

CSE1006
Blockchain and Cryptocurrency
Technologies

THEORY DIGITAL ASSIGNMENT

Decentralized App - goSHARE

Submitted to: Jayakumar S

Name- Saurabh Singh

Reg No. - 19BCI0184

ABSTRACT

When several parties, possibly located all over the world, need to share data and transfer value without having to trust one other, blockchains are employed.

The counterparty risk, as defined by the financial sector, is the danger that the other party will not keep their half of the contract. Through a new combination of mathematics, cryptography, and peer-to-peer networking, blockchains fully eliminated the counterparty risk.

In this project we propose a blockchain-based platform for user modelling that allows users to contribute data without losing control or ownership. This platform addresses three major issues: maintaining privacy and user control, as well as providing sharing incentives.

Introduction

Blockchain is a data structure used to create a public or private distributed digital transaction ledger which, instead of resting with a single provider, is shared among a distributed network of computers.

Blockchains eliminate the problem of trust that affect other databases in the following ways:

- Full decentralization
- Extreme fault tolerance
- Independent verification

How a blockchain works ?

"Transactions" are the interactions between accounts in a blockchain network. They could be monetary transactions, such as transmitting ether, the Ethereum cryptocurrency. They could also be data transmissions like a comment or a user name. A "block" is a collection of transactions.

On the blockchain, each account has a unique signature that identifies whose account began the transaction. Anyone can read or write data on a public blockchain. Reading data on the public blockchain is free, but writing to it is not. This fee, called as "gas" and priced in ether, deters spam and pays for network security.

How a Transaction works ?

1. Bob attempts to send Alice 1 ETH
 2. Bob and Alice's transaction is combined with other transactions that have occurred since the last block
 3. Miners compete to validate the block with the new set of transactions
 4. The victorious miner creates a new block and receives a reward.
 5. With the transaction validated, Alice receives 1 ETH.
-

What is a smart contract?

A smart contract is code that runs on the EVM. Smart contracts can accept and store ether, data, or a combination of both. Then, using the logic programmed into the contract, it can distribute that ether to other accounts or even other smart contracts.

Smart contracts are written in a language called Solidity. Solidity is statically typed, and supports inheritance, libraries, and complex user-defined types, among much else. Solidity syntax is similar to JavaScript.

What is Ethereum?

Ethereum is a blockchain that allows you to run programs in its trusted environment. Ethereum has a virtual machine, called the Ethereum Virtual Machine (EVM). The EVM allows code to be verified and executed on the blockchain, providing guarantees it will be run the same way on everyone's machine. This code is contained in "smart contracts" .

Decentralized applications (dapps)

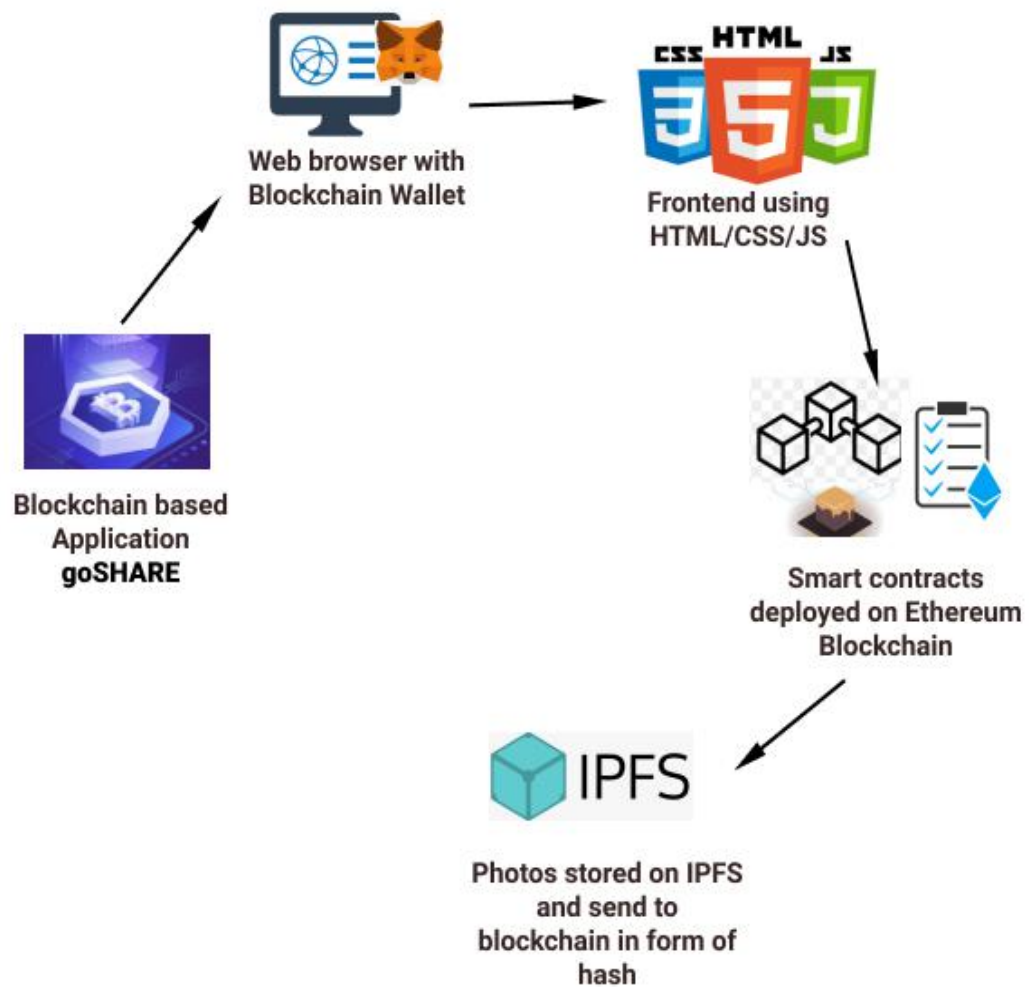
Applications using smart contracts for their processing and/or datastorage are called "decentralized applications", or "dapps". The user interfaces for these dapps consist of familiar languages such as HTML, CSS, and JavaScript.

