EQUIFY

Introduction to Blockchain

Submitted by:

Amit G. Patil (19104004) B11

Sanjoli Goyal (19104007) B11

Muskan Jain(19104010)B11

Under the supervision of:

Dr. Naveen Kumar Gupta

Department of CSE/IT

Jaypee Institute of Information Technology University, Noida

MAY, 2022

Table Of Contents

Abstract :

At present, Crowdfunding source of raising funds typically for startups or projects has gained popularity with most startups resorting to the use of Crowdfunding platforms to raise funds in exchange of equity because it is relatively inexpensive and uncomplicated in nature. In the existing model, Pool of people contribute small amounts of money towards a project or cause and expect some financial returns. The call for a solution to issues related to security, investor abuse and illegal transactions that could plague crowdfunding has led me to investigate the implications of blockchain in Crowdfunding. So we are proposing an ethereum based crowdfunding application Equify.

An equify crowdfunding app managed with smart contracts that allows users to invest in projects with crypto in return of equity.

What makes equify truly unique is its decentralized and autonomous approach towards crowdfunding using smart contracts that are deployed on the Ethereum blockchain and a mobile app created using Google's Flutter which can be used on both Android and IOS.

Startups on the platform can create camps in which normal users can purchase equity by investing CTV (Collective token) , a ERC20 fungible token exclusive only to the Equify platform. This platform exclusive ERC20 token along with smart contracts enables Equify to tackle the issue of trust and security that plagues all the existing crowdfunding platforms.

What are the benefits of using smart contracts for crowdfunding

Different crowdfunding platforms describe various advantages of using smart contracts for crowdfunding processes optimization. The three major things that every platform highlights are:

- Higher speed of transactions or identity verification checks
- Better liquidity for previously illiquid assets via asset tokenization
- Enhanced security – which apparently comes from a larger "secure blockchain" concept
- Faster funds disbursement – however at the cost of skipping the verification, according to this white-paper

Scope of the project

Functional requirements :

1. **Registration:** Users can register using email and password.
2. **Login:** User Login his account.
3. **Home page:** Users can visit his home page.
4. **User Detail Screen:** Users can view his information like CTV Balance,username,Email-Id,Camps Owned,etc..
5. **Search Camps Screen:** Users can search the camps on this screen.

Non functional requirements

1. Portability
Applications could be accessed by mobile or desktop devices.
2. Security
Email password authentication is being used as well as database security using tokens will be maintained.
3. Scalability
Backend is restful so that it could be easily scaled using PM2 servers. Smart Contracts are fundamentally scalable.
4. Reusability
DRY(Do not repeat yourself) principle is the main thought kept while developing the application and multiple component based applications architecture is used
5. Flexibility
This project can be a stepping up project for further advances and always leaves a room for improvement and extension of functionalities.

Description of the modules of the project.

1. Client
/public/index.html - frontend html entry point of the application
/src - Pizza app website components and react state logic is mentioned here

2. Models
Contains four models of Camp details, collab model , user authentication model and user details model.

3. Routes
Backend Node.js routes namely for authentication and blockchain related queries

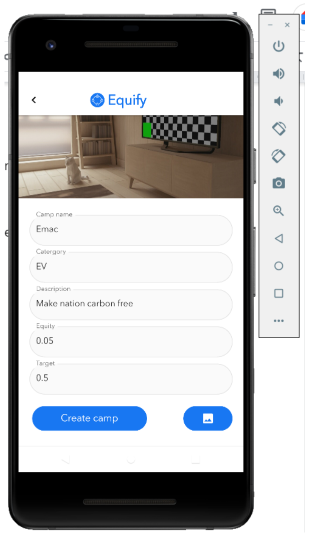
4. server.js
Starting point of nodejs application

5. .env
Environment variable files which store mnemonics, token and camp contract address with support email and passwords

6. package .json
Contains script and npm modules are mentioned here
7. Truffle-config.js
All truffle related configurations are stored in this file

