DANCE COMMUNITY & ENGAGEMENT USING BLOCKCHAIN

Design Report- Phase 4

AUTHORS:

MOJITHA MOHANDAS KURUP
THELMA MELISSA GOMES

DaNce Community and Engagement (DNCE) using Blockchain

(Design Report - Phase 4)

Title: Dance Community and Engagement
First Name 1: Thelma Melissa
Last name 1: Gomes
Email: thelmame@buffalo.edu
Person number: 50416894

First Name 2: Mojitha Mohandas
Last name 2: Kurup
Email: mojitham@buffalo.edu
Person number: 50414795

Contract Address: 0x65122839eFB002eb7F59BbA2ACF8d2209A0C50ca

Introduction:

As dance enthusiasts ourselves, we wanted to implement a project that reflects our interest by building a good system for any issue around it. A common observation made in the dance industry is that due to lack of a global platform for dancers, creators, students and studio spaces to engage with each other, it gets difficult to find good and new talent to continue building a global community.

With our idea, we intend to bring this community closer by providing a global platform for all kinds of people with a passion for dance. Our main goal is to help users (students) to choose their desired classes with all kinds of options, help teachers to publish their dance classes to a wider audience and studio space owners to be able to rent their spaces effectively with more trust.

About our idea:

Our Ethereum based Blockchain is for Dance Community and Engagement. As mentioned earlier, through a fungible token we create using our smart contract, we provide a forum for all dance lovers to engage and build a great community that combines business and passion. We will implement a ERC-20 standard based token for the above system.

Our token symbol is 'DNCE'. We perceive that keeping this name for our token gives a lovely tone to day-to-day conversations including it – for example, someone says, "Hey can you help me with some DNCE tokens? I'm falling short of them and want to enroll in an upcoming class!"

Why our system can be useful:

Our system aims to bridge the communication and engagement gap between the users involved in real life. Also, by having a decentralized system, the formality of a particular person/ entity taking control is avoided, thus it facilitates diversification and optimizes resource distribution among the users. It also provides an added trust layer between the entrusted parties (which is the essence of blockchain), by verification and validation of the users through application-specific conditions.

For each of our users, this can be useful in some or the other way. For example, teachers, especially the ones who are new and not yet popular, get an equal opportunity to showcase their work and gain clients/ students. As Blockchain ensures immutable ledgers, Studio owners can be assured of the parties' details involved with them while renting the space, thus avoiding any losses. In traditional systems, there can be a scope of abandonment which can risk losses to the owner. Students wanting to learn new dance forms, can do so with a variety of options – with popular and budding teachers being some of them. They can also share or transfer their DNCE tokens among peers to build a friendly community while also learning and growing themselves.

Requirements for Phase 4:

- Application Smart Contract with ERC20 token
- Dapp Web Interface
- Truffle (Development and Testing IDE)
- Ganache (Etheruem Blockchain provider for testing)
- Node.js (Web Server)
- Npm (Node Package Manager)
- Metamask Wallet
- Remix IDE (Optional)
- Visual Studio Code
- Code implementation screenshots
- Architectural diagrams

Technologies/ Skills used:

- Solidity
- HTML, CSS, Javascript
- Web3 Provider

User information:

Our system has different kinds of users: clients/ students, teachers and studio owners.

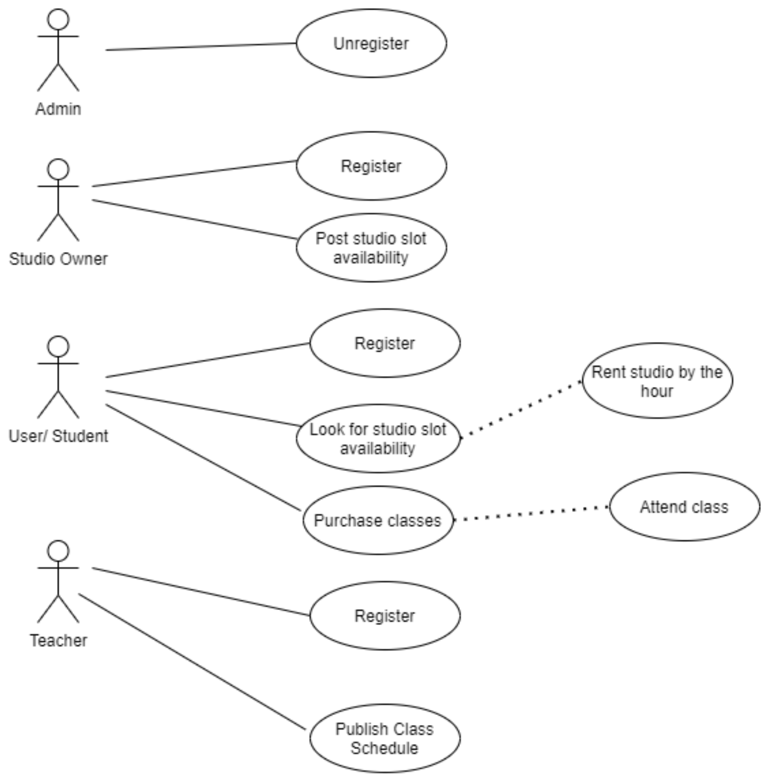
Below are the features for each kind of user:

1. Teachers: They can publish their classes.
2. Students: They can purchase a class they desire.
3. Studio Renter: They can rent a studio space from the owner/ previous buyer.
4. Studio Owner: They can lease their studio space by providing slot availability.