A dapp could be solution for many industries such as Record keeping, Finance, Supply chains, Real Estate, Marketplace.

Project:

A peer to peer photo sharing platform which allows users to upload their clicked photos on blockchain and earn incentives if other people on network likes their photo and donate them in form of cryptocurrencies. This platform will be censorship resistant, nobody could remove the data when uploaded.

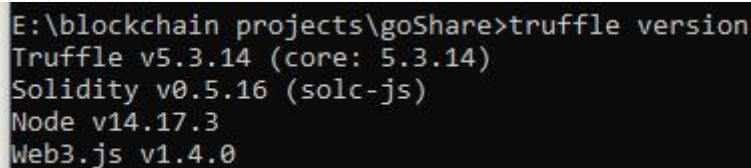
Architecture:



Modules:

Setting up the development environment

- Nodejs and npm
- Git
- **Truffle** (npm install -g truffle) -
 - Truffle is a world-class development environment, testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM). Open source framework for rapid Dapp development.



```
E:\blockchain projects\goShare>truffle version
Truffle v5.3.14 (core: 5.3.14)
Solidity v0.5.16 (solc-js)
Node v14.17.3
Web3.js v1.4.0
```

- **Ganache**
 - A personal Ethereum blockchain which you can use to run tests, execute commands, and inspect state while controlling how the chain operates. In blockchain we have to pay gas fees for every function or transaction and Ganache provides us Ethereum(fake for testing) which enable us to develop/deploy our dapp.
-

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK
140

GAS PRICE
20000000000

GAS LIMIT
6721975

HARDFORK
MUIRGLACIER

NETWORK ID
5777

RPC SERVER
HTTP://127.0.0.1:7545

MINING STATUS
AUTOMINING

WORKSPACE
PUMPED-BALLS

SWITCH

MNEMONIC

olive expose similar course hammer under tower knock ordinary canal canyon library