User Interface of the project

Add new camp Details



Equify

Camp name: Emac

Catergory: EV

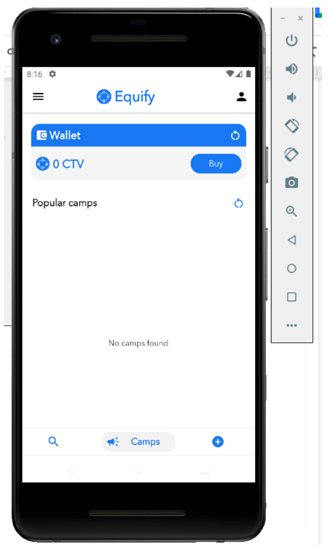
Description: Make nation carbon free

Equity: 0.05

Target: 0.5

Create camp

Home Screen



8:16

Equify

Wallet

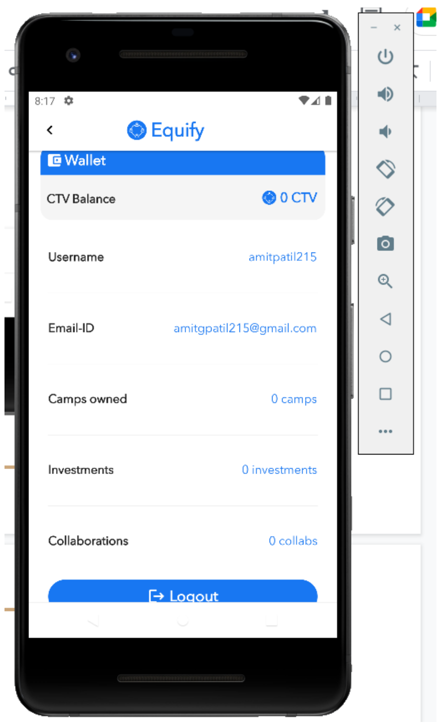
0 CTV Buy

Popular camps

No camps found

Camps

User Details Screen



8:17

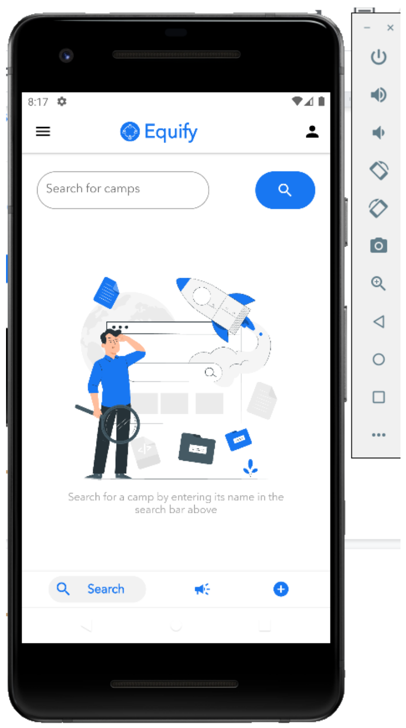
Equify

Wallet

CTV Balance	0 CTV
Username	amitpatil215
Email-ID	amitgpatil215@gmail.com
Camps owned	0 camps
Investments	0 investments
Collaborations	0 collabs

Logout

Search Camps



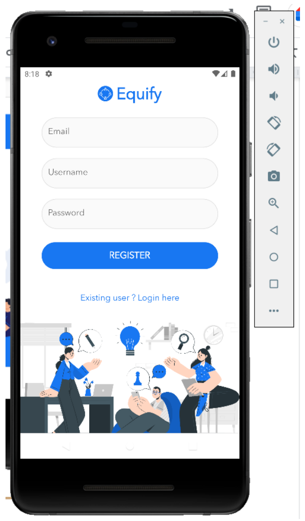
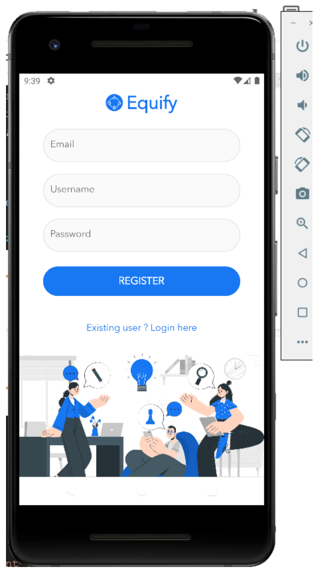
8:17

Equify

Search for camps

Search for a camp by entering its name in the search bar above

Search

Login	Register
	

Login screenshot:

Equify

Email

Username

Password

REGISTER

Existing user ? Login here

Register screenshot:

