DECENTRALISED SOCIAL MEDIA APPLICATION WITH REWARD SYSTEM

PROJECT REPORT

Submitted by

SRIVATSAV R (2019103066)

SACHIN RAGHUL T (2019103573)

SANJEEV K M (2019103576)

CS6611 - CREATIVE AND INNOVATIVE PROJECT

ANNA UNIVERSITY :: CHENNAI

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DECENTRALIZED SOCIAL MEDIA APPLICATION WITH REWARD SYSTEM"** is the bonafide work of **"Srivatsav R, Sachin Raghul T, Sanjeev K M"** who carried out the project work as a part of Creative and Innovative Project Laboratory.

DATE :	**SIGNATURE**
PLACE :	**Dr. K THANGARAMYA**
	COURSE IN-CHARGE
	TEACHING FELLOW
	DEPARTMENT OF CSE
	ANNA UNIVERSITY
	CHENNAI

TABLE OF CONTENTS

ABSTRACT

Social media applications like facebook provides decent service at the extremely high cost of constantly disregarding user privacy, common decency, and laws around the world, and yet they still have billions of users as they are the monopoly. Monopolies breed complacency, low quality, and high costs. And Facebook has a monopoly over your friend network. We can’t leave because everyone is on Facebook. Switching to a new platform means every single one of your friends has to make a new account, download a new app, and you have to rebuild that whole network—if you can convince them at all. It’s the same story for any platform (though most don’t charge such a high price), and it’s unavoidable. We believe the solution is a decentralized social network which is encrypted at rest. When the user has the key to decrypt and modify their own data, they have complete control, and can grant and revoke control from third parties. Everyone’s data is just ‘out there’, many copies floating around in encrypted blobs that anyone can host or download but only friends can decrypt. Decentralization also provides robustness against censorship, internet outages, and would-be social monopolies.

CHAPTER 1

INTRODUCTION

1.1 Ethegram

The key to this decentralized paradigm is not merely security, which is not too hard with public key cryptography, but user-friendly security, which lets us have the conveniences we're used to in centralized systems, but keeps the network secure and open to anyone interfacing with it in whatever way they please. In this project, we propose **Ethegram**, a social media platform developed by using the state-of-the-art **blockchain** technology and introduce a social networking decentralized application. The application is based on **Ethereum blockchain** platform for data records and **IPFS** for distributed data storage service. The application would reward users for quality content they contribute on the platform.

We achieve these features, including confidentiality, metadata hiding, profiles, friend networks, instant messaging, groups, and much more through a carefully constructed profile file tree distributed peer to peer over IPFS. It can be easily updated, distributed via deltas rather than bulk transfers, and hosted without being able to glean any information about that user. Offline or network-partitioned use is natural for our system, and it will handle such network difficulties gracefully—spreading data to what peers it can reach and recovering effortlessly once the partition is resolved. Most importantly of all, it lays the foundation for a secure system that people may actually want to use.

When the user creates a post, other users of the application can view and like the post if they like the content. Each like generates a single token in the application account of the creator. These tokens can be redeemed into Ethers from the platform itself. The funds are directly transferred to the account address from the contract balance. Each token when redeemed gives 0.0001 Ether. Thus, the cost of creating a new post is recovered when the user receives 10 likes on his post. More likes would ensure that the one-time registration cost is also recovered. Content creators are rewarded only based on how much their content is liked by other users of the application.

1.2 Problem Statement

As we are aware, Facebook provides decent service at the extremely high cost of constantly disregarding user privacy, common decency, and laws around the world, and yet they still have billions of users as they are the monopoly. Monopolies breed complacency, low quality, and high costs. And Facebook has a monopoly over your friend network. We can't leave because everyone is on Facebook. Switching to a new platform means every single one of your friends has to make a new account, download a new app, and you have to rebuild that whole network—if you can convince them at all. It's the same story for any platform (though most don't charge such a high price), and it's unavoidable. Hence, we believe the solution is a decentralized social network which is encrypted.

1.3 Contribution

Online social networks (OSN) are becoming more important in people's daily life, however, all popular OSNs are centralized, and this raises a series of security, privacy and management issues. A decentralized architecture based on blockchain technology provides the ability to solve above issues. In this paper, an OSN service is developed based on blockchain technology in order to make it operate decentralized. Large volume of data norm ally required low-security requirements can be stored in Interplanetary Filesystem (IPFS) to make data decentralized. A decentralized autonomous organization is developed for user autonomy, users can self-manage the OSN in a democratic way.

CHAPTER 2

LITERATURE SURVEY

Ningyuan Chen and David Siu-Yeung Cho, 2021 [8] Online social networks (OSN) are becoming more important in people's daily life, however, all popular OSNs are centralized, and this raises a series of security, privacy and management issues. A decentralized architecture based on blockchain technology provides the ability to solve above issues. In this paper, an OSN service is developed based on blockchain technology in order to make it operate decentralized. Large volume of data normally required low-security requirements can be stored in Interplanetary Filesystem (IPFS) to make data decentralized. A decentralized autonomous organization is developed for user autonomy, users can self-manage the OSN in a democratic way.

Barbara Guidi, 2021 [4] Social media is becoming one of the dominant ways to communicate. Before social media, people were extremely limited in their means to interact with others, and they were limited largely to the people that they knew in person. However, this impact on people in real life has damaged privacy. Alternative solutions have been proposed in order to overcome current social media issues. In this direction, blockchain is one of the most promising, and several blockchain-based social media have been proposed. In this paper, we analyze blockchain online social media from the technical point of view in order to understand the current trend of social DApps and to describe which characteristics are important in a blockchain-based social media scenario.

Q. Xu et. al., 2018 [10] The Internet of Things, or IoT, is the name that the IT folks have given to the now billions of physical devices throughout the world that are connected to the internet. These devices not only collect data, but they share it as well. Through the proliferation of wireless networks as well as cheap processors, it’s now possible to turn a multitude of physical objects into an IoT device. These things can range from a smart thermostat in your home that you control with your smartphone, to a sophisticated driverless car that’s filled with hundreds of sensors collecting and transmitting data back to make sure it is operating efficiently.

IPFS, 2018 [6] The InterPlanetary File System (IPFS) is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. IPFS uses content-addressing to uniquely identify each file in a global namespace connecting all computing devices. IPFS allows users to host and receive content in a manner similar to BitTorrent. As opposed to a centrally located server, IPFS is built around a decentralized system of user-operators who hold a portion of the overall data, creating a resilient system of file storage and sharing. Any user in the network can serve a file by its content address, and other peers in the network can find and request that content from any node who has it using a distributed hash table (DHT).

A. Tar, 2018 [1] A smart contract is a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. The objectives of smart contracts are the reduction of need in trusted intermediators, arbitrations and enforcement costs, fraud losses, as well as the reduction of malicious and accidental exceptions.

Antorweep Chakravorty and Chunming Rong, Ushare, 2017 [2] This work provides a systematic literature review of blockchain-based applications across multiple domains. The aim is to investigate the current state of blockchain technology and its applications and to highlight how specific characteristics of this disruptive technology can revolutionise "business-as-usual" practices. To this end, the theoretical underpinnings of numerous research papers published in high ranked scientific journals during the last decade, along with several reports from grey literature as a means of streamlining our assessment and capturing the continuously expanding blockchain domain, are included in this review. Based on a structured, systematic review and thematic content analysis of the discovered literature, we present a comprehensive classification of blockchain-enabled applications across diverse sectors such as supply chain, business, healthcare, IoT, privacy, and data management, and we establish key themes, trends and emerging areas for research. We also point to the shortcomings identified in the relevant literature, particularly limitations the blockchain technology presents and how these limitations spawn across different sectors and industries. Building on these findings, we identify various research gaps and future exploratory directions that are anticipated to be of significant value.

C. Jin et. al, [9] Multimedia Tools and Applications welcomes submissions for the new Tracks on Medical Applications of Multimedia, Biometrics and HCI, Digital Games and VR/AR, and Multimedia and Education. Multimedia Tools and Applications publishes original research articles on multimedia development and system support tools as well as case studies of multimedia applications. It also features experimental and survey articles. The journal is intended for academics, practitioners, scientists and engineers who are involved in multimedia system research, design and applications.

Buterin, 2014 [3] Being the first example of a digital asset, which simultaneously has no backing or "intrinsic value" and no centralized issuer or controller. However, another, arguably more important, part of the Bitcoin experiment is the underlying blockchain technology as a tool of distributed consensus, and attention is rapidly starting to shift to this other aspect of Bitcoin. Commonly cited alternative applications of blockchain technology include using on-blockchain digital assets to represent custom currencies and financial instruments ("colored coins"), the ownership of an underlying physical device ("smart property"), non-fungible assets such as domain names ("Namecoin"), as well as more complex applications involving having digital assets being directly controlled by a piece of code implementing arbitrary rules ("smart contracts") or even blockchain-based "decentralized autonomous organizations" (DAOs).

M. Andrychowicz et. al,2014 [7] The blockchain has fueled one of the most enthusiastic bursts of activity in applied cryptography in years, but outstanding problems in security and privacy research must be solved for blockchain technologies to go beyond the hype and reach their full potential. At the first IEEE Privacy and Security on the Blockchain Workshop (IEEE S&B), we presented peer-reviewed papers bringing together academia and industry to analyze problems ranging from deploying newer cryptographic primitives on Bitcoin to enabling usecases like privacy-preserving file storage. We overview not only the larger problems the workshop has set out to tackle, but also outstanding unsolved issues that will require further cooperation between academia and the blockchain community.

Datta. A et. al, 2011 [7] Provides current and future trends in creating intelligent social networks, and the main players and their social networks applications. Presents web-mining techniques, visualization techniques, social networks and Semantic Web, and many other topics. Includes contributions from world experts in the field of social networks from both academia and private industry. Presents standards for social networks, case studies, and a variety of applications.

CHAPTER 3

PROPOSED SYSTEM

3.1 SYSTEM ARCHITECTURE

The Smart Contract of this application will manage the entire transactions of the DApp. The user will have to register once initially with his/her wallet and this user address will be stored in a mapping in the smart contract. Each account on the application will be bound to a single account address only, in order to avoid creation of multiple accounts from same address. This smart contract will be deployed on Ethereum Virtual Machine and the frontend UI will interact with users. Hence, the entire system forms a three-level architecture where the frontend UI is responsible for user interaction while the web3 and IPFS libraries are responsible for API calls from frontend client to the backend blockchain and the IPFS storage respectively. At the top is the frontend UI, which is in charge of receiving user input for registration data, posts data and messages and passing them to the web3 library and IPFS library. IPFS library is invoked when the user creates a post with an image or video, or when the user sends a message in the chat system of the application. The image is stored on the IPFS data storage and a IPFS hash is returned which is stored on blockchain as record data. Fig 3.1 depicts the architecture diagram.