HD PATH

m/44'/60'/0'/0/account_index

ADDRESS

BALANCE

TX COUNT

INDEX

0x5B98cc5Bfac377544422Dd8eD45a103e2B40366f

99.69 ETH

120

0

ADDRESS

BALANCE

TX COUNT

INDEX

0x54860Ca7F9C8E4Edcdd41343a48BF3B9eF4A121A

99.99 ETH

13

1

ADDRESS

BALANCE

TX COUNT

INDEX

0xD3A582793e443f4934802B26057f62D281153295

99.89 ETH

7

2

ADDRESS

BALANCE

TX COUNT

INDEX

0x0922310A71Ee729b6E3C873A358ba3ab56dF9D4e

100.00 ETH

0

3

ADDRESS

BALANCE

TX COUNT

INDEX

0x1A60B81f961212871F7c0538b3cf3F416C65a2E1

100.00 ETH

0

4

ADDRESS

BALANCE

TX COUNT

INDEX

0xC79562A1FcFcAbAD604806b5cd77DEDd10FfC08e

100.00 ETH

0

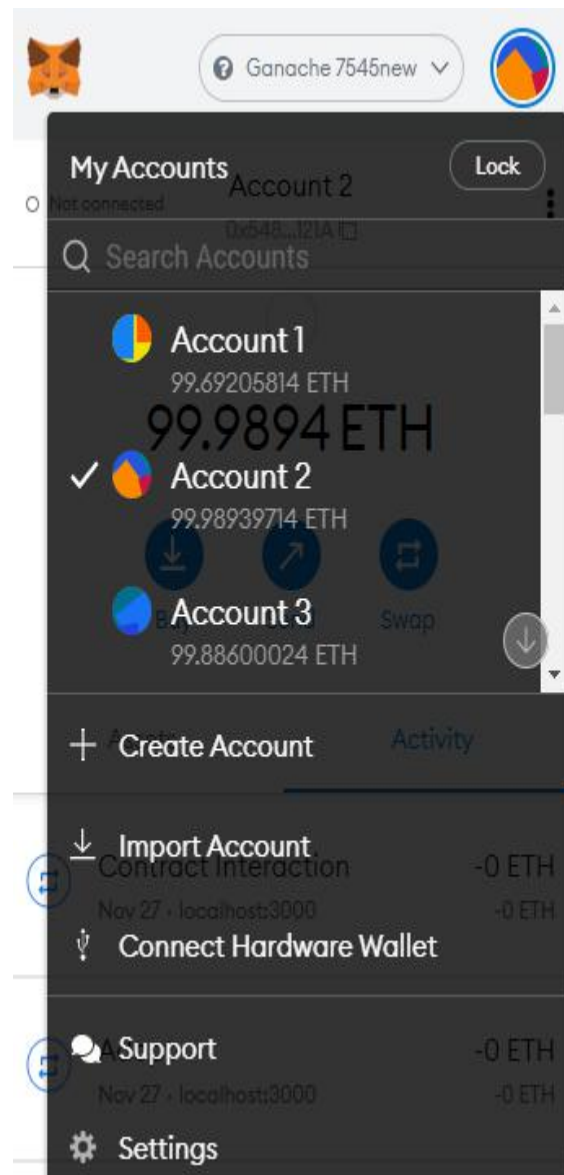
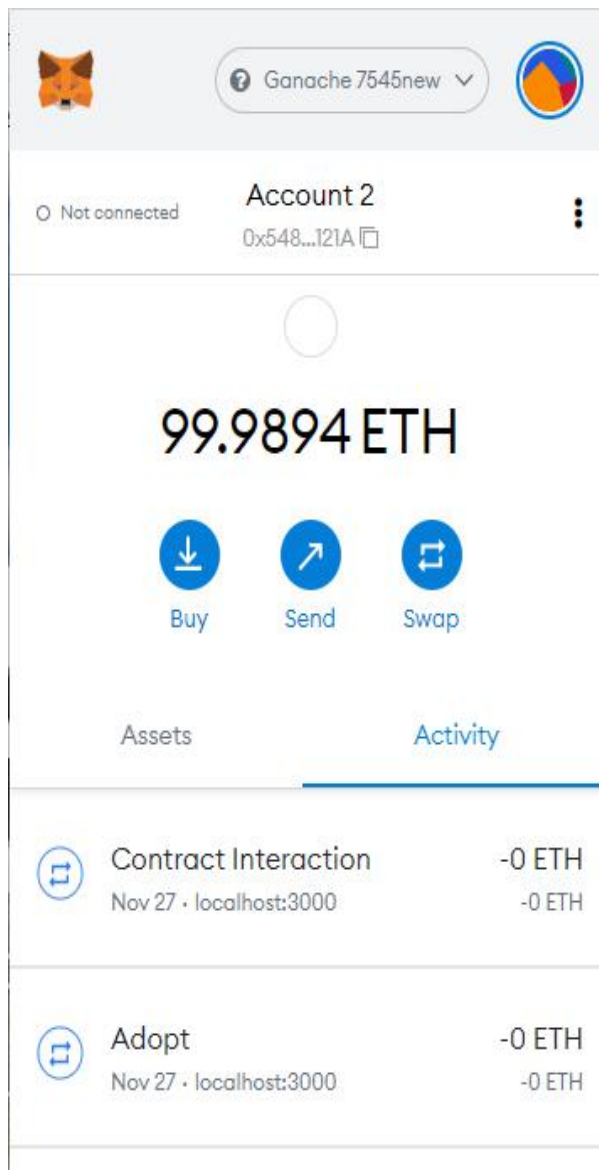
5