Equify

Email

Username

Password

REGISTER

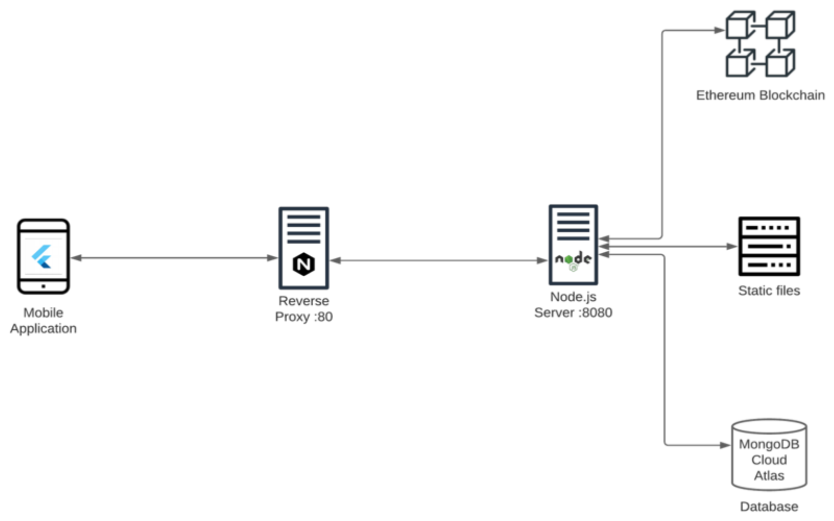
Existing user ? Login here

Tools And technologies used

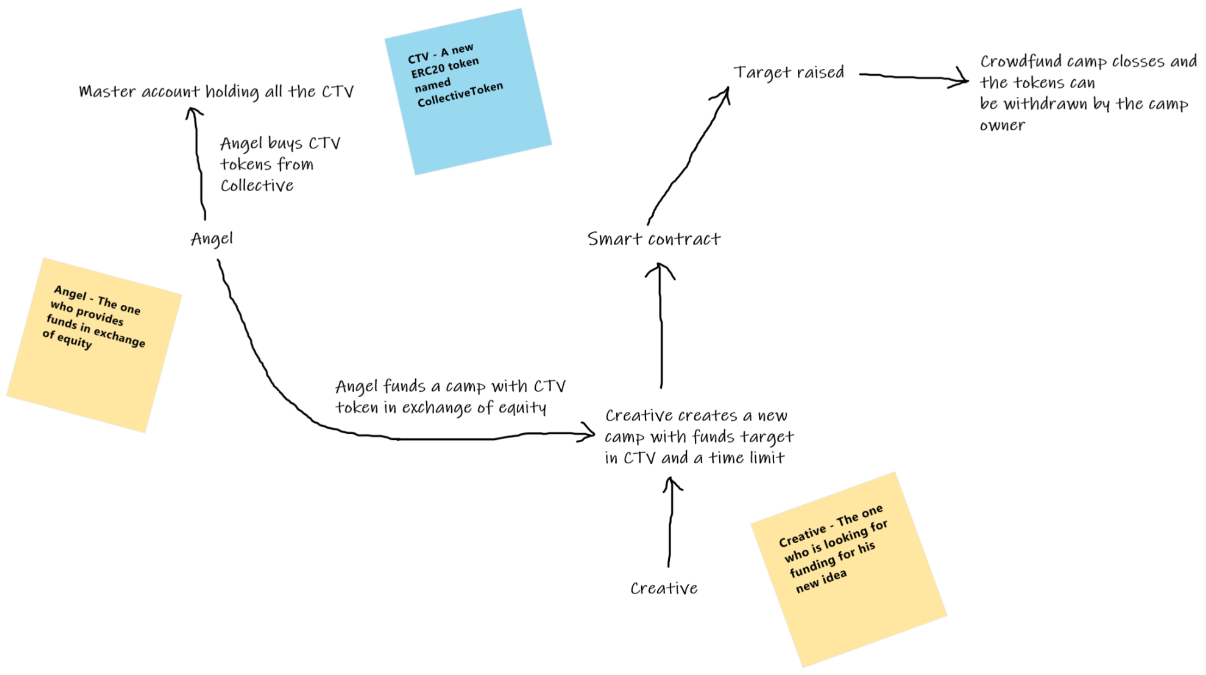
1. **Frontend** - Flutter, Dart
2. **Backend** - Node.js, Express
3. Packages used
    a. Truffle Wallet provider
    b. Axios
    c. Express
    d. Web3
    e. Mocha http
4. **Blockchain** - Ethereum Solidity 0.8.0
5. **Hosting**
    a. Ethereum test net nodes - Moralis Speedy Nodes
    b. Ropsten test net
6. **Database** - Mongoose & MongoDB

6. Implementation Details

6.1 High level system design



6.2 How equity crowdfunding works


CTV - A new ERC20 token named CollectiveToken
Master account holding all the CTV
Angel buys CTV tokens from Collective
Angel
Angel - The one who provides funds in exchange of equity
Angel funds a camp with CTV token in exchange of equity
Creative creates a new camp with funds target in CTV and a time limit
Creative
Creative - The one who is looking for funding for his new idea
Smart contract
Target raised
Crowdfund camp closses and the tokens can be withdrawn by the camp owner

Camp and collaborator details

Camp - A camp that is created by a Creative (OP - Original Posters) on Equify to raise funding

Camp details structure consist of a list of angels, target, equity and whether target is reached or not.

```
// Camp details struct

  struct CampDetails{
      bool campExists;
      uint fundingRaised;
      address[] angelList;
      CollaboratorsList[] colList;
      uint target;
      uint equity;
      bool targetReached;
  }
```

Angel - Angels are the users/investors who buy equity in exchange of CTV - CollectiveToken

Structure of collaborators

Col - Collaborator who collaborates in a camp in exchange of CTV
It has three fields amount, collaborators positions and address

```
// Struct to save collaborator details

  struct CollaboratorsList {
      address colAddress;
      uint amount;
      string position;
  }
```

Purchasing Equity
It ensures camp should exist before trying to get equity. If the campus funding limit is reached after adding an amount then we mark the target reached as true.

```
// Buying equity in the camp
```

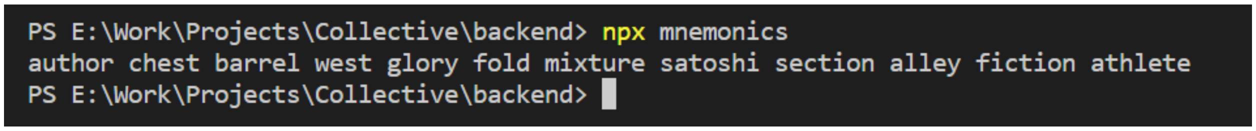
```
  function buyEquity(address _angel,address _camp,uint _amount) public {
      require(camps[_camp].campExists == true && camps[_camp].targetReached == false,'Camp
not found');
      if(camps[_camp].fundingRaised + _amount >= camps[_camp].target){
          camps[_camp].targetReached = true;
      }
      funding[_camp][_angel] = funding[_camp][_angel] + _amount;
      camps[_camp].fundingRaised = camps[_camp].fundingRaised + _amount;
      camps[_camp].angelList.push(_angel);


      if(camps[_camp].targetReached){
          emit targetReachedForCamp( _camp);
      }

  }
```

Collab
Checking if a camp exists or not. If it exists then push to the collaborators list.

```
  // Collaborate in a camp

  function collab(address _col,address _camp,string memory _position,uint _amount) public {
      require(camps[_camp].campExists == true,'Camp not found');
      camps[_camp].colList.push(CollaboratorsList({
          colAddress : _col,
          amount : _amount,
          position : _position
      }));
  }
```