Architectural Diagrams:

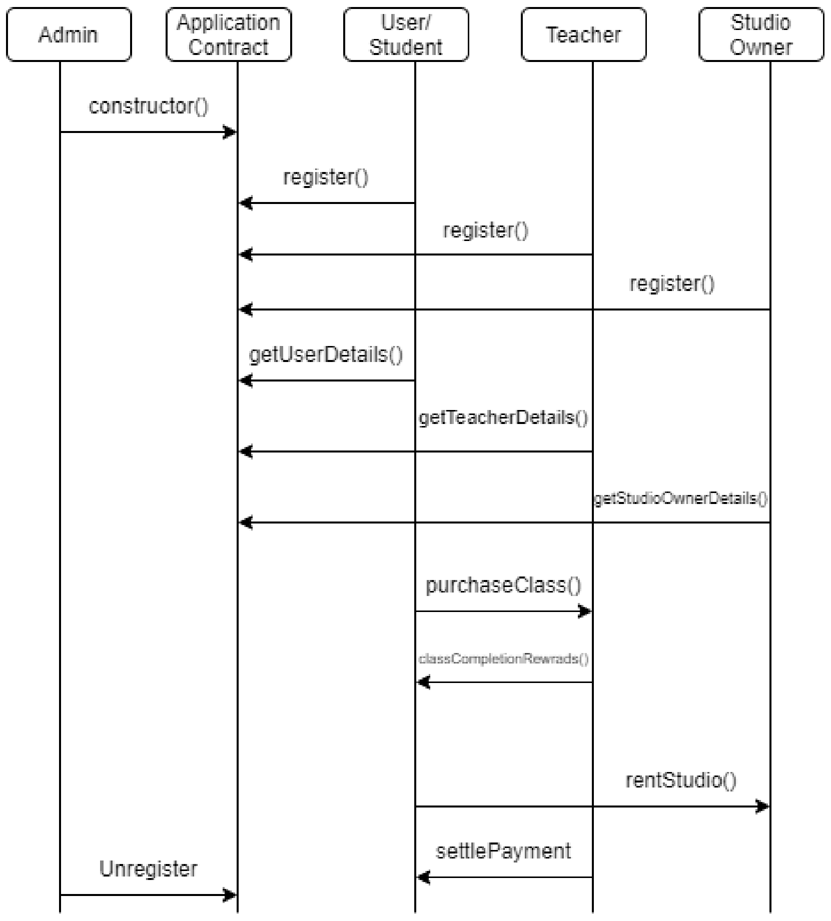
1. **Use – Case Diagram:**

Fig:1.1



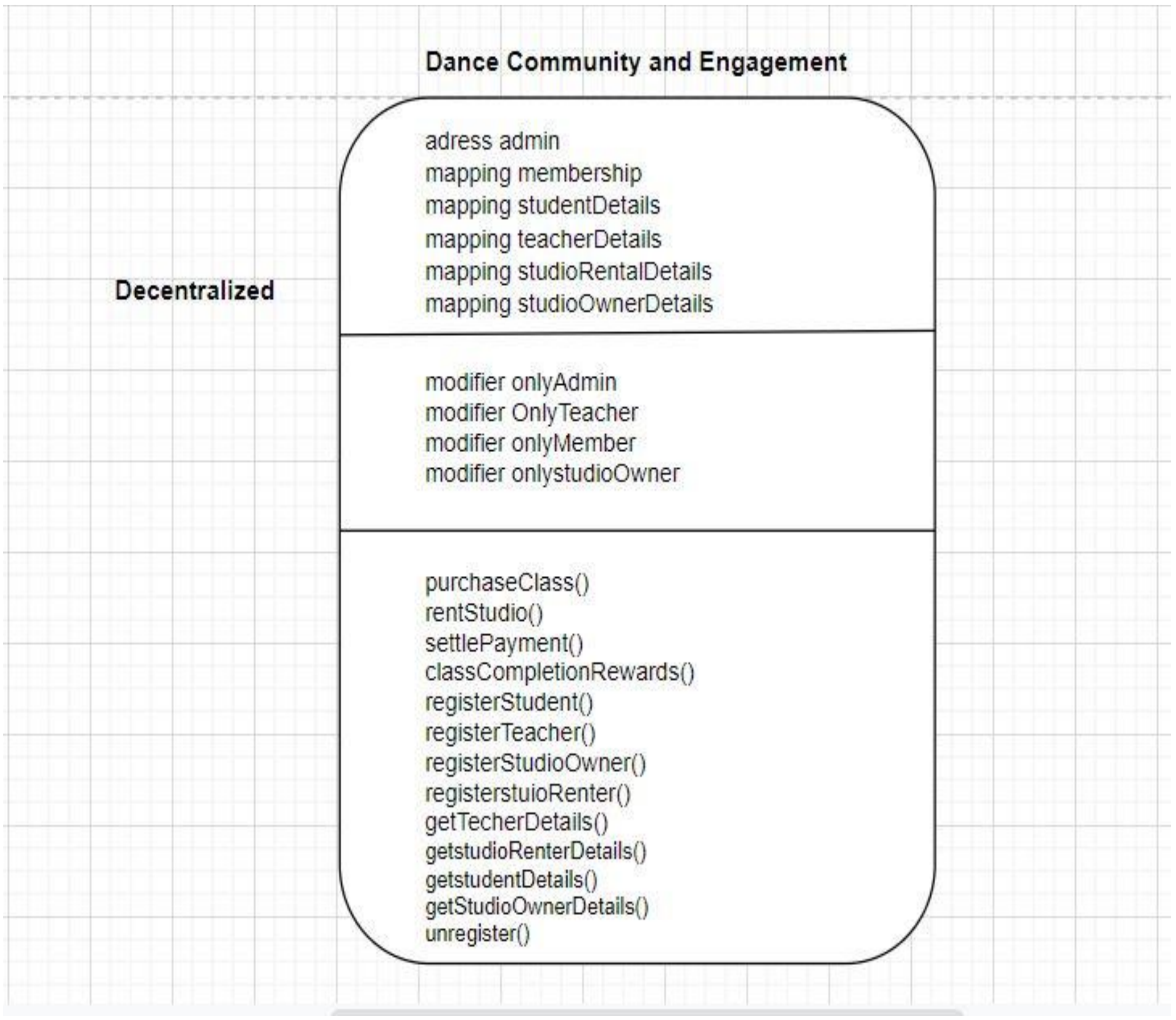
2. **Sequence Diagram**

Fig 1.2



3. **Contract Diagram:**

Fig 1.3



Working of Dapp along with Smart Contract integration:

Our application provides a convenient web interface for the users of our system to Register themselves, check the details of themselves and fellow users as well as purchase classes or rent studio spaces. Additionally, rewards are granted for users who purchase more than 5 classes.

Below are the steps of how our DNCE Decentralized application is implemented and launched:

Part 1: Implementation and testing of Smart Contract

1. Open Ganache once installed. Launch a new QuickStart workspace. Note the details for the mnemonic and network as it will be needed further.
2. In Chrome browser, open Metamask Wallet and add a new Network. Fill in the below details. This will then link you wallet to Ganache.

   RPC Server URL and port, Network ID, Chain ID.

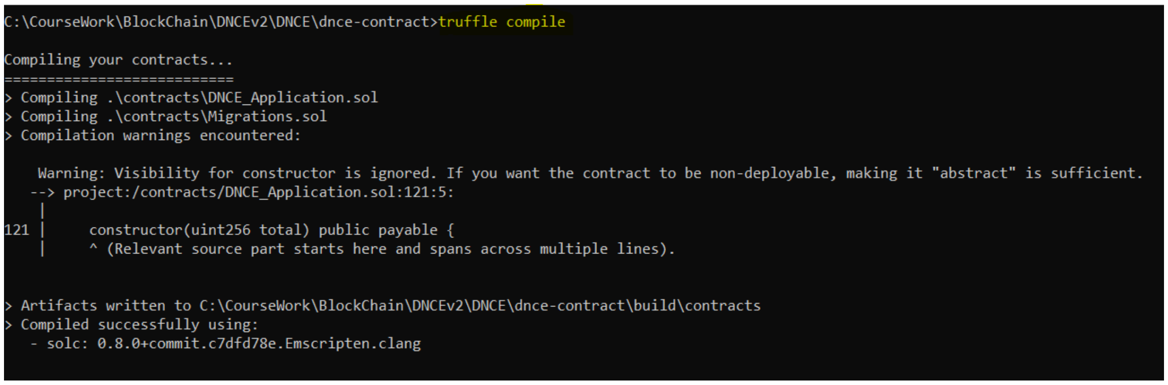
3. If you were already connected over another network in Metamask (say Ropsten), lock out of it and recover the account required for Ganache with its seed phrase (mnemonic).
4. In your file explorer, make a new directory for your project workspace – DNCE.
5. cd into it and create 2 new directories to maintain the structure of a Blockchain Dapp.
   dnce-app
   dnce-contract
6. Open CMD terminal in that workspace.
7. cd into dnce-contract and now we'll begin testing and implementing the application smart contract.
8. Execute the below command to create the truffle environment.
   truffle init
9. It will create the required directories and truffle-config.js (this file will have our network details).
10. Add your Solidity (.sol) code file in the dnce-contract/contracts/ folder.
11. Compile all the contracts with the below command.
    truffle compile
12. Once the compilation is successful, you can execute the below command to implement the contracts and create their respective addresses.
    truffle migrate –reset
13. Note down the Smart contract address for the DNCE application for future use.