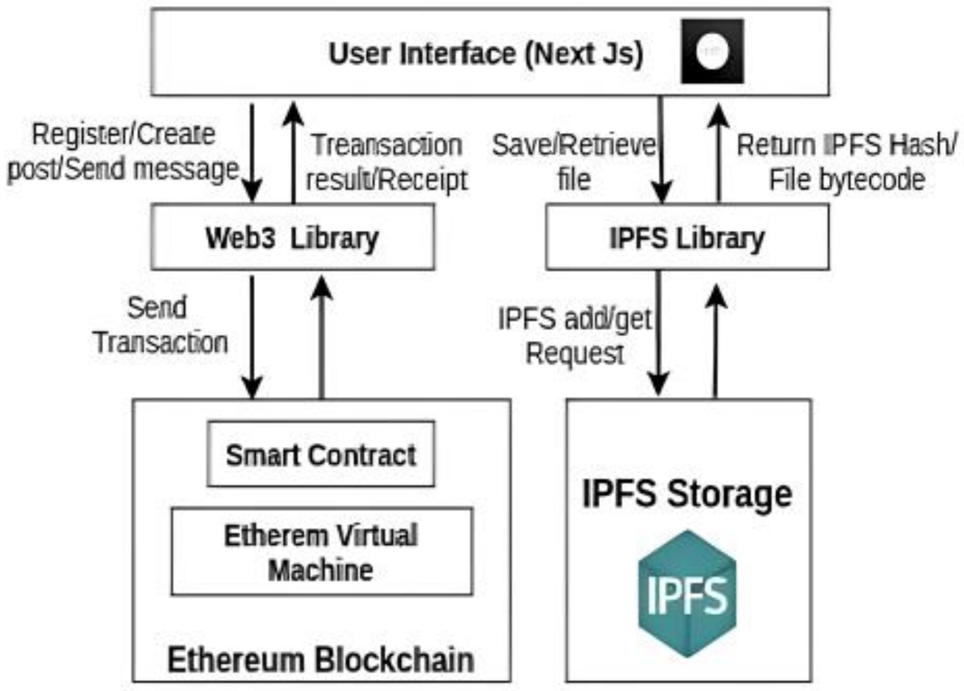


Fig 3.1 Architecture diagram

3.2 PROPOSED METHODOLOGY

The proposed methodology of Decentralized Social Media Application is composed of the following methods.

3.2.1 Process of registering

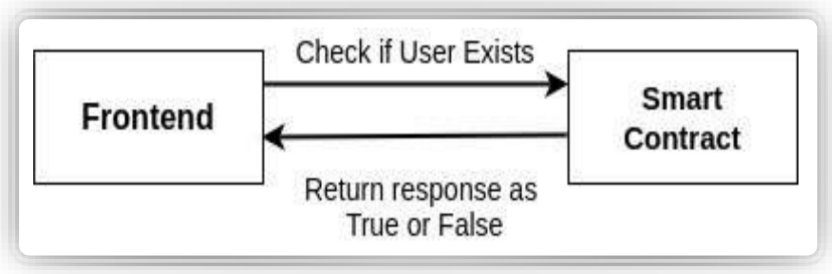


Fig 3.2.1 Process for registering

When the client starts, the user can register into the application only if he/she is not already registered. This is handled by the users mapping and frontend. If the account address of user already exists in the users mapping, the frontend registration is disabled, else the frontend sends a transaction along with 0.002 Ether (the cryptocurrency of Ethereum blockchain) to the blockchain and a new user account is registered with the application. This amount goes into the contract balance and it enables the contract to reward the users based on the likes (or upvotes) their content gets. It also discourages malicious actors to get registered on the application. This process is shown in Fig 3.2.1.

3.2.2 Process for creating post

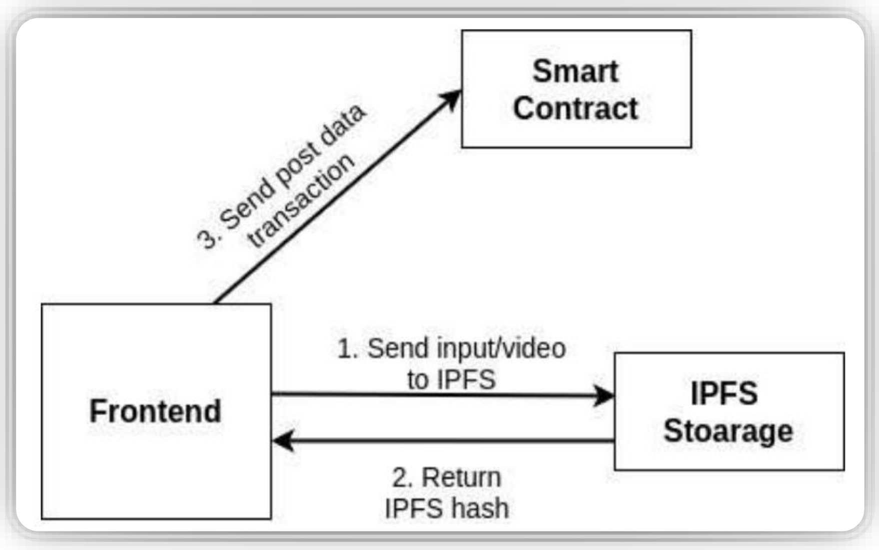


Fig 3.2.2 Process for creating post

While creating a new post, if an image or video is uploaded by the user along with the post, the frontend client loads the image or video into a byte array and sends the byte data to IPFS storage endpoint through IPFS library. When the data storage is successful, it returns the corresponding IPFS hash in return. This hash is stored on the blockchain and is later used to retrieve the image or video when required. All the post data along with creator's address, image/video IPFS hash and current timestamp is sent as a transaction to the blockchain along with an amount of 0.001 Ether to the blockchain and the post is created. Other users can now like (upvote) or comment on the post. The process of creating post is shown in Fig. 3.2.2

3.2.3 Reward System

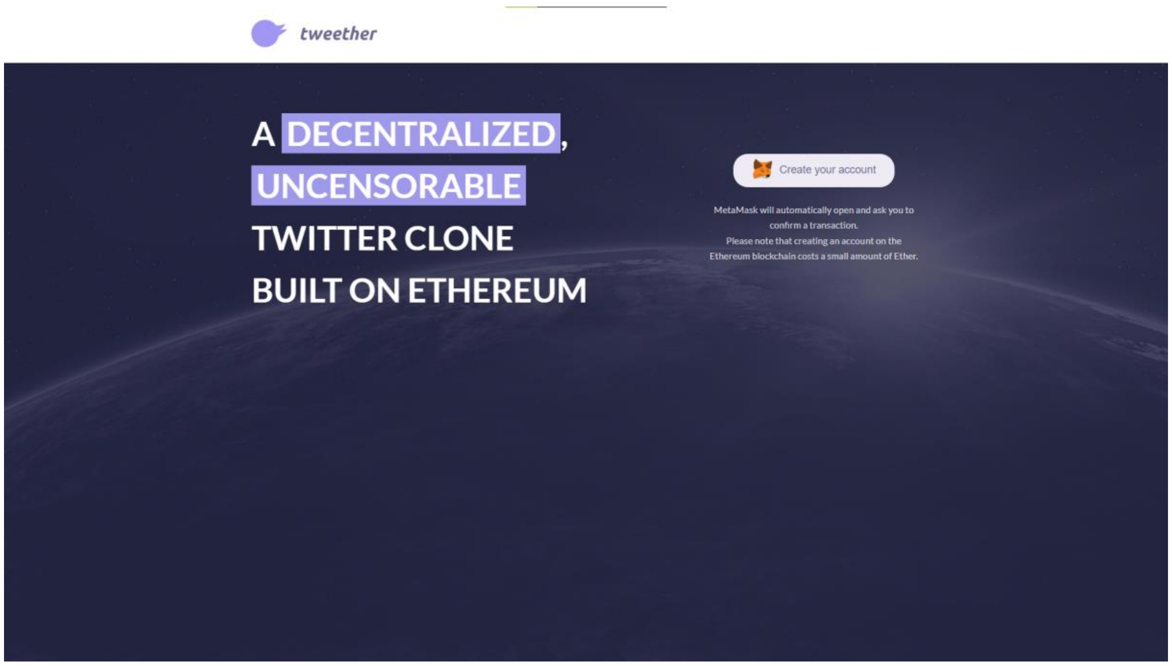
When the user creates a post, other users of the application can view and upvote the post if they like the content. Each upvote generates a single token in the application account of the creator. These tokens can be redeemed into Ethers from the platform itself. The funds are directly transferred to the account address from the contract balance. Each token when redeemed gives 0.0001 Ether. Thus, the cost of creating a new post is recovered when the user receives 10 upvotes on his post. Content creators are rewarded only based on how much their content is liked by other users of the application.

CHAPTER 4

IMPEMENTATION

4.1 Ethegram UI

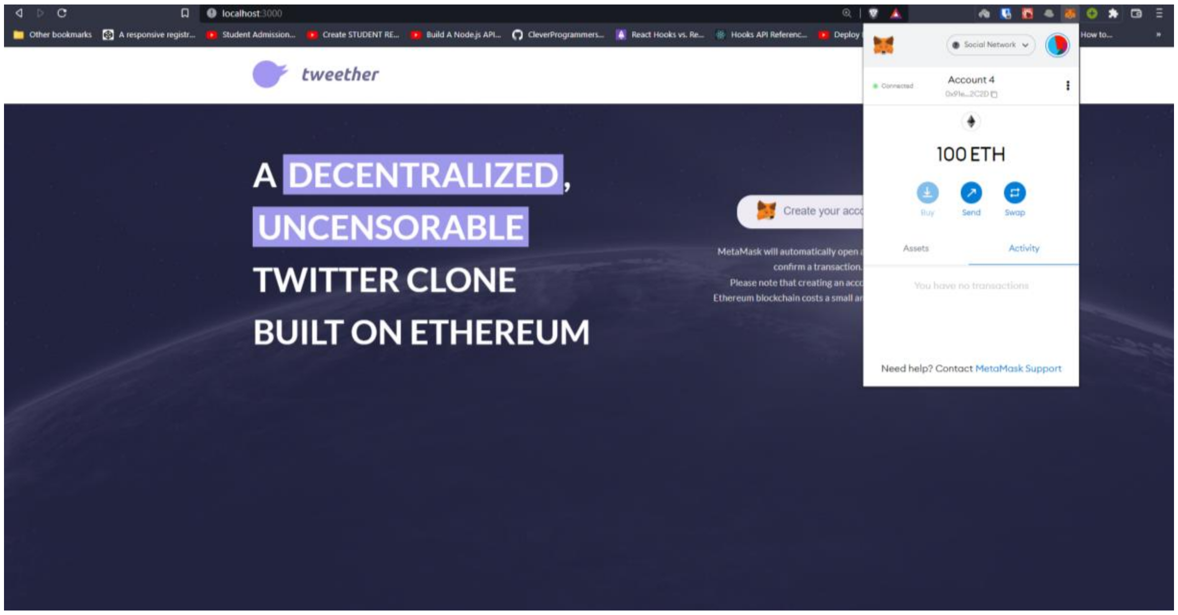
When the user opens the application, he/she is able to the home page of the application as shown in Fig 4.1.1



MetaMask will automatically open and ask you to confirm a transaction.
Please note that creating an account on the Ethereum blockchain costs a small amount of Ether.