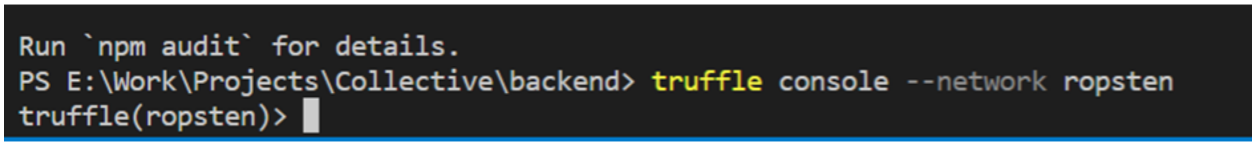
Deploying to TestNet

Generating Mnemonics



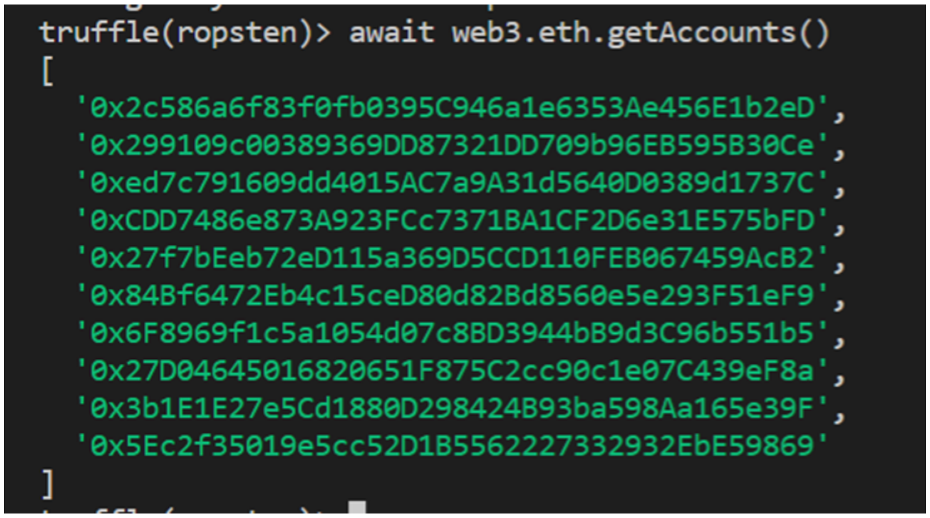
```
PS E:\Work\Projects\Collective\backend> npx mnemonics
author chest barrel west glory fold mixture satoshi section alley fiction athlete
PS E:\Work\Projects\Collective\backend>
```

Connecting to the truffle console



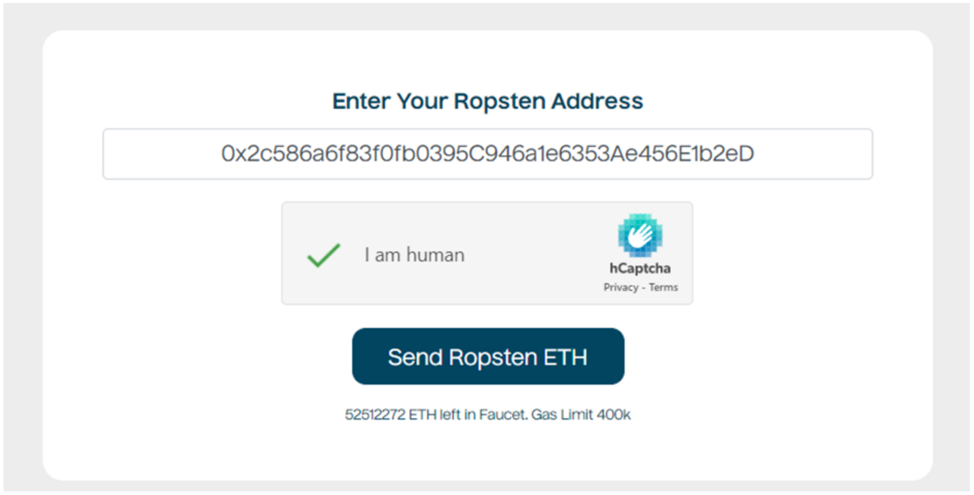
```
Run `npm audit` for details.
PS E:\Work\Projects\Collective\backend> truffle console --network ropsten
truffle(ropsten)>
```

Getting available accounts



```
truffle(ropsten)> await web3.eth.getAccounts()
[
  '0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD',
  '0x299109c00389369DD87321DD709b96EB595B30Ce',
  '0xed7c791609dd4015AC7a9A31d5640D0389d1737C',
  '0xCDD7486e873A923FCc7371BA1CF2D6e31E575bFD',
  '0x27f7bEeb72eD115a369D5CCD110FEB067459AcB2',
  '0x84Bf6472Eb4c15ceD80d82Bd8560e5e293F51eF9',
  '0x6F8969f1c5a1054d07c8BD3944bB9d3C96b551b5',
  '0x27D04645016820651F875C2cc90c1e07C439eF8a',
  '0x3b1E1E27e5Cd1880D298424B93ba598Aa165e39F',
  '0x5Ec2f35019e5cc52D1B5562227332932EbE59869'
]
```