Part 2: Implementation of DNCE Dapp:

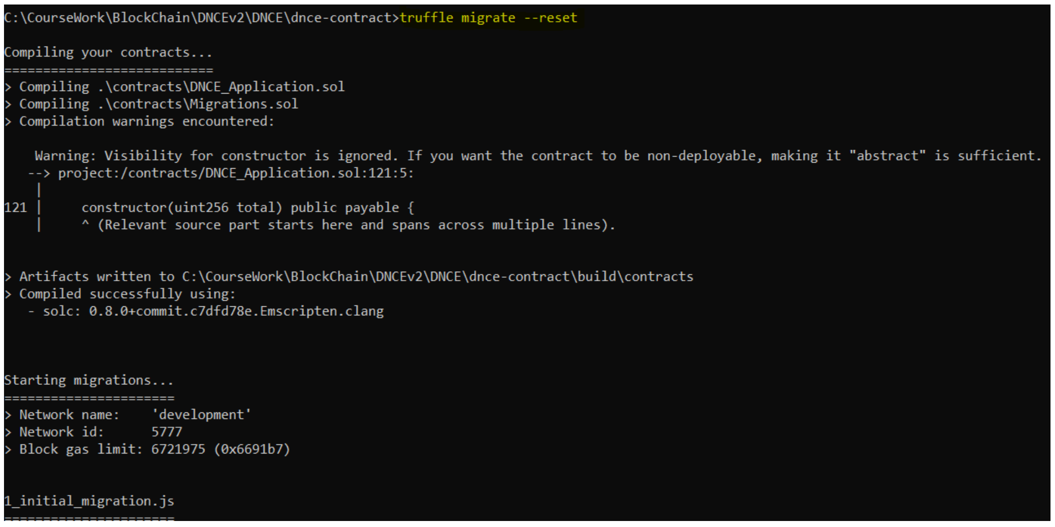
1. cd into dnce-app directory of your application.
2. Create a ‘src’ folder and add all the HTML, CSS and Javascript files required for your Web interface.
3. In order to host the web app, we need to launch it using npm.
4. Execute the below commands:
   npm init
   npm install
   npm start
5. Once the session is started, we can now launch the web app on localhost listening on port 3000.

Screenshots:

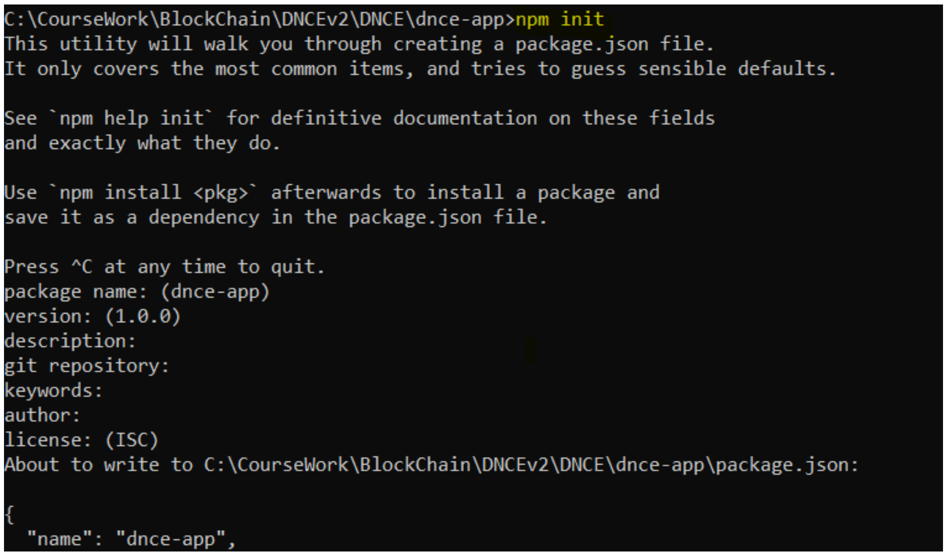
Truffle & Ganache setup:



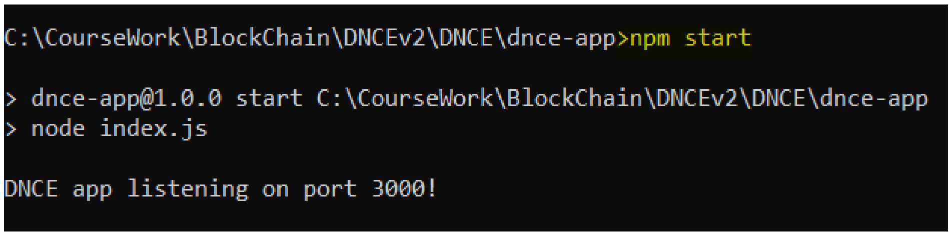
```
C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-contract>truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\DNCE_Application.sol
> Compiling .\contracts\Migrations.sol
> Compilation warnings encountered:

    Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
   --> project:/contracts/DNCE_Application.sol:121:5:
    |
121 |     constructor(uint256 total) public payable {
    |     ^ (Relevant source part starts here and spans across multiple lines).


> Artifacts written to C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-contract\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang
```



```
C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-contract>truffle migrate --reset

Compiling your contracts...
===========================
> Compiling .\contracts\DNCE_Application.sol
> Compiling .\contracts\Migrations.sol
> Compilation warnings encountered:

    Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
   --> project:/contracts/DNCE_Application.sol:121:5:
    |
121 |     constructor(uint256 total) public payable {
    |     ^ (Relevant source part starts here and spans across multiple lines).


> Artifacts written to C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-contract\build\contracts
> Compiled successfully using:
   - solc: 0.8.0+commit.c7dfd78e.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


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

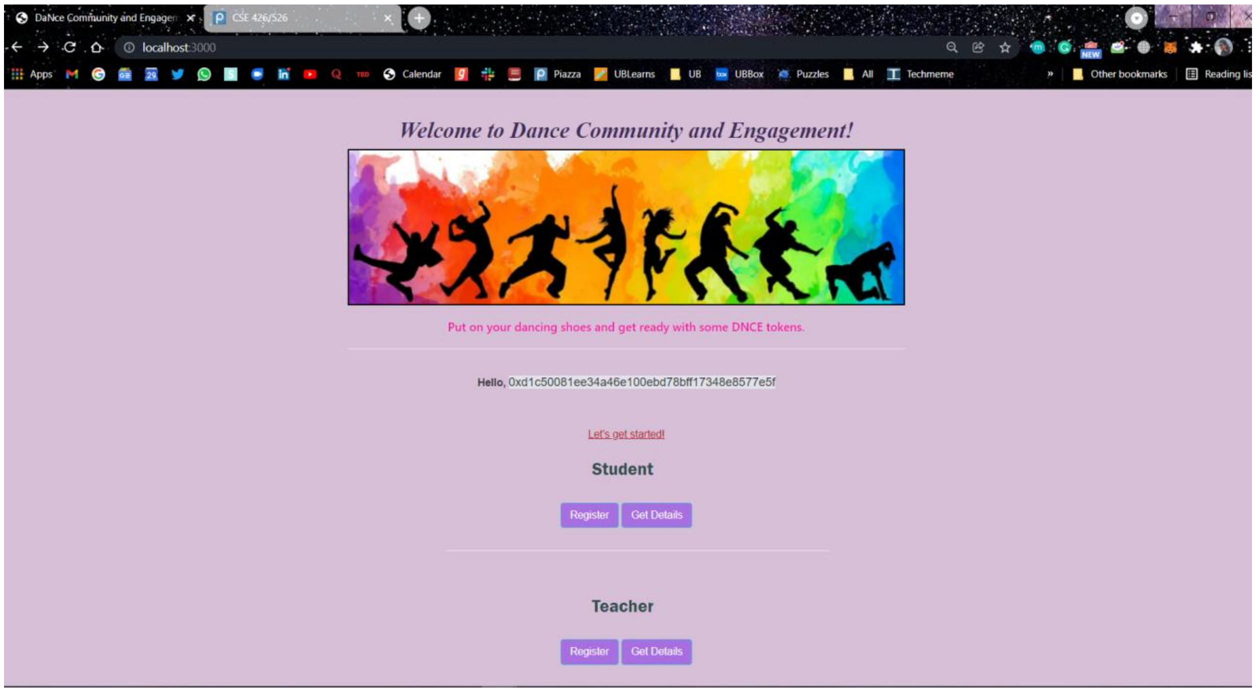


```
C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-app>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (dnce-app)
version: (1.0.0)
description:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-app\package.json:

{
  "name": "dnce-app",
```



```
C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-app>npm start

> dnce-app@1.0.0 start C:\CourseWork\BlockChain\DNCEv2\DNCE\dnce-app
> node index.js