Fig 4.1.1 – Home Page of Ethegram

The user logs in to the MetaMask account as shown in Fig 4.1.2

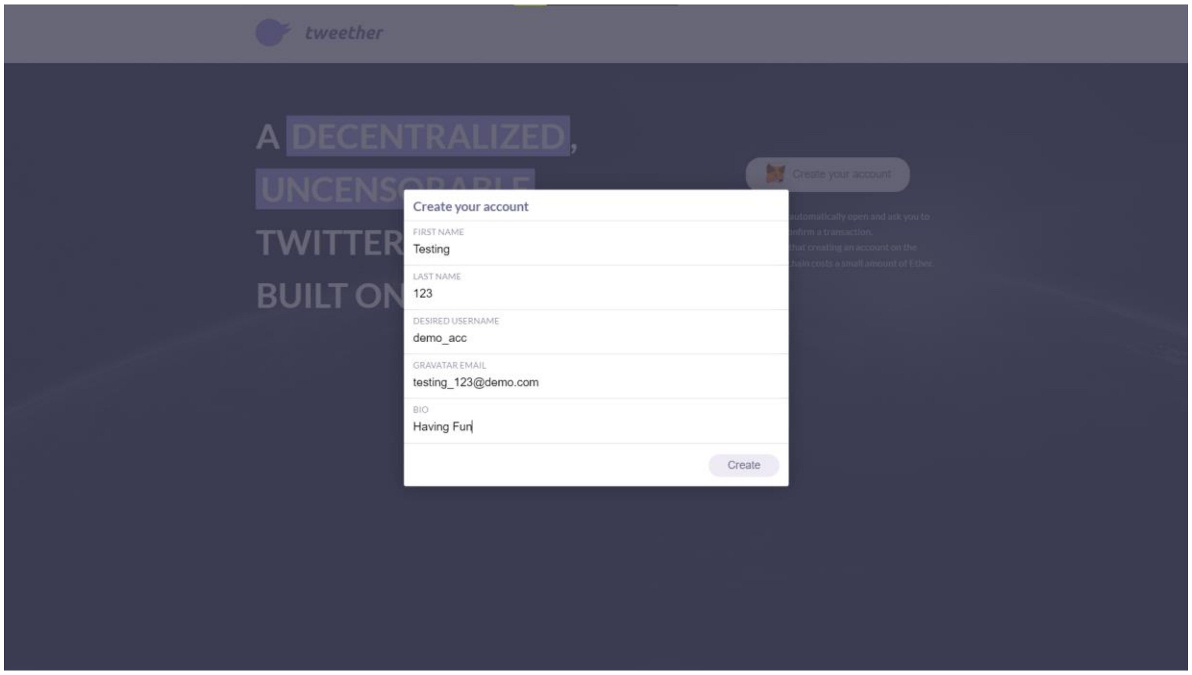


You have no transactions

Need help? Contact MetaMask Support

Fig 4.1.2 – Connecting with Metamask

Here, the user wants to register into Tweether, so the user clicks the “create your account” buuton. On clicking the button, user can fill the registration form to create an account.



Create your account

FIRST NAME
Testing

LAST NAME
123

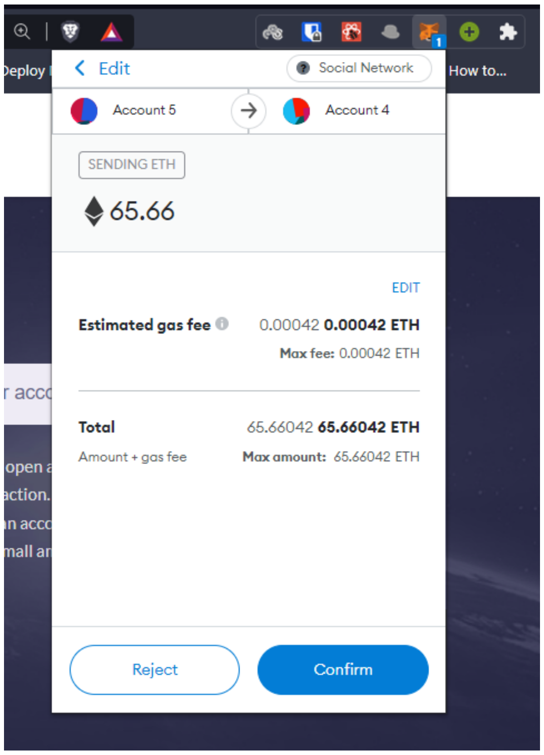
DESIRED USERNAME
demo_acc

GRAVATAR EMAIL
testing_123@demo.com

BIO
Having Fun

Fig 4.1.3 – Creating User account

Fig 4.1.4 shows the balance in the Ethereum wallet of the user’s Metamask account

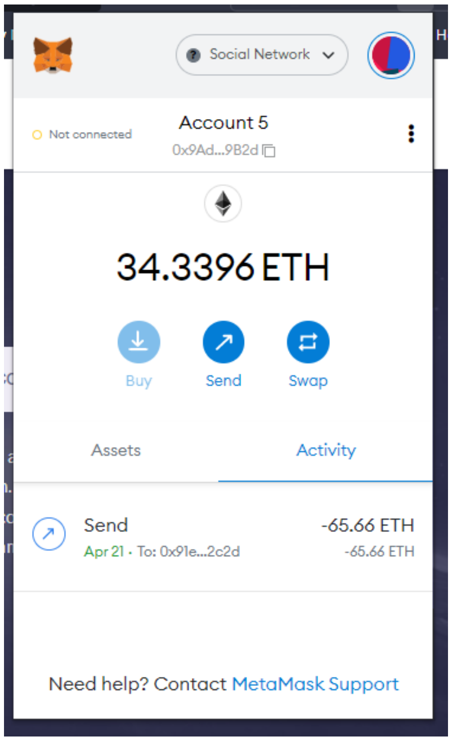


SENDING ETH
65.66

EDIT

Estimated gas fee 0.00042 **0.00042 ETH**
Max fee: 0.00042 ETH

Total 65.66042 **65.66042 ETH**
Amount + gas fee **Max amount:** 65.66042 ETH



Account 5
0x9Ad...9B2d

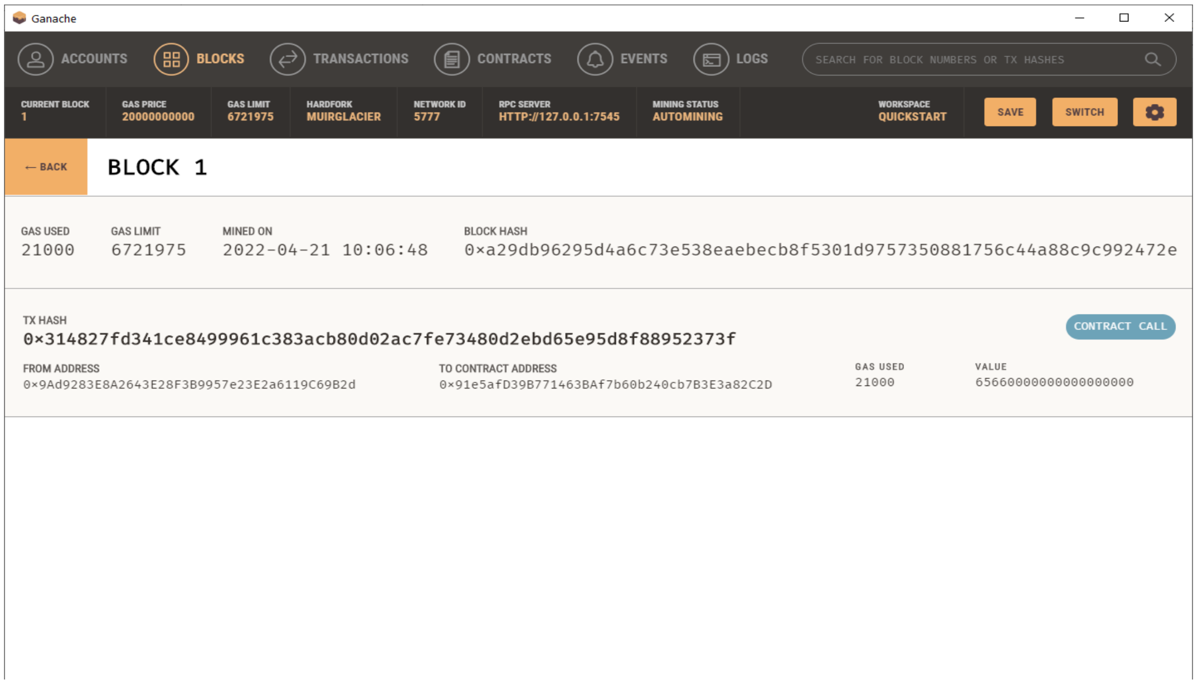
34.3396 ETH

Send -65.66 ETH
Apr 21 · To: 0x91e...2c2d -65.66 ETH

Need help? Contact MetaMask Support

Fig 4.1.4 – Ethereum Wallet using Metamask

Fig 4.1.5 shows the transactions made by the user in Ganache.



CURRENT BLOCK 1 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MUIRGLACIER | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE QUICKSTART

BLOCK 1

GAS USED	GAS LIMIT	MINED ON	BLOCK HASH
21000	6721975	2022-04-21 10:06:48	0x a29db96295d4a6c73e538eaebecb8f5301d9757350881756c44a88c9c992472e

TX HASH
0x314827fd341ce8499961c383acb80d02ac7fe73480d2ebd65e95d8f88952373f

FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0x9Ad9283E8A2643E28F3B9957e23E2a6119C69B2d	0x91e5afD39B771463BAf7b60b240cb7B3E3a82C2D	21000	65660000000000000000

Fig 4.1.5 – Setting up Ganache

4.2 Posting Tweets before IPFS

The user creates “create post” icon to post a new tweet. On clicking the icon, the user is able to post tweets in this application.