Get some etherium from faucet



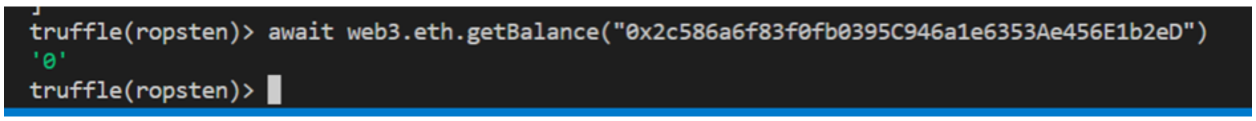
Enter Your Ropsten Address

0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD

52512272 ETH left in Faucet. Gas Limit 400k

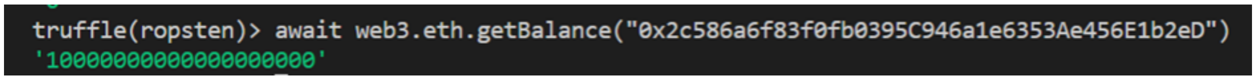
Check for balance
await web3.eth.getBalance("0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD")

Before



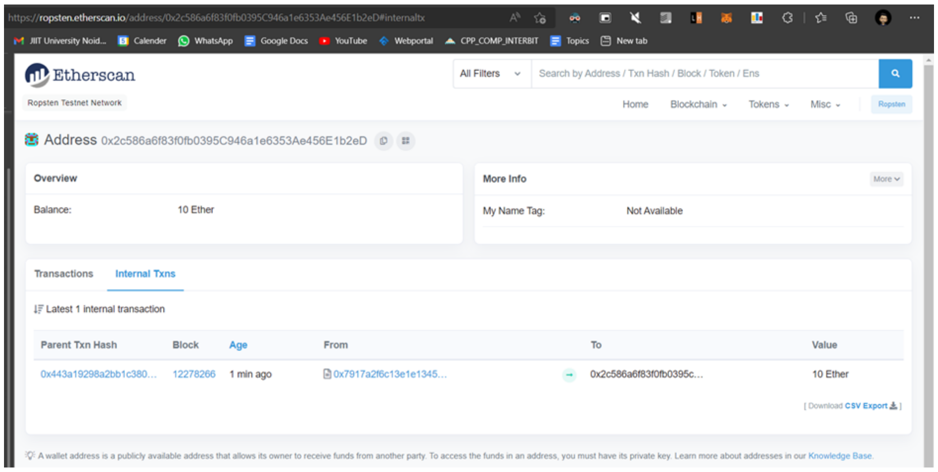
```
truffle(ropsten)> await web3.eth.getBalance("0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD")
'0'
truffle(ropsten)> 
```

After



```
truffle(ropsten)> await web3.eth.getBalance("0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD")
'10000000000000000000'
```

Checking on Etherscan



Address 0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD

Overview

Balance: 10 Ether

More Info

My Name Tag: Not Available

Transactions | Internal Txns

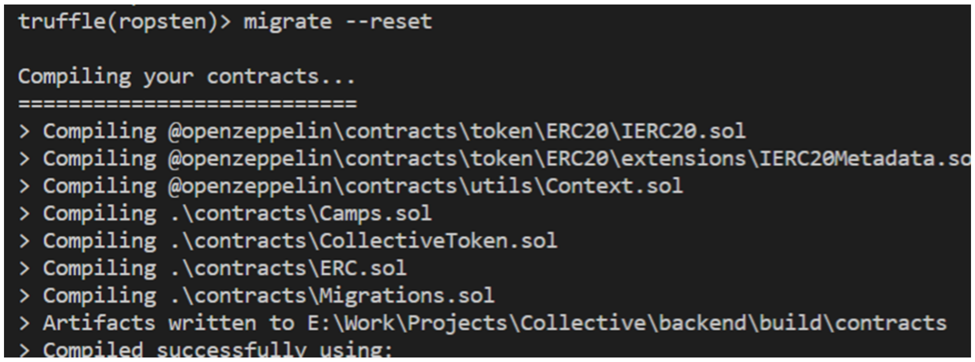
Latest 1 internal transaction

Parent Txn Hash	Block	Age	From	To	Value
0x443a19298a2bb1c380...	12278266	1 min ago	0x7917a2f6c13e1e1345...	0x2c586a6f83f0fb0395c...	10 Ether

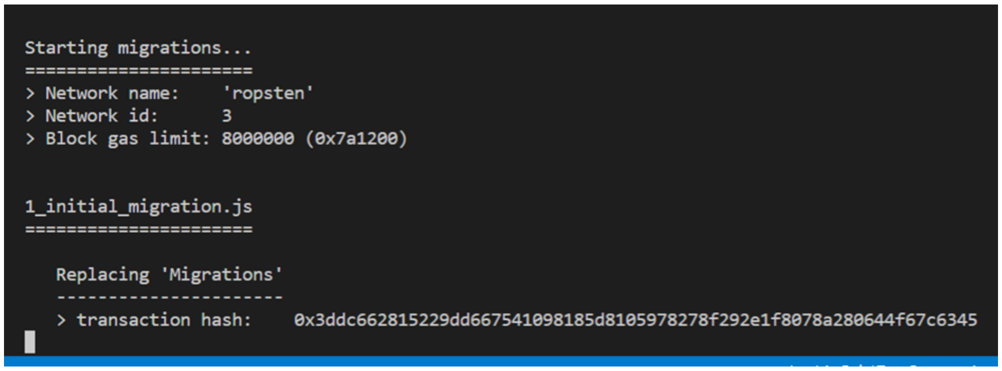
[Download CSV Export]

A wallet address is a publicly available address that allows its owner to receive funds from another party. To access the funds in an address, you must have its private key. Learn more about addresses in our Knowledge Base.