● Web3js

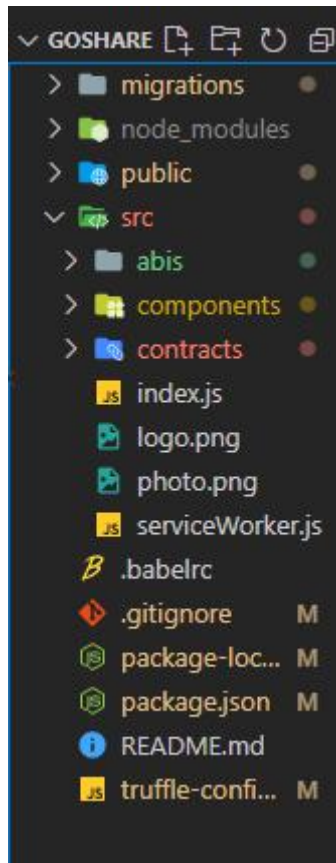
- It's a collection of libraries which will help us to interact with a local or remote ethereum node. Basically when we are programming in our system on truffle framework, we have to write a web3 program from client side which will interact with browser using blockchain wallets such as Metamask.

● Metamask

- Cryptowallet used to interact with Ethereum blockchain. It allows users to access their Ethereum wallet through a browser extension or mobile app, which can then be used to interact with decentralized applications.



Folder Structure:



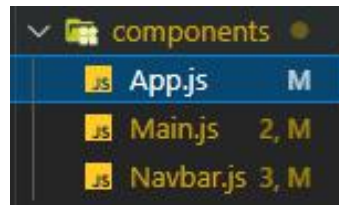
Migrations - Truffle uses a migration system to handle smart contract deployments. A migration is an additional special smart contract that keeps track of changes.

```
const goSHARE = artifacts.require("../src/contracts/goSHARE");

module.exports = function(deployer) {
  deployer.deploy(goSHARE);
};
```

public - contains the index.html file startup for frontend

(FRONTEND) components - contains code for frontend and for connecting app to browser blockchain



App.js

A global App object to manage our application. The web3 JavaScript library interacts with the Ethereum blockchain. It can retrieve user accounts, send transactions, interact with smart contracts, and more.

```
const ipfsClient = require('ipfs-http-client')
const ipfs = ipfsClient({ host: 'ipfs.infura.io', port: 5001, protocol: 'https' }) /
```

Declaring the IPFS here which is a file sharing system that can be leveraged to more efficiently store and share large files. It relies on cryptographic hashes that can easily be stored on a blockchain.

Instantiating the web3

We are using modern dapp browsers or the more recent versions of MetaMask where an ethereum provider is injected into the window object. If so, we use it to create our web3 object, but we also need to explicitly request access to the accounts with `ethereum.enable()`.

```
class App extends Component {  
  
  async componentWillMount() {  
    await this.loadWeb3()  
    await this.loadBlockchainData()  
    this.captureUpload()  
  }  
  async loadWeb3() {  
    if (window.ethereum) {  
      window.web3 = new Web3(window.ethereum)  
      await window.ethereum.enable()  
    }  
    else if (window.web3) {  
      window.web3 = new Web3(window.web3.currentProvider)  
    }  
    else {  
      window.alert('Non-Ethereum browser detected. You should consider trying  
MetaMask!')  
    }  
  }  
}
```

The function to load the blockchain data that is stored after the smart contracts are deployed to the network, it loads and connects to the Metamask Account.

```
async loadBlockchainData() {  
  const web3 = window.web3  
  // Load account  
  const accounts = await web3.eth.getAccounts()  
  this.setState({ account: accounts[0] })  
  // Network ID
```

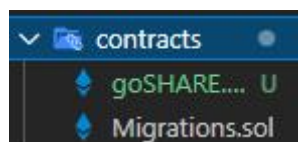
Here we are capturing the image uploaded using a form in **Main.js** and then adding the image and the description to **IPFS**.

```
captureFile = event => {
  event.preventDefault()
  const file = event.target.files[0] //reading the file from html
  const reader = new window.FileReader() //js function to read file
  reader.readAsArrayBuffer(file) //reading it as an array buffer

  //converting it in a format that the ipfs can understand
  reader.onloadend = () => {
    this.setState({ buffer: Buffer(reader.result) })
    console.log('buffer', this.state.buffer)
  }
}

uploadImage = description => {
  console.log("Submitting file to ipfs...")
  //adding file to the IPFS
  ipfs.add(this.state.buffer, (error, result) => {
    console.log('Ipfs result', result)
    if(error) {
      console.error(error)
      return
    }
  })
}
```