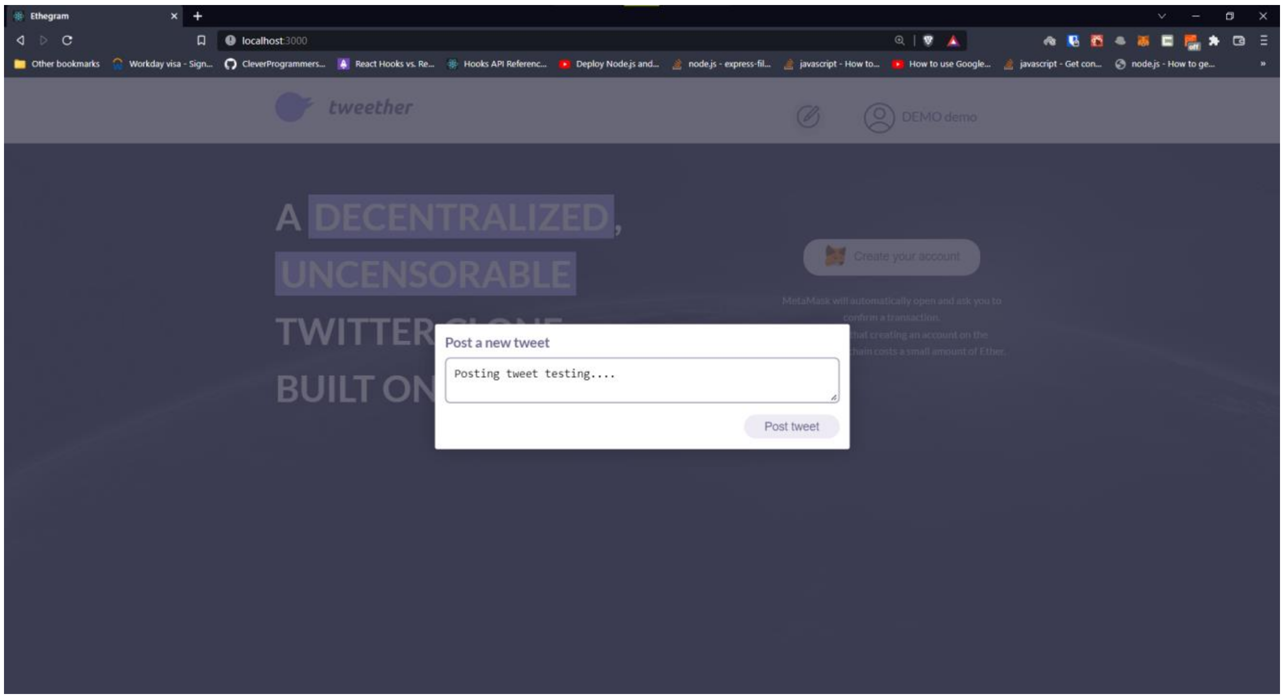


Fig 4.2.1 – Posting tweets without images

After writing the tweet the user clicks the "Post tweet" button, an Metamask windows appears to confirm the transaction to be made. By clicking confirm, the tweet gets posted.

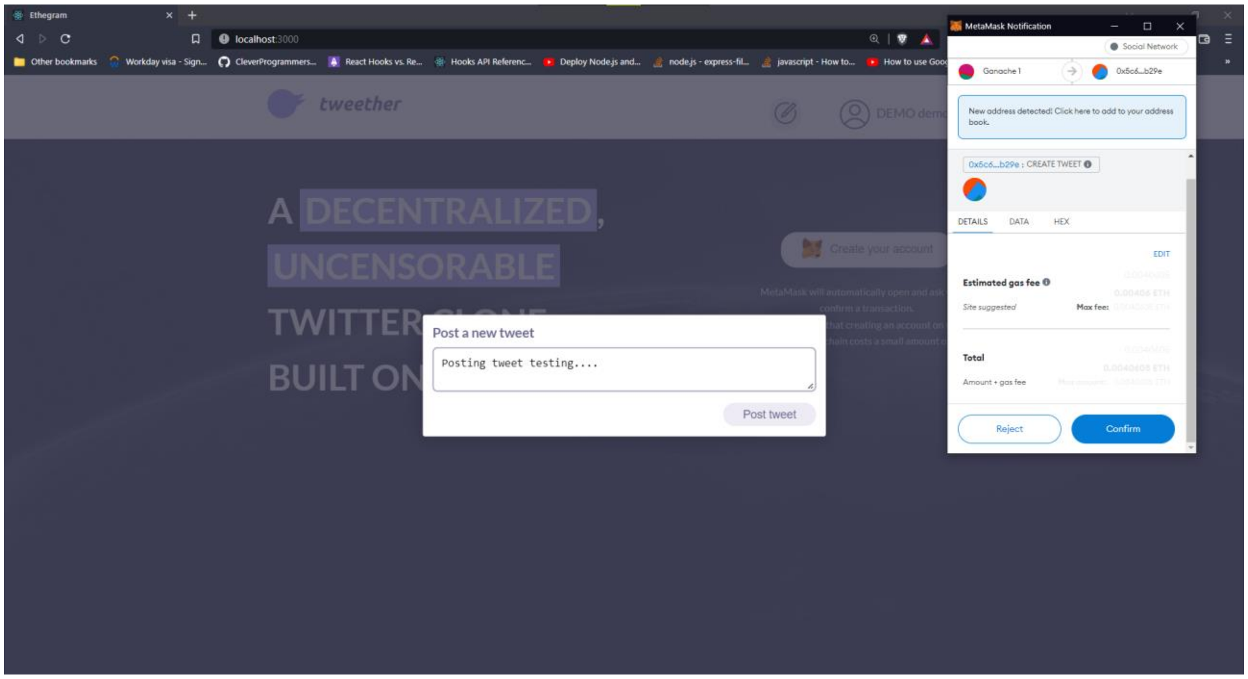

tweether
A DECENTRALIZED, UNCENSORABLE TWITTER CLONE BUILT ON
Post a new tweet
Posting tweet testing....
Post tweet
MetaMask Notification
Ganache 1
0x5c6...b29e
New address detected! Click here to add to your address book.
0x5c6...b29e : CREATE TWEET
DETAILS
DATA
HEX
Estimated gas fee
Total
Reject
Confirm

Fig 4.2.2 – Confirming Ethereum Gas usage for posting tweet

4.3 Profile Page before IPFS

User Profile page which shows the personal information of the user is shown in Fig 4.3.1.

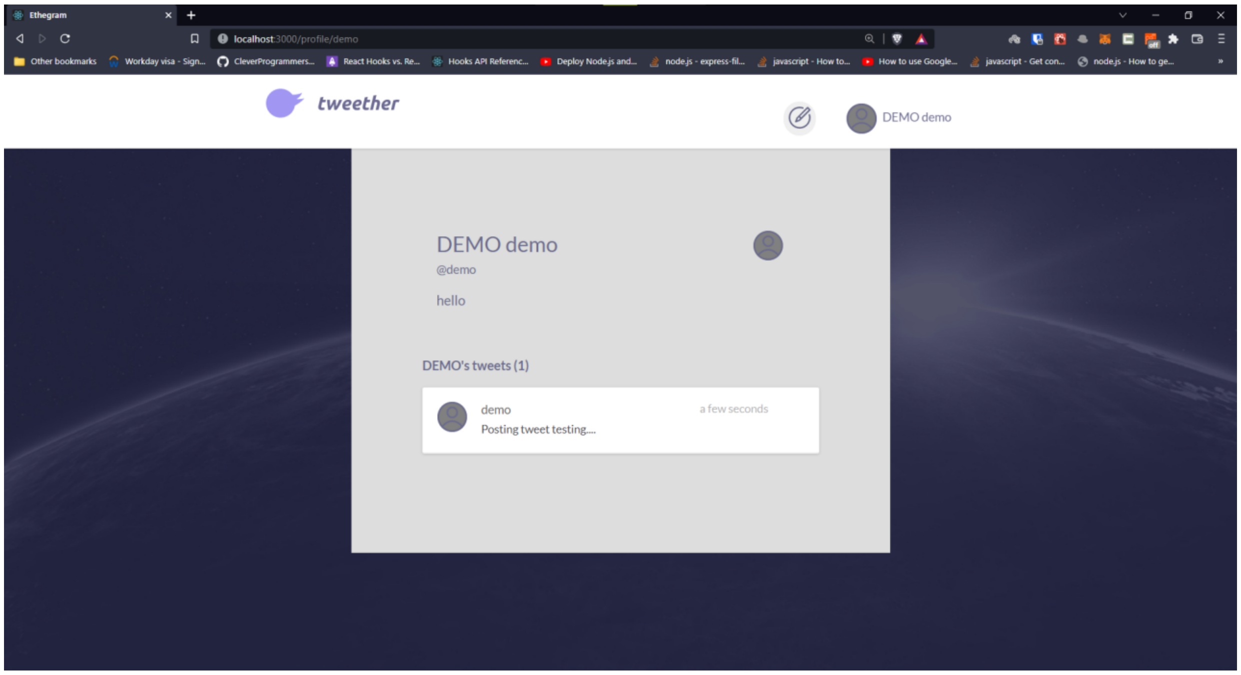
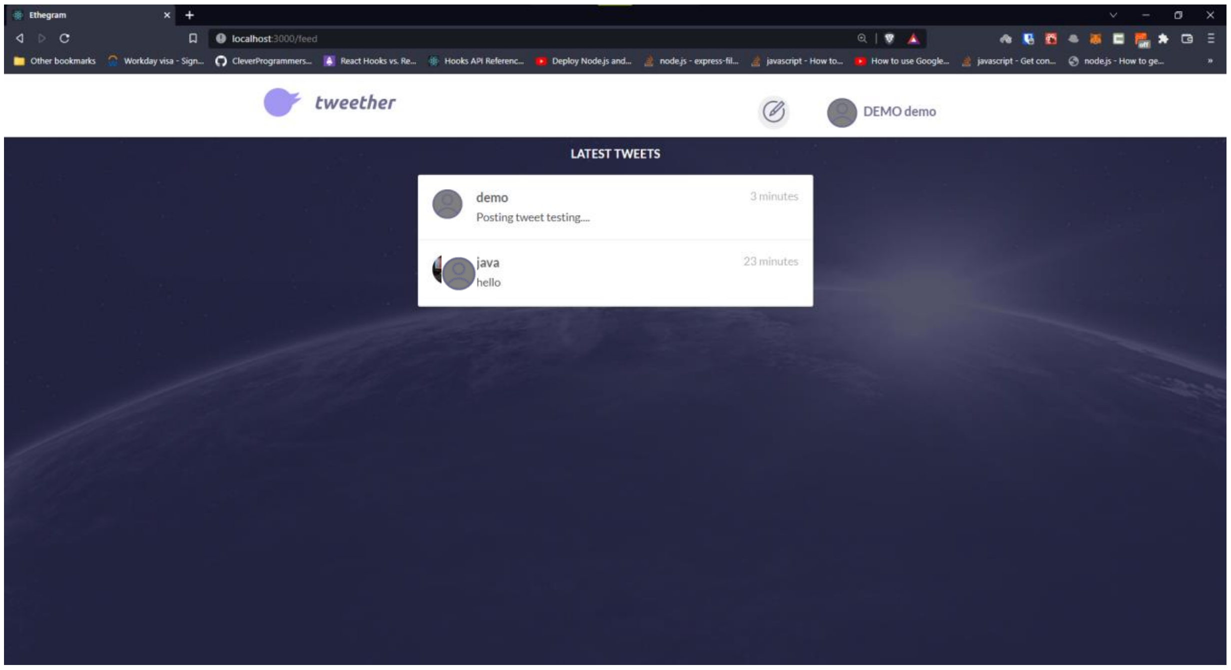


Fig 4.3.1 – Profile page of the user before IPFS

4.4 My Feed Page

The tweets posted by the other users shown in MyFeed page as shown in Fig 4.4.1.



