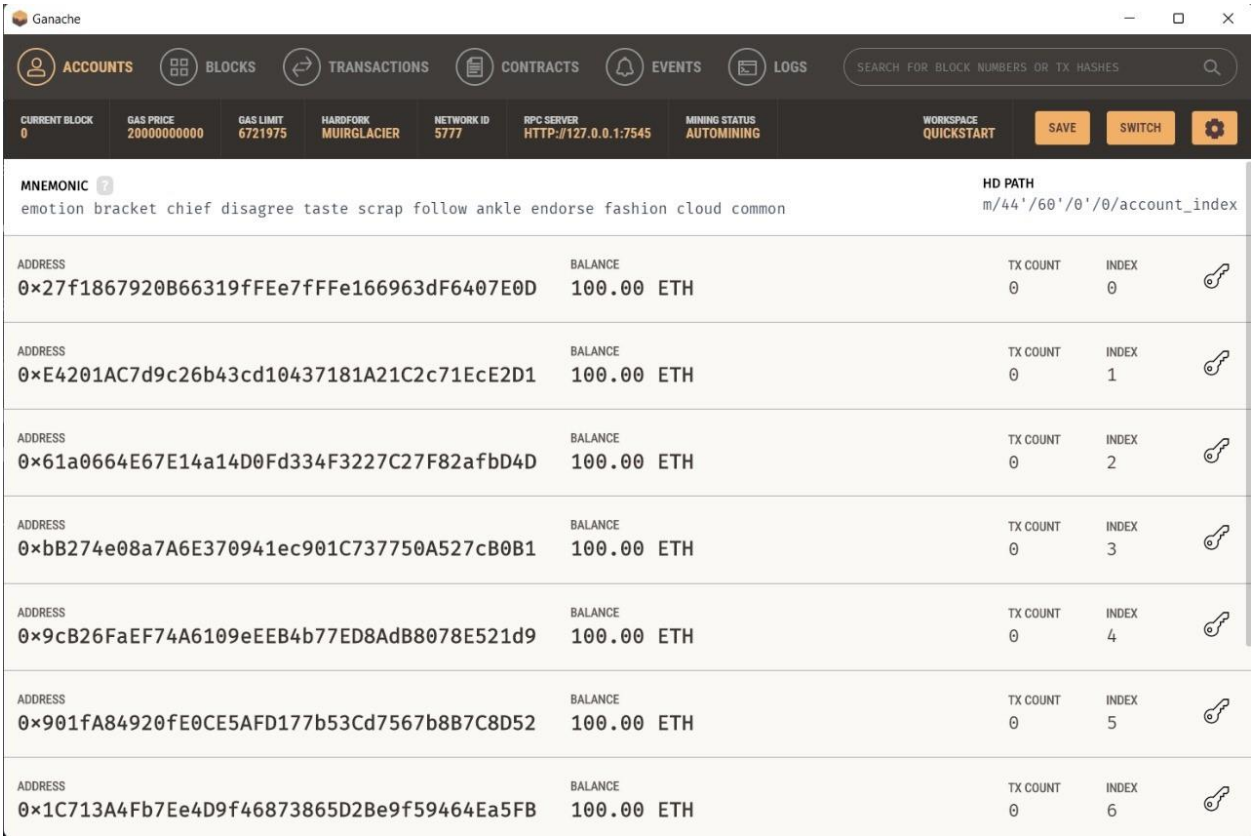
Description	Status
Village NOC	From Gaon Panchayat
Trade License	From Local Authority
Consent to Establish (Pollution NOC)	State Pollution Control Board
Factory License	Approach to Chief Inspection of Factories, State Govt
Power Load	Approach to Power Distribution Company

Stage 1 Stage 2 Stage 3 Stage 4 Stage 5

Select file for stage 1

Choose File No file chosen submit

Ganache where 10 ethereum addresses are created in the test blockchain which it sets up



8.REFERENCES

8.1. LIST OF WEBSITES

1.https://www.academia.edu/50977209/Blockchain_based_crowdfunding_systems

2.https://www.researchgate.net/publication/275657510_Crowdfunding_The_Current_State_Of_Research

3.https://www.researchgate.net/publication/318307115_APPLICATION_OF_BLOCKCHAIN_TECHNOLOGY_IN_CROWDFUNDING

4. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3133176

5. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2725373

6.https://www.academia.edu/43600883/Smart_Contract_and_Blockchain_for_Crowdfunding_Platform