DNCE app listening on port 3000!
```

<u>Homepage:</u>



Welcome to Dance Community and Engagement!

Put on your dancing shoes and get ready with some DNCE tokens.

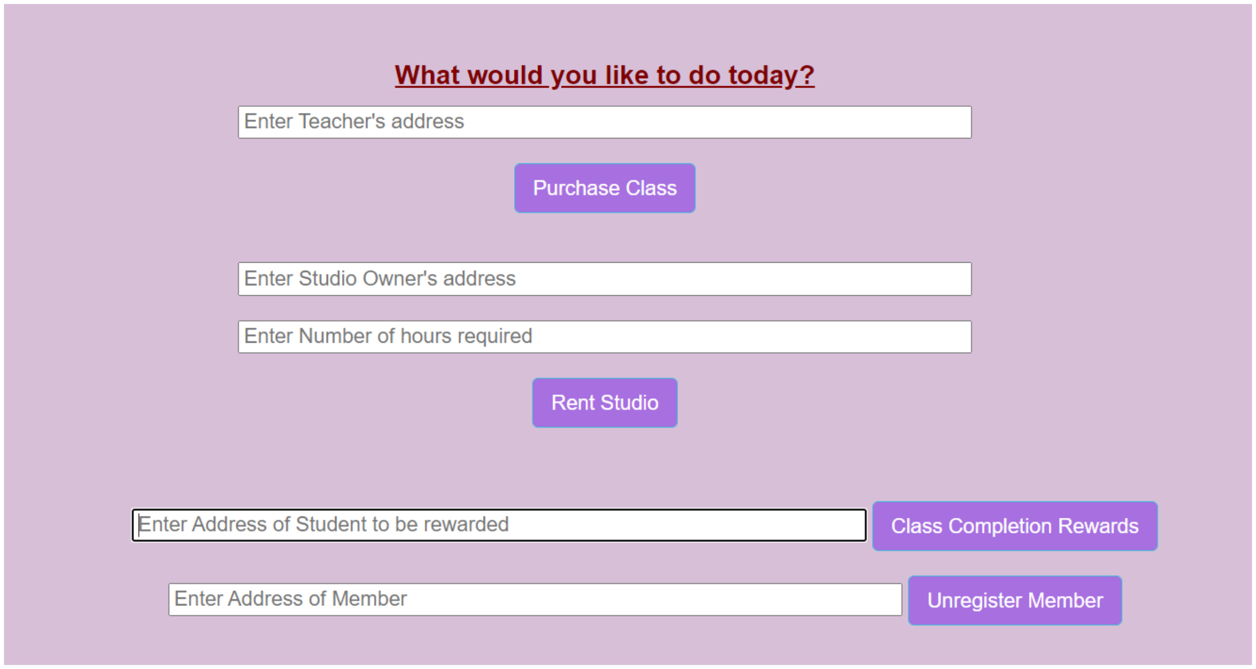
Hello, 0xd1c50081ee34a46e100ebd78bff17348e8577e5f

Let's get started!

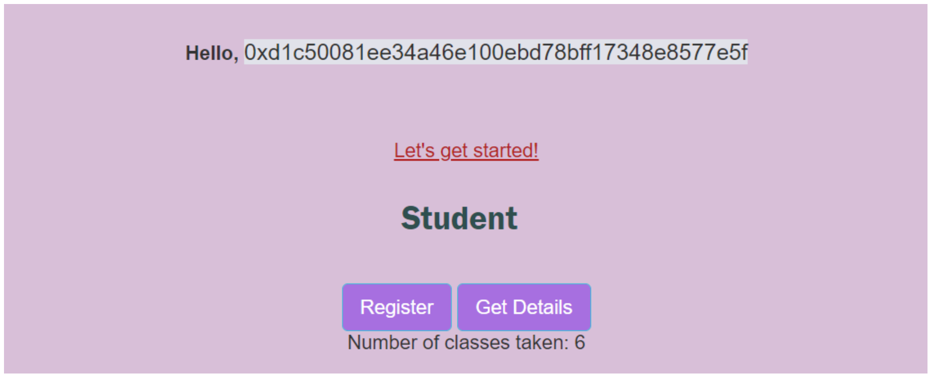
Student

Teacher

Special functions:

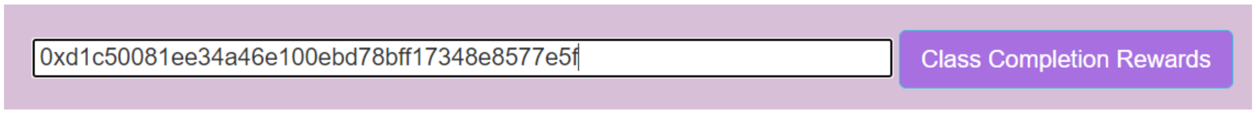

What would you like to do today?
Enter Teacher's address
Purchase Class
Enter Studio Owner's address
Enter Number of hours required
Rent Studio
Enter Address of Student to be rewarded
Class Completion Rewards
Enter Address of Member