Contracts: These contracts work as a backend for this application.



goSHARE.sol

```
contract goSHARE {
  string public name;
  uint public imageCount = 0;
  mapping(uint => Image) public images;
```

Initializing the contract and declare the images in the form of mapping and public so that can be accessed from throughout the code.

```
struct Image {  
    uint id;  
    string hash;  
    string description;  
    uint tipAmount;  
    address payable author;  
}
```

Solidity allows user to create their own data type in the form of structure. The struct contains a group of elements with a different data type.

```
event ImageCreated(  
    uint id,  
    string hash,  
    string description,  
    uint tipAmount,  
    address payable author  
);
```

Event is an inheritable member of a contract. An event is emitted, it stores the arguments passed in transaction logs. These logs are stored on blockchain and are accessible using address of the contract till the contract is present on the blockchain.

```
function uploadImage(string memory _imgHash, string memory _description) public {  
    // Make sure the image hash exists  
  
    function tipImageOwner(uint _id) public payable {  
        // Make sure the id is valid
```

Functions to upload image and donate tip to the user who uploaded the photo.

Source Code :

Github Link- <https://github.com/baazis/goSHARE>

1- goSHARE.sol

```
pragma solidity ^0.5.0;

contract goSHARE {
    string public name;
    uint public imageCount = 0;
    mapping(uint => Image) public images;
    struct Image {
        uint id;
        string hash;
        string description;
        uint tipAmount;
        address payable author;
    }
    event ImageCreated(
        uint id,
        string hash,
        string description,
        uint tipAmount,
        address payable author
    );
    event ImageTipped(
        uint id,
        string hash,
        string description,
        uint tipAmount,
        address payable author
    );
    constructor() public {
        name = "goSHARE";
    }
    function uploadImage(string memory _imgHash, string memory _description)
    public {
        // Make sure the image hash exists
        require(bytes(_imgHash).length > 0);
        // Make sure image description exists
        require(bytes(_description).length > 0);
        // Make sure uploader address exists
        require(msg.sender!=address(0));
        // Increment image id
        imageCount ++;
        // Add Image to the contract
```



```

    images[imageCount] = Image(imageCount, _imgHash, _description, 0,
msg.sender);
    // Trigger an event
    emit ImageCreated(imageCount, _imgHash, _description, 0, msg.sender);
}
function tipImageOwner(uint _id) public payable {
    // Make sure the id is valid
    require(_id > 0 && _id <= imageCount);
    // Fetch the image
    Image memory _image = images[_id];
    // Fetch the author
    address payable _author = _image.author;
    // Pay the author by sending them Ether
    address(_author).transfer(msg.value);
    // Increment the tip amount
    _image.tipAmount = _image.tipAmount + msg.value;
    // Update the image
    images[_id] = _image;
    // Trigger an event
    emit ImageTipped(_id, _image.hash, _image.description, _image.tipAmount,
_author);
}
}

```

2- deploy-contracts.js

```

const goSHARE = artifacts.require("../src/contracts/goSHARE");

module.exports = function(deployer) {
    deployer.deploy(goSHARE);
};

```

3- Main.js

```

import React, { Component } from 'react';
import Identicon from 'identicon.js';

class Main extends Component {
    render() {
        return (
            <div className="container-fluid mt-5">
                <div className="row">
                    <main role="main" className="col-lg-12 ml-auto mr-auto"
style={{ maxWidth: '500px' }}>
                        <div className="content mr-auto ml-auto">

```

```

        <p>&nbsp;</p>
        <h2 style={{textAlign:'center'}}> <u>Share your Photos on
Blockchain </u></h2>
        <br></br>
        <h4>Upload your Image</h4>
        <form onSubmit={(event) => {
            event.preventDefault()
            const description = this.imageDescription.value
            this.props.uploadImage(description)
        }} >
            <input type='file' accept=".jpg, .jpeg, .png, .bmp, .gif, "
onChange={this.props.captureFile} />
            <div className="form-group mr-sm-2">
                <br></br>
                <input
                    id="imageDescription"
                    type="text"
                    ref={(input) => { this.imageDescription = input }}
                    className="form-control"
                    placeholder="Add Image description..."
                    required />
            </div>
            <button type="submit" class="btn btn-warning btn-block btn-
sm">Upload!</button>
        </form>
        <p>&nbsp;</p>
        { this.props.images.map((image, key) => {
            return(
                <div className="card mb-4" key={key} >
                    <div className="card-header">
                        <img
                            className='mr-2'
                            width='30'
                            height='30'
                            src={`data:image/png;base64,${new
Identicon(image.author, 30).toString()}` }
                        />
                        <small className="text-muted">{image.author}</small>
                    </div>
                    <ul id="imageList" className="list-group list-group-
flush">
                        <li className="list-group-item">
                            <p class="text-center"><img
src={`https://ipfs.infura.io/ipfs/${image.hash}`} style={{ maxWidth:
'420px' }}/></p>
                            <p>{image.description}</p>
                        </li>
                        <li key={key} className="list-group-item py-2">