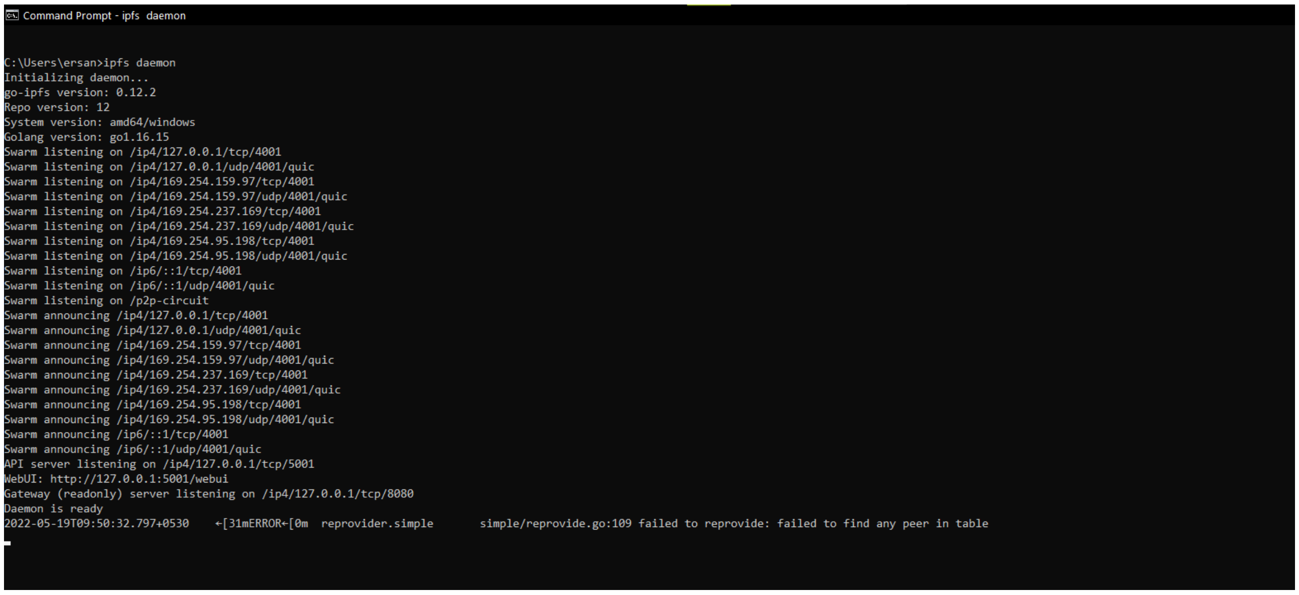
demo 3 minutes
Posting tweet testing....

java 23 minutes
hello

Fig 4.4.1 – My feed page of Ethegram application

4.5 Connecting application with IPFS

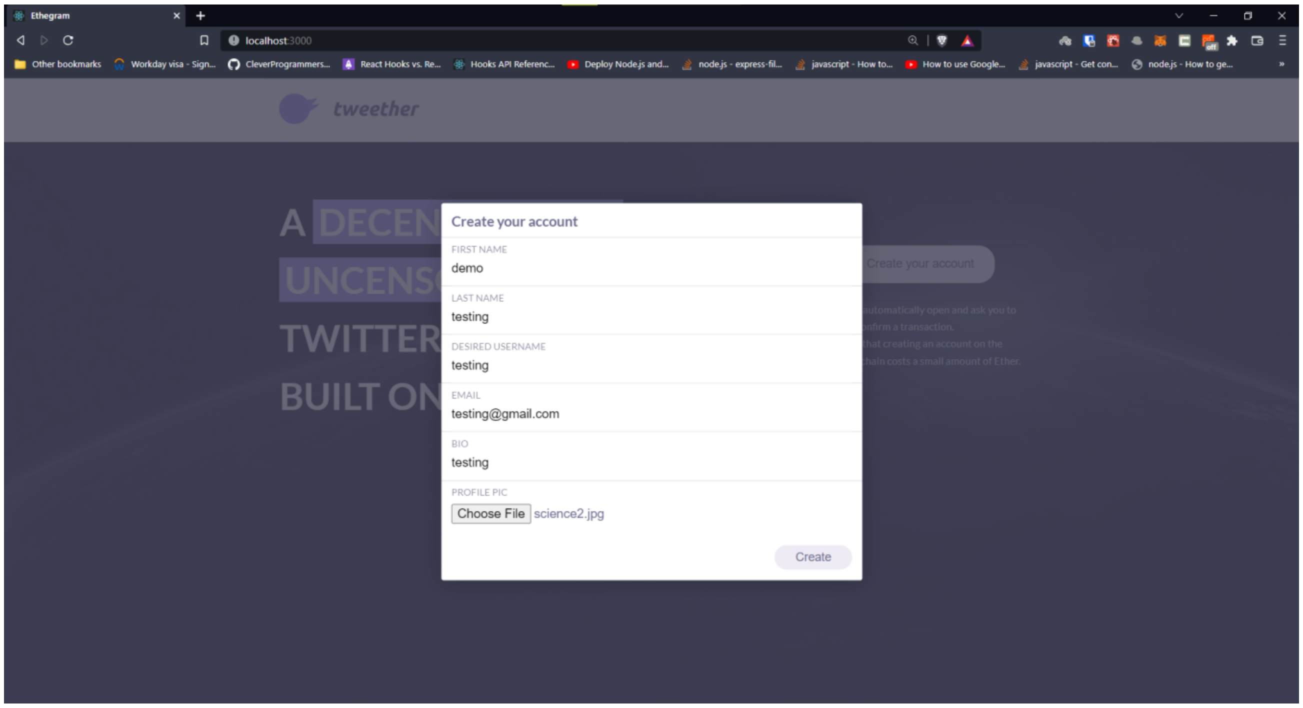
Fig 4.5.1 depicts how the connection is made between IPFS Server and Ethegram application.



```
C:\Users\ersan>ipfs daemon
Initializing daemon...
go-ipfs version: 0.12.2
Repo version: 12
System version: amd64/windows
Golang version: go1.16.15
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/127.0.0.1/udp/4001/quic
Swarm listening on /ip4/169.254.159.97/tcp/4001
Swarm listening on /ip4/169.254.159.97/udp/4001/quic
Swarm listening on /ip4/169.254.237.169/tcp/4001
Swarm listening on /ip4/169.254.237.169/udp/4001/quic
Swarm listening on /ip4/169.254.95.198/tcp/4001
Swarm listening on /ip4/169.254.95.198/udp/4001/quic
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /ip6/::1/udp/4001/quic
Swarm listening on /p2p-circuit
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Swarm announcing /ip4/169.254.159.97/tcp/4001
Swarm announcing /ip4/169.254.159.97/udp/4001/quic
Swarm announcing /ip4/169.254.237.169/tcp/4001
Swarm announcing /ip4/169.254.237.169/udp/4001/quic
Swarm announcing /ip4/169.254.95.198/tcp/4001
Swarm announcing /ip4/169.254.95.198/udp/4001/quic
Swarm announcing /ip6/::1/tcp/4001
Swarm announcing /ip6/::1/udp/4001/quic
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
2022-05-19T09:50:32.797+0530    ←[31mERROR←[0m  reprovider.simple       simple/reprovide.go:109 failed to reprovide: failed to find any peer in table
```

Fig 4.5.1 – Connecting Ethegram application with IPFS

Here, the user can create an account by uploading his/her image with the help of IPFS as shown in Fig 4.5.2.


Create your account

FIRST NAME
demo

LAST NAME
testing

DESIRED USERNAME
testing

EMAIL
testing@gmail.com

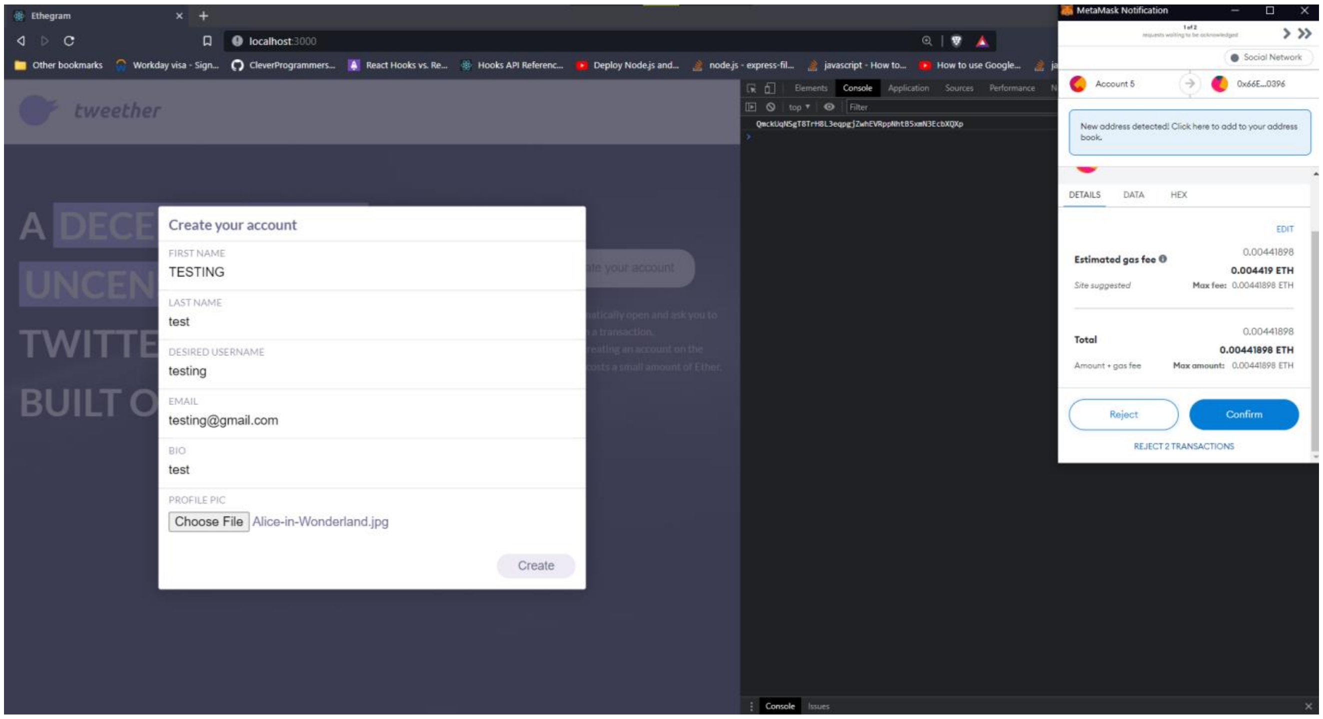
BIO
testing

PROFILE PIC
Choose File science2.jpg

Create

Fig 4.5.2 – Choosing profile pic after IPFS connection

In Fig 4.5.3, an IPFS file hash is created for the uploaded image and by clicking confirm the transaction is made. Finally, a user with a profile image can be created.