Now run migrate command

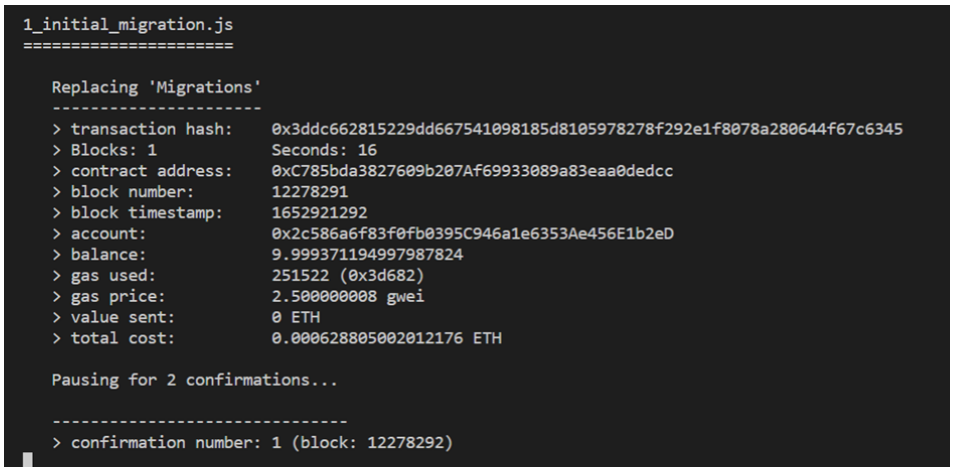


```
truffle(ropsten)> migrate --reset

Compiling your contracts...
===========================
> Compiling @openzeppelin\contracts\token\ERC20\IERC20.sol
> Compiling @openzeppelin\contracts\token\ERC20\extensions\IERC20Metadata.so
> Compiling @openzeppelin\contracts\utils\Context.sol
> Compiling .\contracts\Camps.sol
> Compiling .\contracts\CollectiveToken.sol
> Compiling .\contracts\ERC.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to E:\Work\Projects\Collective\backend\build\contracts
> Compiled successfully using:
```



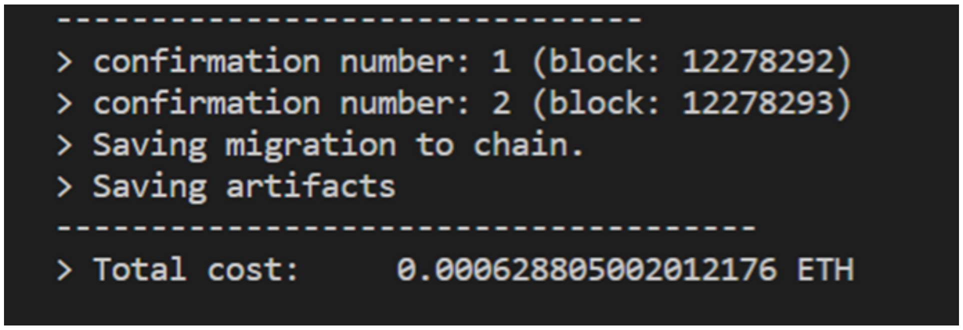
```
Starting migrations...
======================
> Network name:    'ropsten'
> Network id:      3
> Block gas limit: 8000000 (0x7a1200)


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x3ddc662815229dd667541098185d8105978278f292e1f8078a280644f67c6345
```

Pausing for 2 confirmations



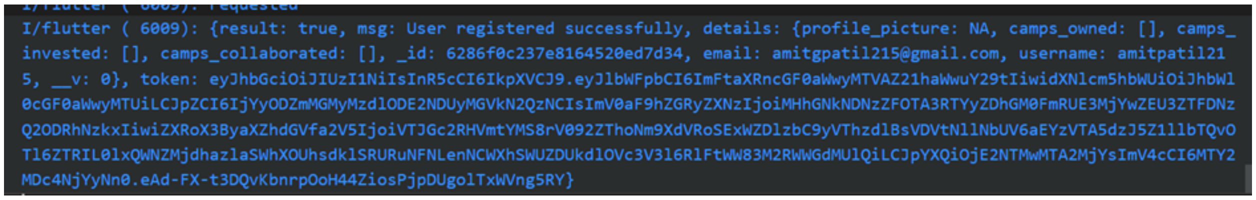
```
1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0x3ddc662815229dd667541098185d8105978278f292e1f8078a280644f67c6345
   > Blocks: 1            Seconds: 16
   > contract address:    0xC785bda3827609b207Af69933089a83eaa0dedcc
   > block number:        12278291
   > block timestamp:     1652921292
   > account:             0x2c586a6f83f0fb0395C946a1e6353Ae456E1b2eD
   > balance:             9.999371194997987824
   > gas used:            251522 (0x3d682)
   > gas price:           2.500000008 gwei
   > value sent:          0 ETH
   > total cost:          0.000628805002012176 ETH

   Pausing for 2 confirmations...

   ------------------------------
   > confirmation number: 1 (block: 12278292)
```

After 2 confirmations



```
   ------------------------------
   > confirmation number: 1 (block: 12278292)
   > confirmation number: 2 (block: 12278293)
   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:     0.000628805002012176 ETH
```