```

```

        <small className="float-left mt-1 text-muted">
            TIPS:
{window.web3.utils.fromWei(image.tipAmount.toString(), 'Ether')} ETH
        </small>
        <button
            className="btn btn-link btn-sm float-right pt-0"
            name={image.id}
            onClick={(event) => {
                let tipAmount = window.web3.utils.toWei('0.1',
'Ether')

                console.log(event.target.name, tipAmount)
                this.props.tipImageOwner(event.target.name,
tipAmount)
            }}
        >
            TIP 0.1 ETH
        </button>
    </li>
</ul>
</div>
)
    })}
</div>
</main>
</div>
</div>
);
}
}
export default Main;

```

Results:

Truffle console - allows us to run JS scripts on console

```
truffle(development)> gos = await goSHARE.deployed()
undefined
truffle(development)> gos
TruffleContract {
  constructor: [Function: TruffleContract] {
    _constructorMethods: {
      configureNetwork: [Function: configureNetwork],
      setProvider: [Function: setProvider],
      new: [Function: new],
      at: [AsyncFunction: at],
      deployed: [AsyncFunction: deployed],
      defaults: [Function: defaults],
      hasNetwork: [Function: hasNetwork]
```

Here we can see the JS version of the smart contract using truffle console and the address of the blockchain where it is running with its name.

```
truffle(development)> gos.address
'0x80Aeb1f77Ba16968b4d32Ff73Da467F340Dc9375'
truffle(development)> name = await gos.name()
undefined
truffle(development)> name
'goSHARE'
```


Deploying smart contract and migrating it using truffle migrate.

```
2_deploy_contracts.js
=====

Replacing 'goSHARE'
-----
> transaction hash:    0xb962b74714b3a88c4f3957fa055763d18c6a15a0f11cfe15c63ad817544a6b3b
> Blocks: 0           Seconds: 0
> contract address:   0xF69757D748DC47f419354d9fD901b93794112b43
> block number:       148
> block timestamp:    1638101495
> account:            0x5B98cc5Bfac377544422Dd8eD45a103e2B40366f
> balance:            99.64612008
> gas used:           867170 (0xd3b62)
> gas price:          20 gwei
> value sent:         0 ETH
> total cost:         0.0173434 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:         0.0173434 ETH
```

Application:

goSHARE - Decentralize your photo sharing0x5B98cc5Bfac377544422Dd8eD45a103e2B40366f

Share your Photos on Blockchain

Upload your Image


Choose FileNo file chosen



Add Image description...

Upload!

Now we chose a file and add the description of the file and click on Upload. Metamask will open as pop up and user can select to reject or accept uploading of the image.

Account 1- 0x5B98cc5Bfac377544422Dd8eD45a103e2B40366f

 Account 1

  0xF69...2b43


New address detected! Click here to add to your address book.