Create your account

FIRST NAME
TESTING

LAST NAME
test

DESIRED USERNAME
testing

EMAIL
testing@gmail.com

BIO
test

PROFILE PIC
Choose File Alice-in-Wonderland.jpg

Create

```
QmckUqNSgT8TrH8L3eqpgjZwhEVRppNhtB5xmN3EcbXQXp
```

MetaMask Notification

1 of 2

Social Network

Account 5 → 0x66E...0396

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

DETAILS DATA HEX

EDIT

Estimated gas fee 0.00441898 **0.004419 ETH**
Site suggested Max fee: 0.00441898 ETH

Total 0.00441898 **0.00441898 ETH**
Amount + gas fee Max amount: 0.00441898 ETH

Reject Confirm

REJECT 2 TRANSACTIONS

Fig 4.5.3 – IPFS file hash

The uploaded image using IPFS hash is shown in Fig 4.5.4.

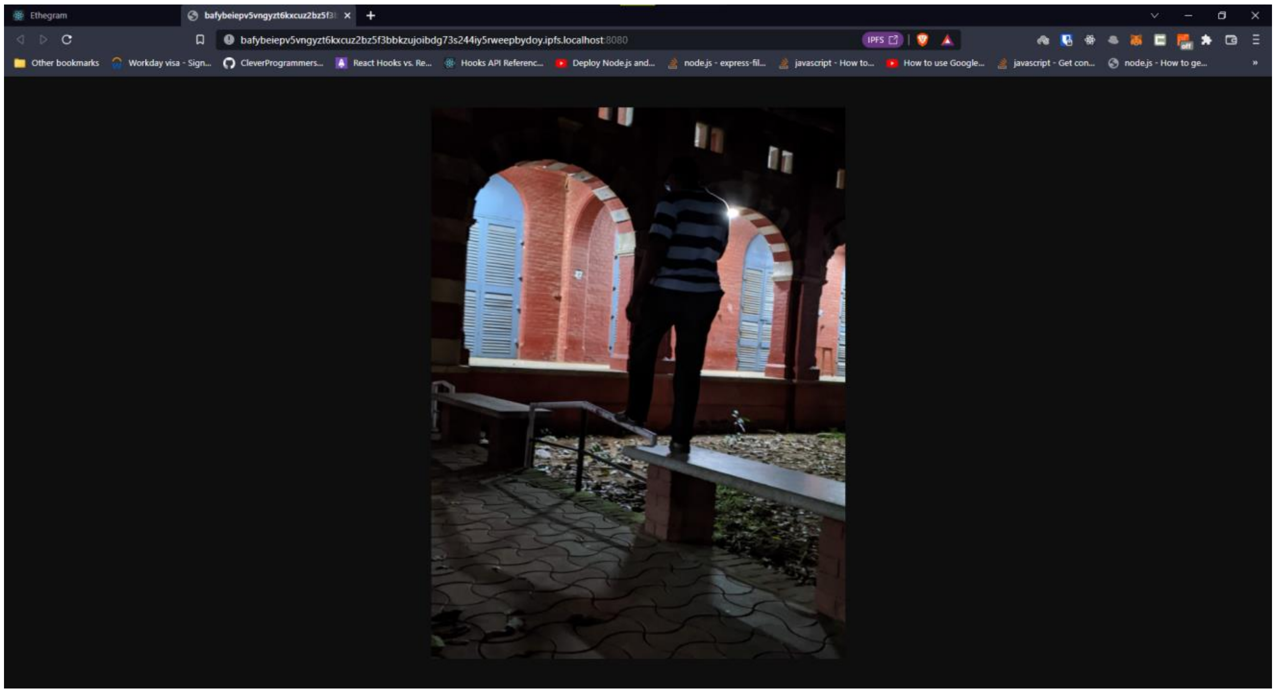
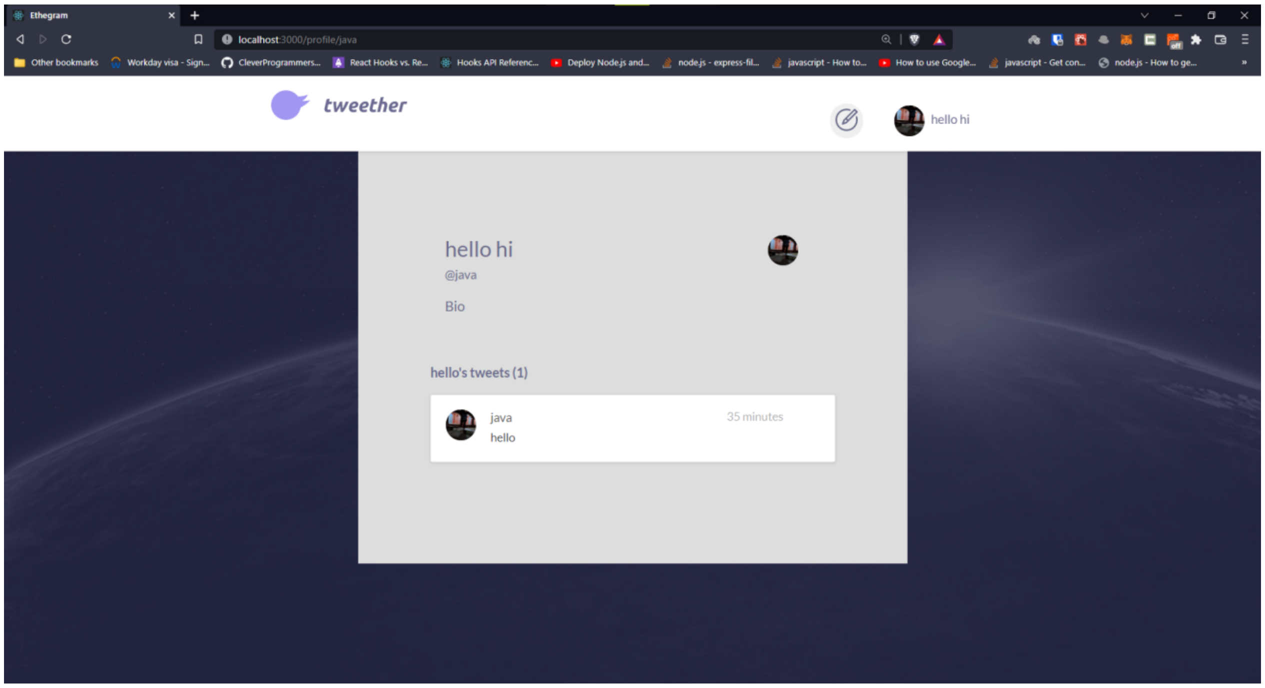


Fig 4.5.4 – Uploaded image using IPFS hash

The profile page which contains user information and tweets posted by the user with the profile image is shown in Fig 4.5.5.



hello hi
@java
Bio

hello's tweets (1)

java — 35 minutes
hello

Fig 4.5.5 – Profile page with uploaded image

Although there is a field to upload image while creating tweets, they can also be created without uploading any images.

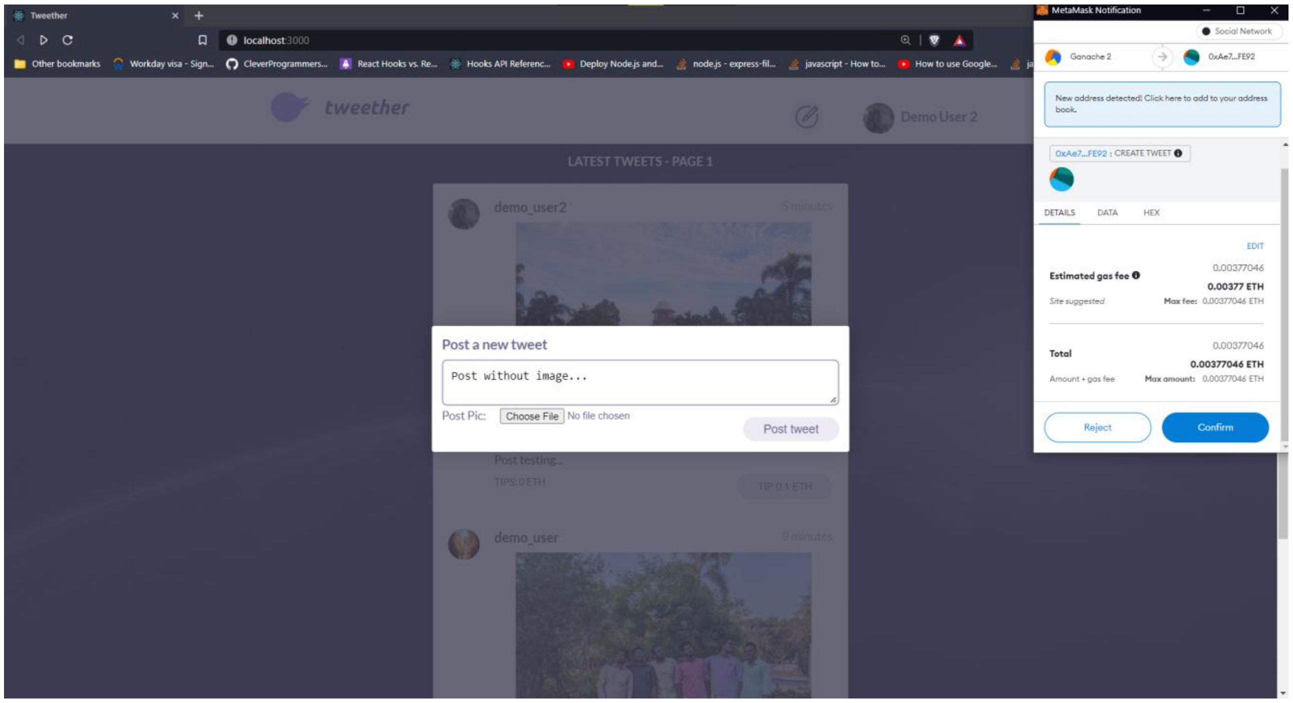


Fig 4.5.6 – Posting tweet without image

Fig 4.5.7 shows the MyFeed page after creating tweets with and without uploading any image.