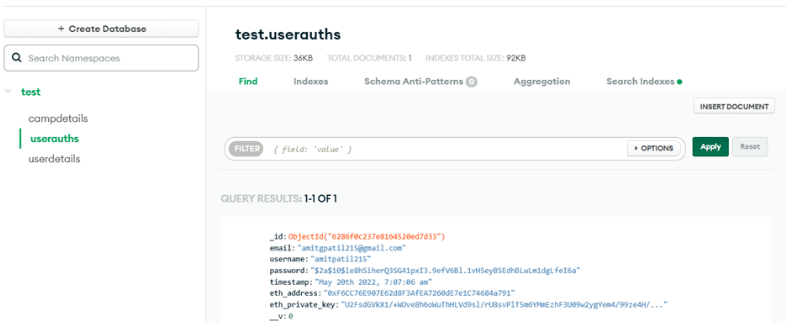
User registration response



```
I/flutter ( 6009): requested
I/flutter ( 6009): {result: true, msg: User registered successfully, details: {profile_picture: NA, camps_owned: [], camps_
invested: [], camps_collaborated: [], _id: 6286f0c237e8164520ed7d34, email: amitgpatil215@gmail.com, username: amitpatil21
5, __v: 0}, token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFtaXRncGF0aWwyMTVAZ21haWwuY29tIiwidXNlcm5hbWUiOiJhbWl
0cGF0aWwyMTUiLCJpZCI6IjYyODZmMGMyMzdlODE2NDUyMGVkN2QzNCIsImV0aF9hZGRyZXNzIjoiMHhGNkNDNzZFOTA3RTYyZDhGM0FmRUE3MjYwZEU3ZTFDNz
Q2ODRhNzkxIiwiZXRoX3ByaXZhdGVfa2V5IjoiVTJGc2RHVmtYMS8rV092ZThoNm9XdVRoSExWZDlzbC9yVThzdlBsVDVtNllNbUV6aEYzVTA5dzJ5Z1llbTQvO
Tl6ZTRIL0lxQWNZMjdhazlaSWhXOUhsdklSRURuNFNLenNCWXhSWUZDUkdlOVc3V3l6RlFtWW83M2RWWGdMUlQiLCJpYXQiOjE2NTMwMTA2MjYsImV4cCI6MTY2
MDc4NjYyNn0.eAd-FX-t3DQvKbnrpOoH44ZiosPjpDUgolTxWVng5RY}
```

MongoDB Collections

1. User Authentication Collection

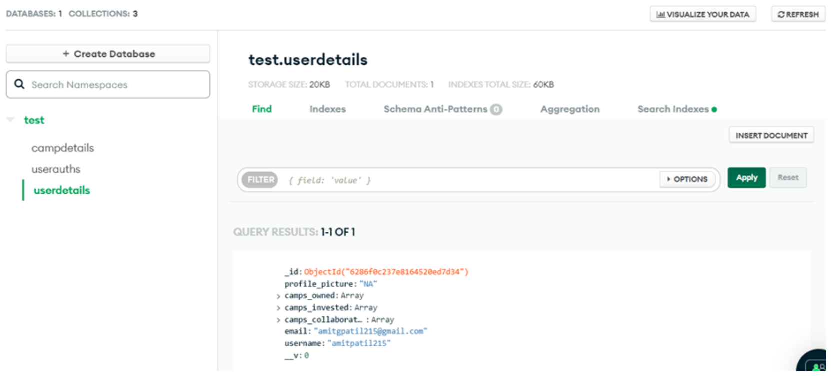

test.userauths

STORAGE SIZE: 36KB TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 92KB

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId("6286f0c237e8164520ed7d33")
email: "amitgpatil215@gmail.com"
username: "amitpatil215"
password: "$2a$10$le8hSiherQ3SG41pxI3.9efV6Bl.1vH5eyBSEdhBLwLm1dgLfeI6a"
timestamp: "May 20th 2022, 7:07:06 am"
eth_address: "0xF6CC76E907E62d8F3AfEA7260dE7e1C74684a791"
eth_private_key: "U2FsdGVkX1/+WOve8h6oWuThHLVd9sl/rU8svPlT5m6YMmEzhF3U09w2ygYem4/99ze4H/..."
__v: 0
```

2. User Details


DATABASES: 1 COLLECTIONS: 3

test.userdetails

STORAGE SIZE: 20KB TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 60KB

QUERY RESULTS: 1-1 OF 1