http://localhost:3000

CONTRACT INTERACTION

DETAILS

DATA

Estimated gas fee  0.00650114 0.00650114 ETH

Site suggested

Max fee: 0.00650114 ETH

Total

0.00650114 0.00650114 ETH

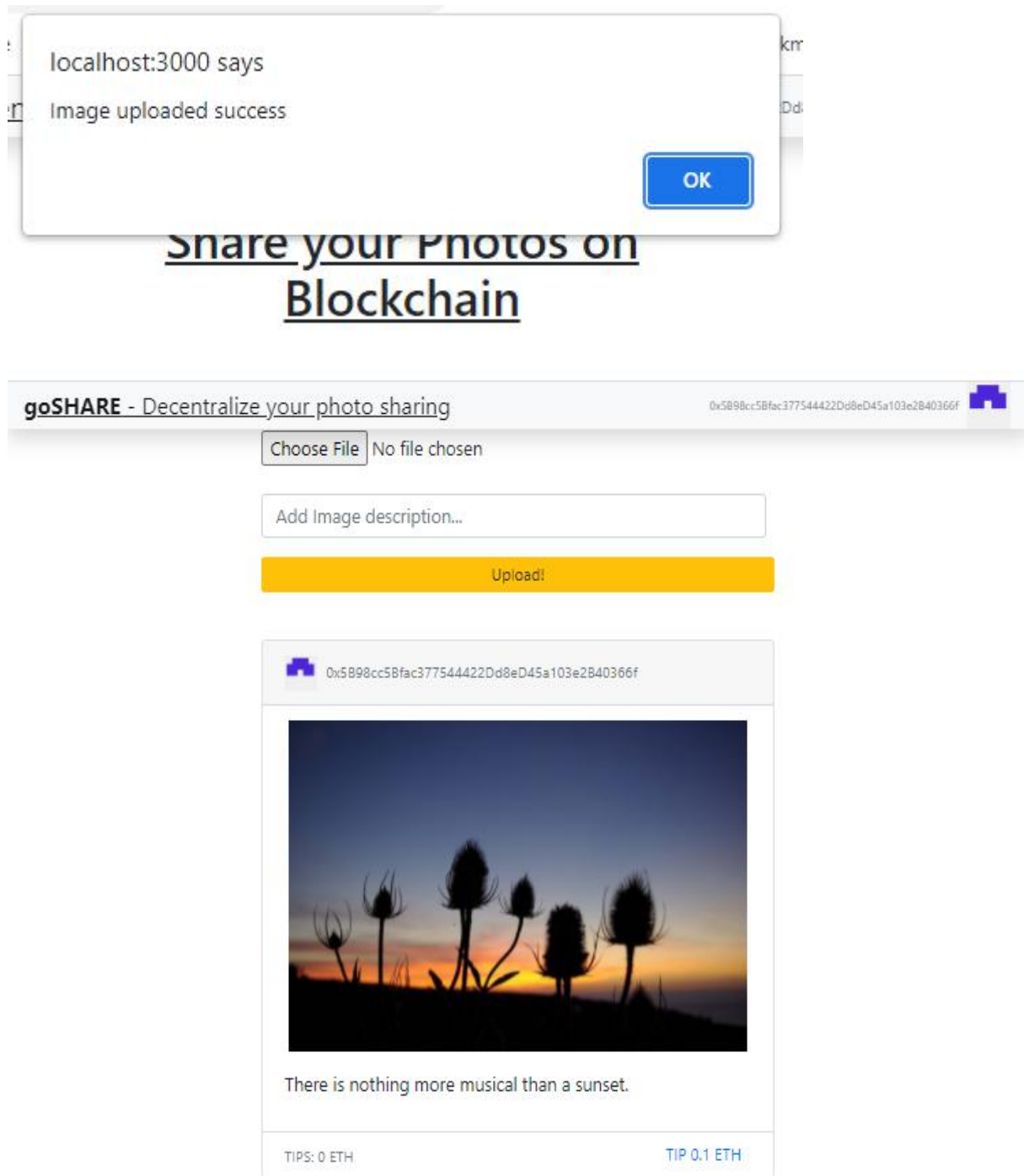
Amount + gas fee

Max amount: 0.00650114 ETH

Reject

Confirm

As Image got uploaded. We can see the image uploaded on site and other users on network can also see this and give a tip if they like it.



Now Account 2 = 0x54860Ca7F9C8E4Edcdd41343a48BF3B9eF4A121A

When a new user joins they can see all the images uploaded on site and can upload their own images.

Account 2 → 0xF69...2b43

New address detected! Click here to add to your address book.

http://localhost:3000

CONTRACT INTERACTION

DETAILS DATA

Estimated gas fee ⓘ 0.0054272 **0.0054272 ETH** [EDIT](#)
Site suggested **Max fee:** 0.0054272 ETH


Total 0.0054272 **0.0054272 ETH**
Amount + gas fee **Max amount:** 0.0054272 ETH

Reject Confirm

Similarly while uploading they have to pay a little amount of ether as a form of payment and the images will be uploaded on site. On the top right we can see the id of user who is joined.

goSHARE - Decentralize your photo sharing


0x54860Ca7F9C8E4Edcdd41343a48BF3B9eF4A121A




There is nothing more musical than a sunset.