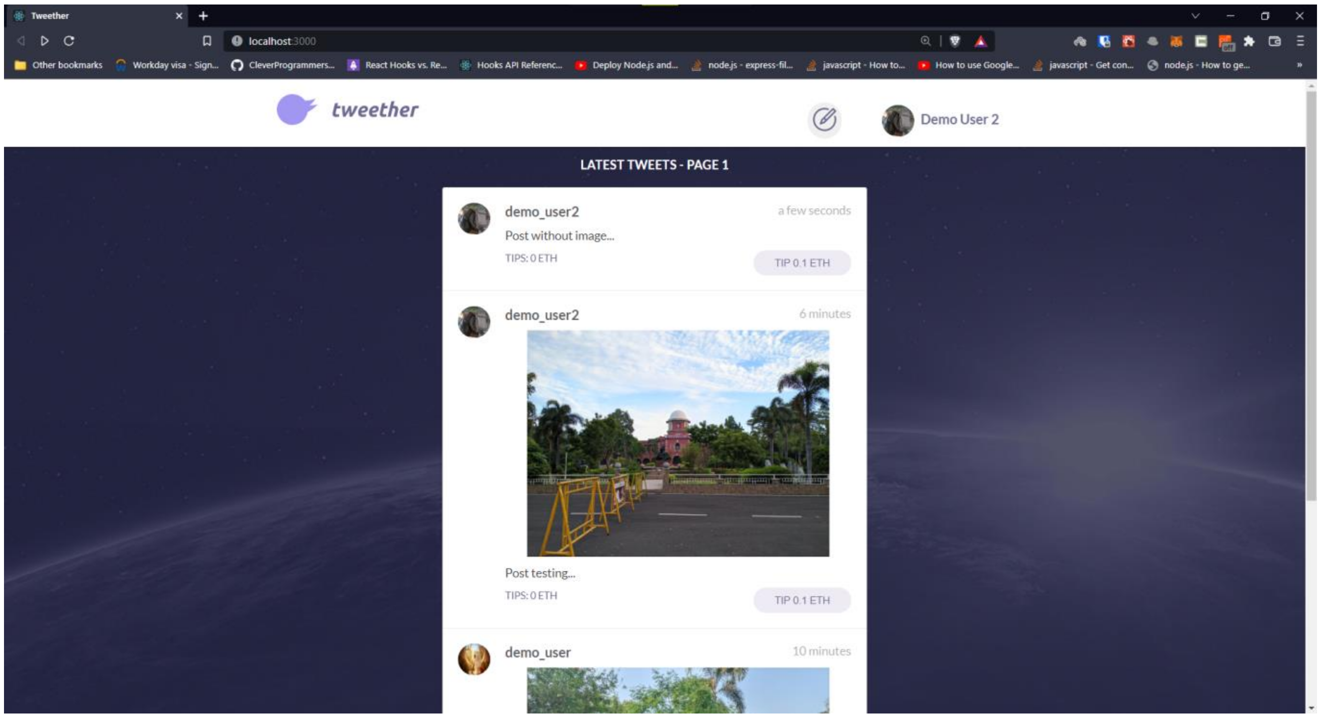


Fig 4.5.7 – After posting tweet without image

The user can upload an image while posting tweets. By clicking confirm button the transaction is being approved and the tweet gets posted with the image.

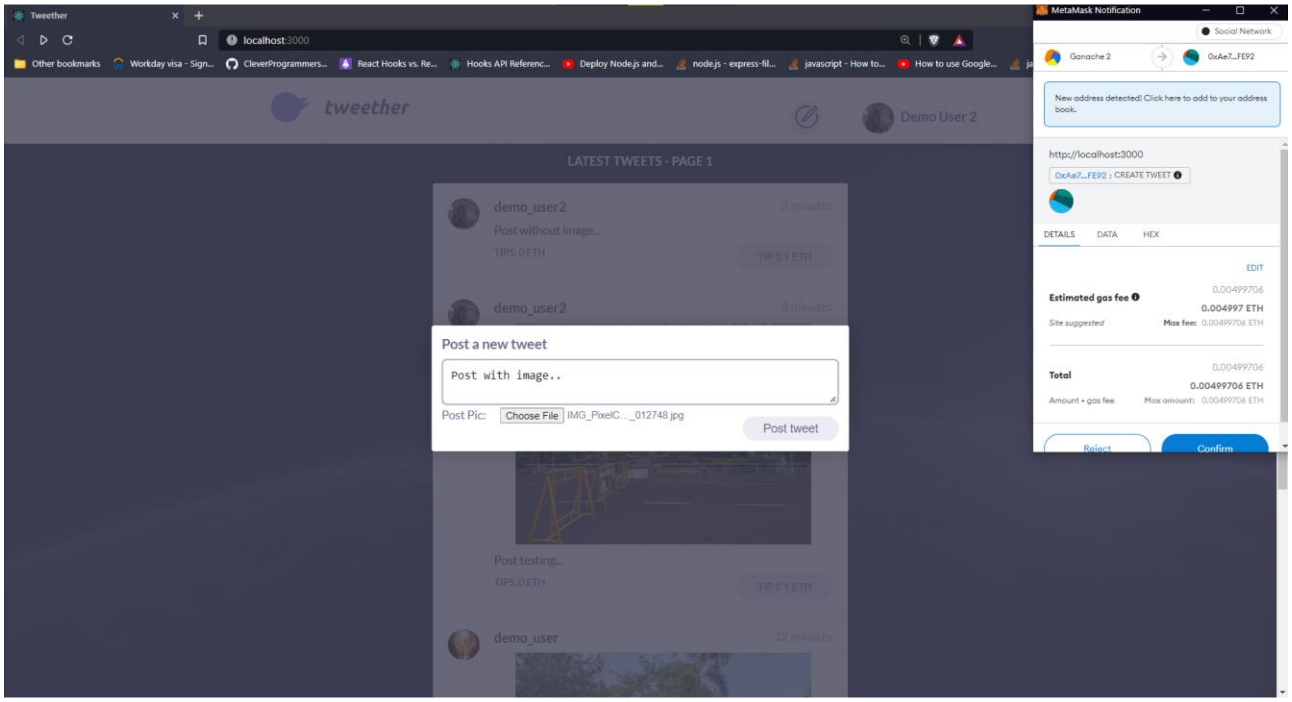


Fig 4.5.8 – Posting tweet with image

Fig 4.5.9 shows the MyFeed page after posting a tweet with an image

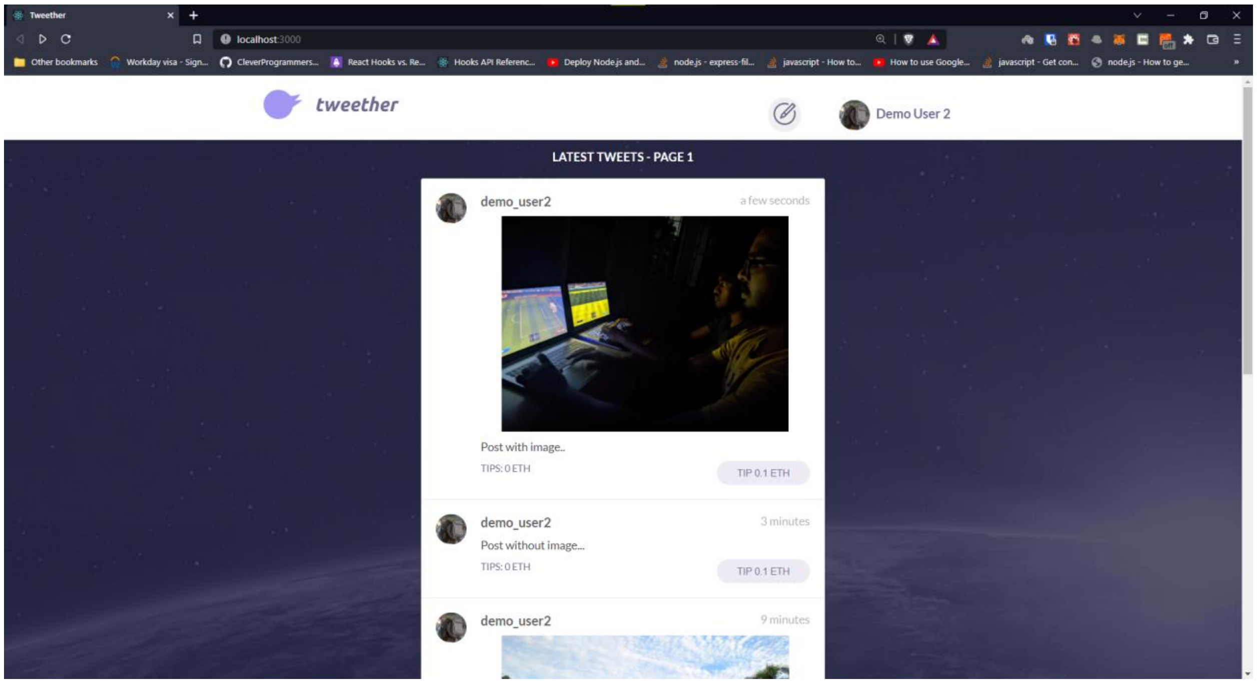


Fig 4.5.9 – After posting tweet with image

4.6 Reward System

NOTE : INITIAL BALANCE of demo user (Ganache 1) and demo user2 (Ganache 2)

The user can tip 0.1 ethers to other user by clicking the button given below the tweet. Fig 4.6.1 shows the balance of the Ethereum wallet of the user before and after tipping a tweet.

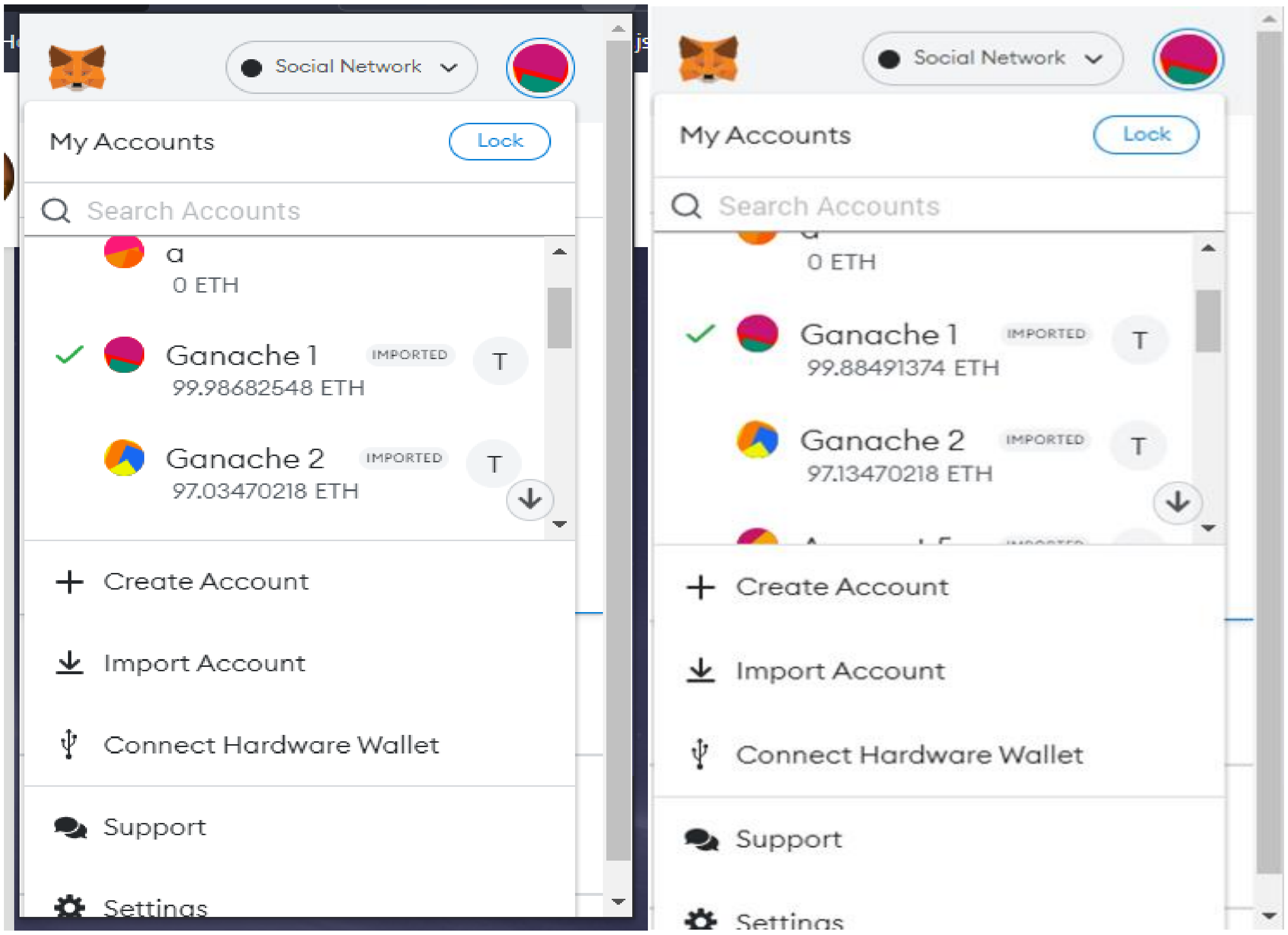


Fig 4.6.1 Before tipping After tipping

(Demo user TIPS 01.ETH TO POST CREATED BY demo user2)