```
  _id: ObjectId("6286f0c237e8164520ed7d34")
  profile_picture: "NA"
> camps_owned: Array
> camps_invested: Array
> camps_collaborat… : Array
  email: "amitgpatil215@gmail.com"
  username: "amitpatil215"
  __v: 0
```

Test Results

Create camp,buy equity and get details

- ✔ Create a new camp
  Testing working of new create camp functionality
- ✔ Buy equity
  Testing working of equity purchase by the user
- ✔ Get camp details
  Getting camp details which is a view function.

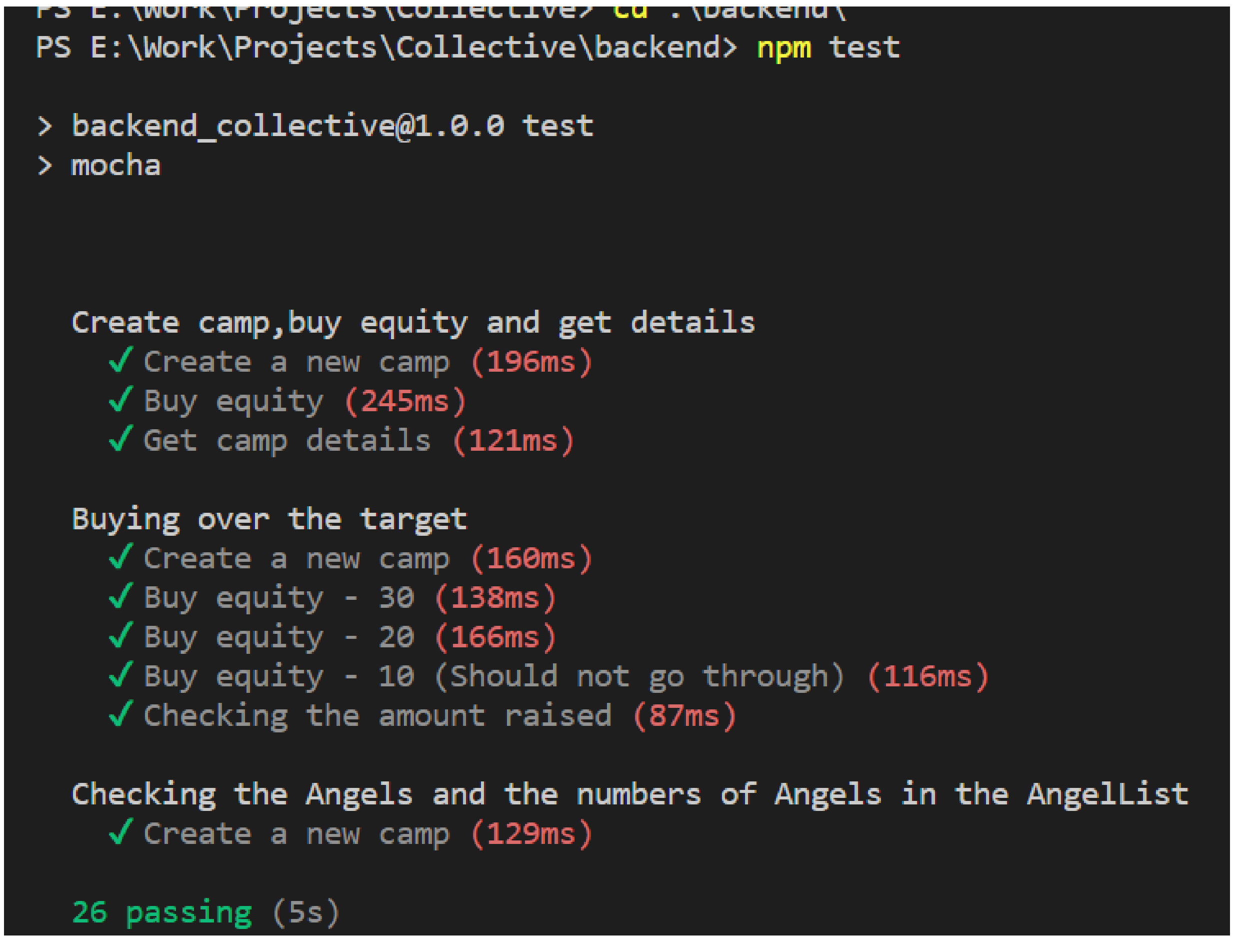
Buying over the target

- ✔ Create a new camp
  Testing Creation of new camp
- ✔ Buy equity - 30
  Purchasing Equity 30 percent
- ✔ Buy equity - 20
  Purchasing equity 20 Percent
- ✔ Buy equity - 10
  Purchasing Equity 10 percent
- ✔ Checking the amount raised
  Testing view function to check how much amount is raised

Checking the Angels and the numbers of Angels in the AngelList

- ✔ Create a new camp
- ✔ Create a new camp
- ✔ Buy equity - 20
- ✔ Checking the amount raised
- ✔ Adding First collaborator with amount
  Testing Collaborator function
- ✔ Adding Second collaborator with amount
  Adding more collaborators
- ✔ Fetching and checking camp
  Fetching camp and checking camp details

Mocha test details



```
PS E:\Work\Projects\Collective> cd .\backend\
PS E:\Work\Projects\Collective\backend> npm test

> backend_collective@1.0.0 test
> mocha



  Create camp,buy equity and get details
    ✓ Create a new camp (196ms)
    ✓ Buy equity (245ms)
    ✓ Get camp details (121ms)

  Buying over the target
    ✓ Create a new camp (160ms)
    ✓ Buy equity - 30 (138ms)
    ✓ Buy equity - 20 (166ms)
    ✓ Buy equity - 10 (Should not go through) (116ms)
    ✓ Checking the amount raised (87ms)

  Checking the Angels and the numbers of Angels in the AngelList
    ✓ Create a new camp (129ms)

  26 passing (5s)
```

References

1. “Decentralized crowdfunding platform using smart contracts,” *IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9774132. [Accessed: 22-May-2022].
2. Starkenmann, Olivier. (2017). Implementation of a Crowdfunding Decentralized Application on Ethereum.
3. Ashari, Firmansyah. (2020). Smart Contract and Blockchain for Crowdfunding Platform. International Journal of Advanced Trends in Computer Science and Engineering. 9. 3036-3041. 10.30534/ijatcse/2020/83932020.
4. Roth, Jakob and Schär, Fabian and Schöpfer, Aljoscha, The Tokenization of Assets: Using Blockchains for Equity Crowdfunding (August 27, 2019).
5. Applying Ethereum Smart Contracts to Blockchain-Based Crowdfunding System to Increase Trust and Information Symmetry | 2021 7th International Conference on Computer Technology Applications, 2022