TIPS: 0 ETH

TIP 0,1 ETH



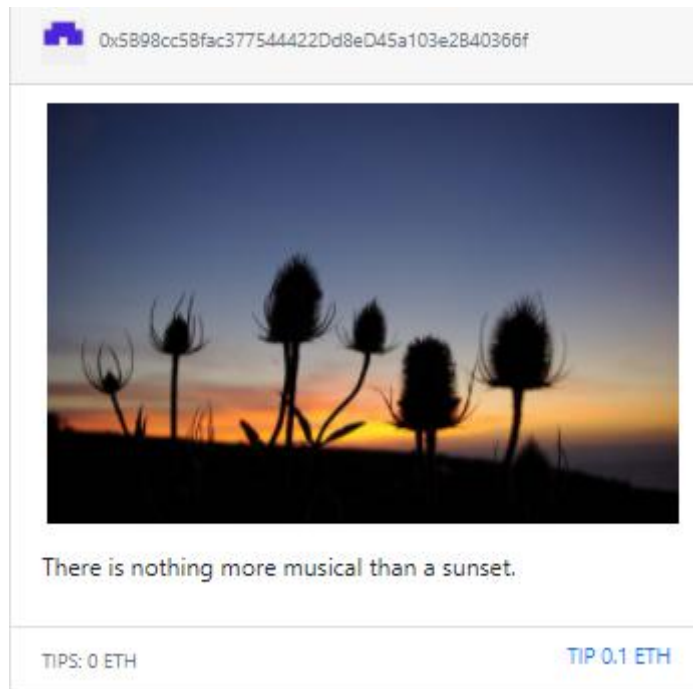
0x54860Ca7F9C8E4Edcdd41343a48BF3B9eF4A121A



A beautiful design made on Figma

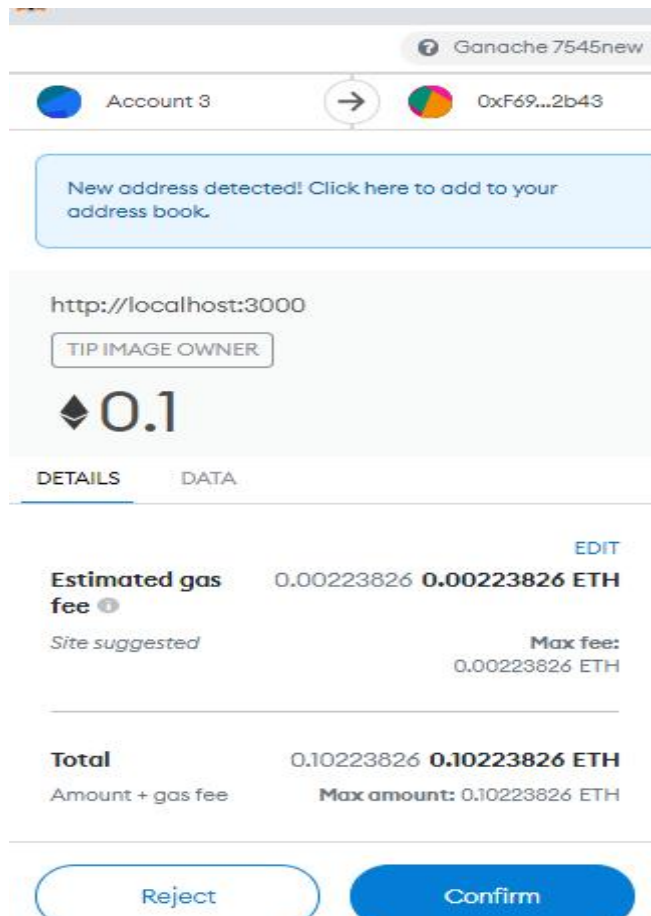
Tipping the Images:

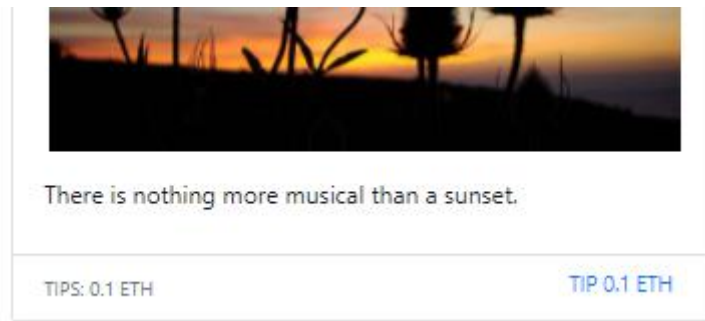
Account 3:



Tips =0.1 ETH

Now a user clicks on Tip 0.1 ETH and have to pay the same amount to the user who uploaded the image





Tip will be updated and the most tipped image will be shown on top of website.

Conclusion:

Completed making of the peer to peer sharing platform goSHARE. This platform gives an opportunity to users to join and upload images and earn using them.

Learnt about new concepts of Solidity and Blockchain. For future enhancements, I will try to make changes in the UI and upload the site online so that others can use it.

Thankyou