CHAPTER 5

CONCLUSION

As a result, a decentralised social network has been devised and implemented, which answers the questions highlighted previously. As a result, only the users control the content they upload in our Tweether application. Everyone's data is just 'out there,' in encrypted blobs that anybody may host or download but that only friends can decipher. Decentralization also makes it resistant to censorship, internet disruptions, and potential social monopolies. The decentralised social network paradigm has not only security, which is not difficult with public key cryptography, but also user-friendly security, which allows us to have the conveniences we've come to expect from centralised systems while keeping the network secure and open to anyone who wants to interact with it in any way they want.

5.1 Future Works

Future work needs to be done in order to develop the same as a mobile application without compromising on any of the aspects of web application. The smart contract of the application can also be improved since Solidity, the language of Ethereum Smart Contracts, currently does not support passing of complex data structures like multi-dimensional arrays.

APPENDIX

1. STORAGE CONTARCTS

Tweet Storage :

```
pragma solidity >=0.4.22 <0.9.0;

import "../helpers/BaseStorage.sol";

contract TweetStorage is BaseStorage {
    mapping(uint256 => Tweet) public tweets;
    mapping(uint256 => uint256[]) public userTweetIds;
    uint256[] public tweetIds;

    struct Tweet {
        uint256 id;
        string text;
        uint256 userId;
        uint256 postedAt;
        string postHash;
        uint256 tipAmount;
    }

    uint256 latestTweetId = 0;

    function createTweet(
        uint256 _userId,
        string memory _text,
        string memory postHash
    ) public onlyController returns (uint256) {
        latestTweetId++;

        tweets[latestTweetId] = Tweet(
            latestTweetId,
            _text,
            _userId,
            block.timestamp,
            postHash,
            0
        );
        userTweetIds[_userId].push(latestTweetId);
        tweetIds.push(latestTweetId);

        return latestTweetId;
    }

    function getTweetIdsFromUser(uint256 _userId)
        public
        view
        returns (uint256[] memory)
    {
        return userTweetIds[_userId];
    }

    function getNumTweets() public view returns (uint256) {
        return tweetIds.length;
    }

    function tipPost(uint256 _id, address _authorAddr)
        public
        payable
        onlyController
    {
        address payable _author = payable(_authorAddr);
        _author.transfer(msg.value);

        tweets[_id].tipAmount = tweets[_id].tipAmount + msg.value;
    }
}
```

User Storage :

```
pragma solidity >=0.4.22 <0.9.0;

import "../helpers/BaseStorage.sol";

contract UserStorage is BaseStorage {
    mapping(uint256 => Profile) public profiles;
    mapping(address => uint256) public addresses;
    mapping(bytes32 => uint256) public usernames;
    mapping(uint256 => address) public uidtoaddr;

    struct Profile {
        uint256 id;
        bytes32 username;
        bytes32 firstName;
        bytes32 lastName;
        string bio;
        string gravatarEmail;
        string profileHash;
    }

    uint256 latestUserId = 0;

    function createUser(
        address _address,
        bytes32 _username,
        bytes32 _firstName,
        bytes32 _lastName,
        string memory _bio,
        string memory _gravatarEmail,
        string memory _profileHash
    ) public onlyController returns (uint256) {
        latestUserId++;

        profiles[latestUserId] = Profile(
            latestUserId,
            _username,
            _firstName,
            _lastName,
            _bio,
            _gravatarEmail,
            _profileHash
        );

        addresses[_address] = latestUserId;
        usernames[_username] = latestUserId;
        uidtoaddr[latestUserId] = _address;

        return latestUserId;
    }
}
```

2. CONTROLLER CONTRACTS

Tweet Controller :

```
pragma solidity >=0.4.22 <0.9.0;

import "../helpers/BaseController.sol";
import "../ContractManager.sol";
import "./TweetStorage.sol";
import "../users/UserStorage.sol";

contract TweetController is BaseController {
    function createTweet(string memory _text, string memory _postHash)
        public
        returns (uint256)
    {
        ContractManager _manager = ContractManager(managerAddr);

        address _userStorageAddr = _manager.getAddress("UserStorage");
        UserStorage _userStorage = UserStorage(_userStorageAddr);
        uint256 _userId = _userStorage.addresses(msg.sender);

        require(_userId != 0);

        address _tweetStorageAddr = _manager.getAddress("TweetStorage");
        TweetStorage _tweetStorage = TweetStorage(_tweetStorageAddr);

        return _tweetStorage.createTweet(_userId, _text, _postHash);
    }

    function tipPost(uint256 _id) public payable {
        ContractManager _manager = ContractManager(managerAddr);

        address _tweetStorageAddr = _manager.getAddress("TweetStorage");
        TweetStorage _tweetStorage = TweetStorage(_tweetStorageAddr);

        require(_id > 0 && _id <= _tweetStorage.getNumTweets());

        (
            uint256 id,
            string memory text,
            uint256 userId,
            uint256 postedAt,
            string memory postHash,
            uint256 tipAmount
        ) = _tweetStorage.tweets(_id);

        address _userStorageAddr = _manager.getAddress("UserStorage");
        UserStorage _userStorage = UserStorage(_userStorageAddr);

        require(
            _userStorage.addresses(msg.sender) != 0 &&
                _userStorage.uidtoaddr(userId) != address(0)
        );

        _tweetStorage.tipPost{value: msg.value}(
            _id,
            _userStorage.uidtoaddr(userId)
        );
    }
}
```

User Controller :

```
pragma solidity >=0.4.22 <0.9.0;

import "../helpers/BaseController.sol";
import "../ContractManager.sol";
import "./UserStorage.sol";

contract UserController is BaseController {
    function createUser(
        bytes32 _username,
        bytes32 _firstName,
        bytes32 _lastName,
        string memory _bio,
        string memory _gravatarEmail,
        string memory _profileHash
    ) public returns (uint256) {
        ContractManager _manager = ContractManager(managerAddr);

        address _userStorageAddr = _manager.getAddress("UserStorage");
        UserStorage _storage = UserStorage(_userStorageAddr);

        require(_storage.addresses(msg.sender) == 0);
        require(_storage.usernames(_username) == 0);

        return
            _storage.createUser(
                msg.sender,
                _username,
                _firstName,
                _lastName,
                _bio,
                _gravatarEmail,
                _profileHash
            );
    }
}
```

3. IPFS

Connecting to IPFS :

```
const { create } = require("ipfs-http-client");
const ipfs = create("http://localhost:5001");

export default ipfs;
```

Sending File Buffer to IPFS :

```
const [formState, setForm] = useState({
    firstName: "",
    lastName: "",
    username: "",
    gravatarEmail: "",
    bio: "",
    profile: null,
  });

  const updateField = (fieldName, e) => {
    e.preventDefault();

    if (fieldName === "profile") {
      const file = e.target.files[0];
      if (file) {
        const reader = new window.FileReader();
        reader.readAsArrayBuffer(file);
        reader.onloadend = () => {
          formState[fieldName] = Buffer(reader.result);
          setForm(formState);
        };
      } else {
        formState[fieldName] = null;
        setForm(formState);
      }
    } else {
      formState[fieldName] = e.target.value;
      setForm(formState);
    }
  };

  const createuser = async (e) => {
    e.preventDefault();
    var hash = "";

    if (formState["profile"]) {
      try {
        const uploadResult = await ipfs.add(Buffer.from(formState["profile"]));
        console.log(uploadResult.path);
        hash = uploadResult.path;
      } catch (e) {
        return alert(e);
      }
    }

    for (let key in formState) {
      if (key !== "profile") {
        if (!formState[key]) {
          return alert(`You must fill in your ${key}!`);
        }
      }
    }

    const { firstName, lastName, username, bio, gravatarEmail } = formState;

    try {
      const res = await createUser(
        username,
        firstName,
        lastName,
        bio,
        gravatarEmail,
        hash
      );

      if (res.tx !== undefined) {
        alert(`Your user has been created!`);
        window.location.reload();
      } else throw res.message;
    } catch (err) {
      alert(`Sorry, we couldn't create your user: ${err}`);
    }

    onClose();
  };
```

REFERENCES

1. A. Tar. (July 26 2018) Smart contracts, explained. [Online]. Available: https://cointelegraph.com/ explained/smart-contracts-explained.
2. Antorweep Chakravorty and Chunming Rong, “Ushare: user controlled social media based on blockchain”, Conference Paper – January 2017
3. Buterin, Ethereum White Paper. 2014, p.23.
4. Barbara Guidi, An Overview of Blockchain Online Social Media from the Technical Point of View, 2021
5. Datta, A, Buchegger, S, Vu, L.H, Strufe, T, Rzadca, K. In Handbook of Social Network Technologies and Applications; Springer
6. IPFS. [Online]. Available: https://ipsfs.io/, (2018)
7. M. Andrychowicz, S. Dziembowski, D. Malinowski,and L. Mazurek. In 2014 IEEE Symposium on Security and Privacy, pages 443–458. IEEE, 2014
8. Ningyuan Chen, David Siu-Yeung Cho, A Blockchain based Autonomous Decentralized Online Social Network, 2021
9. Q. Xu, C. Jin, M. F. B. M. Rasid, B. Veeravalli, and K. M. M. Aung, "Blockchain-based decentralized content trust for docker images", Multimedia Tools and Applications, pp. 1–26, 2017.
10. Q. Xu, K. M. M. Aung, Y. Zhu, and K. L. Yong, “A blockchain-based storage system for data analytics in the internet of things,” in New Advances in the Internet of Things. Springer, 2018, pp. 119–138.