Unregister Member

Eg. of Special functions -- classCompletionRewards(): where students will be rewarded with 20 tokens on completion of 6 or more classes

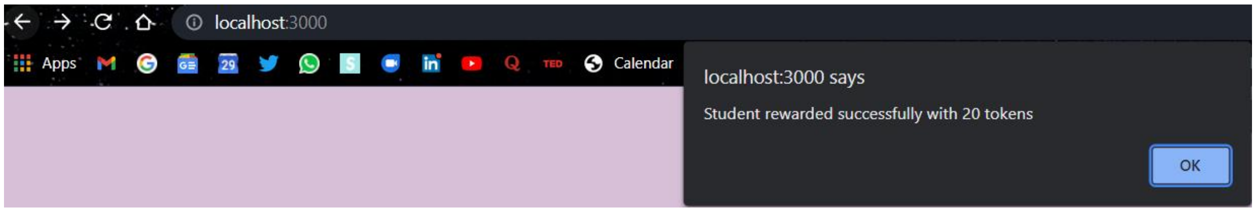


Hello, 0xd1c50081ee34a46e100ebd78bff17348e8577e5f

Number of classes taken: 6



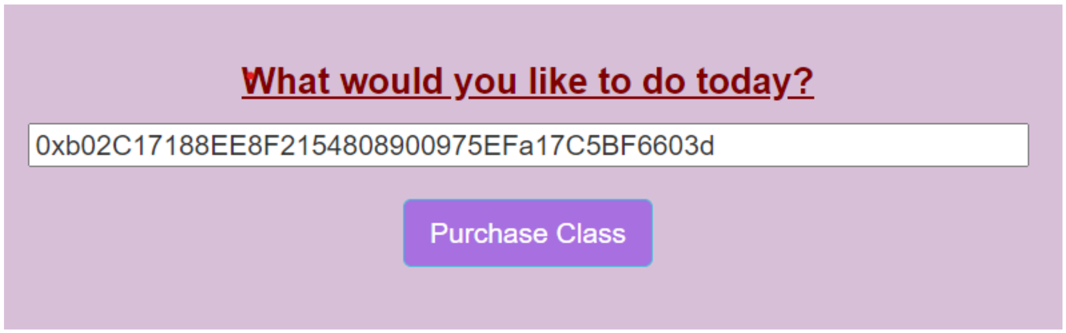
0xd1c50081ee34a46e100ebd78bff17348e8577e5f



localhost:3000 says

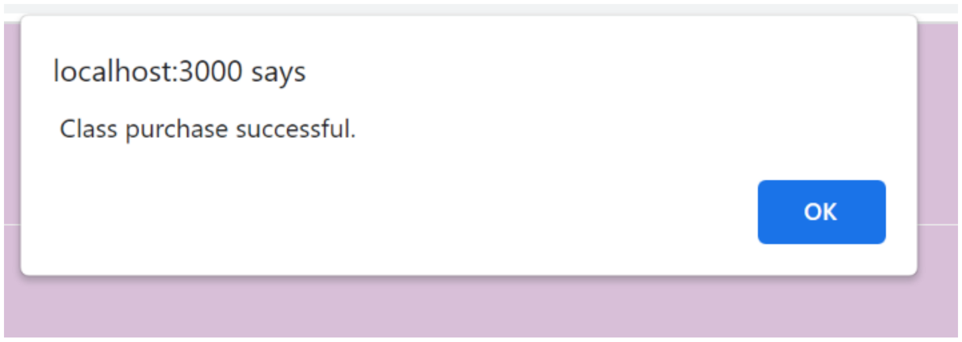
Student rewarded successfully with 20 tokens

purchaseClass(): using teacher's decentralized identity



What would you like to do today?

0xb02C17188EE8F2154808900975EFa17C5BF6603d

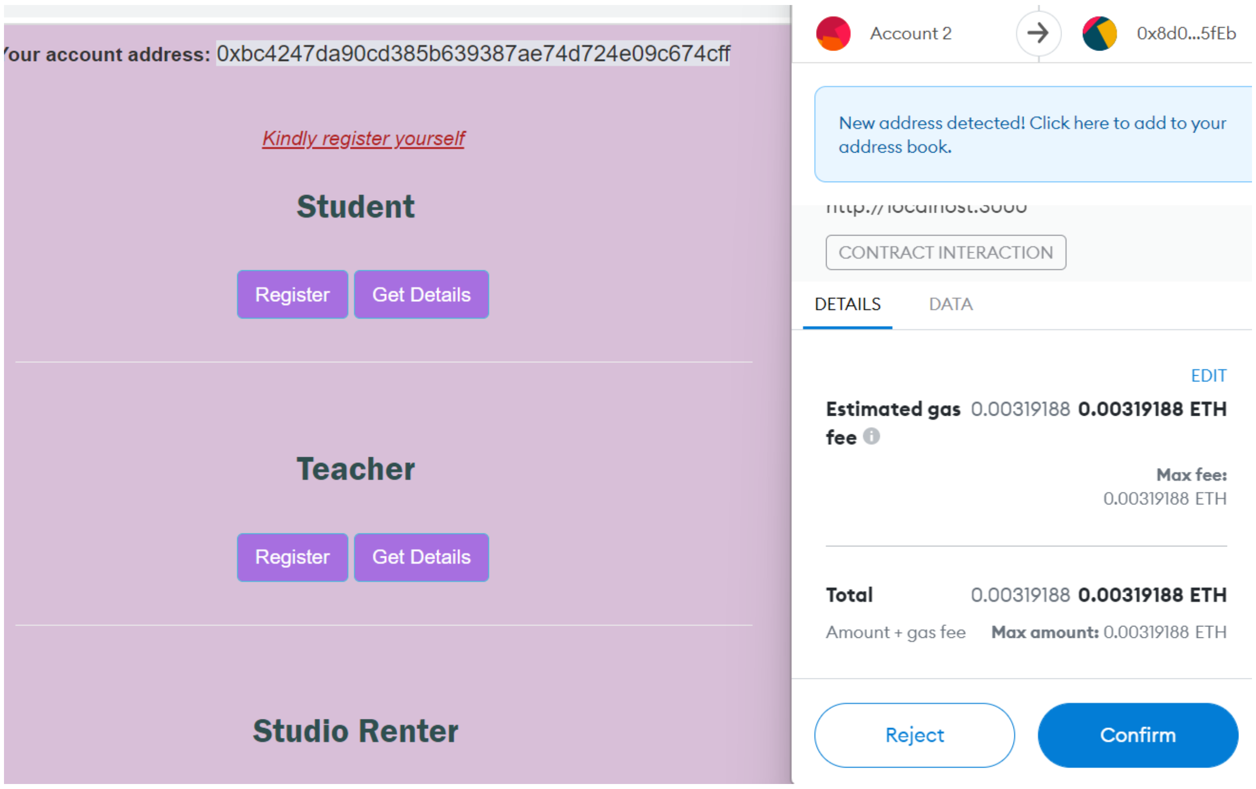


localhost:3000 says

Class purchase successful.

User Registration:

1. **Student**



'our account address: 0xbc4247da90cd385b639387ae74d724e09c674cff

Kindly register yourself

Student

Teacher

Studio Renter

Account 2 → 0x8d0...5fEb

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

http://localhost:3000

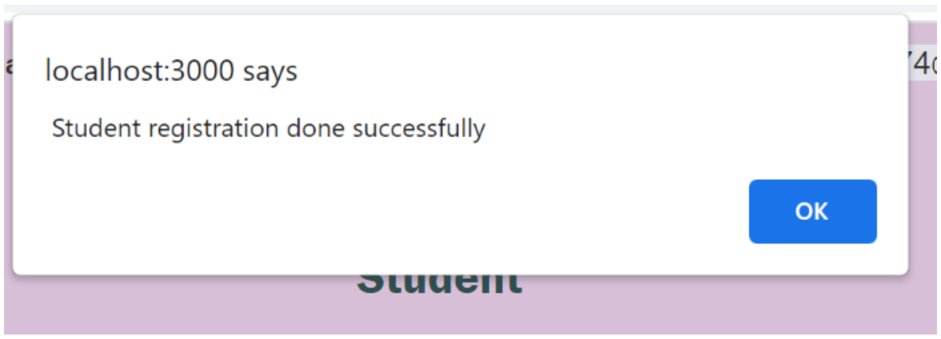
CONTRACT INTERACTION

Estimated gas fee 0.00319188 **0.00319188 ETH**

Max fee: 0.00319188 ETH

Total 0.00319188 **0.00319188 ETH**

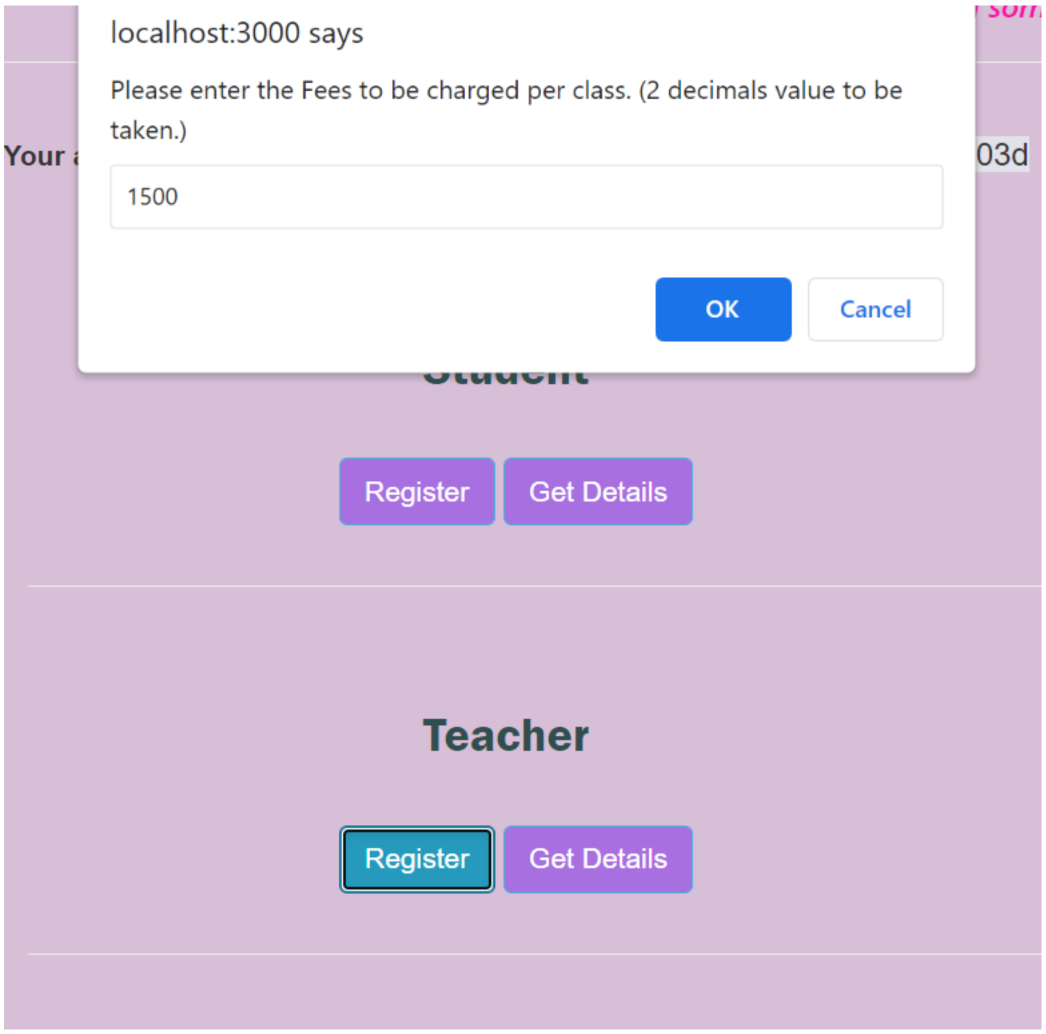
Amount + gas fee **Max amount:** 0.00319188 ETH



localhost:3000 says

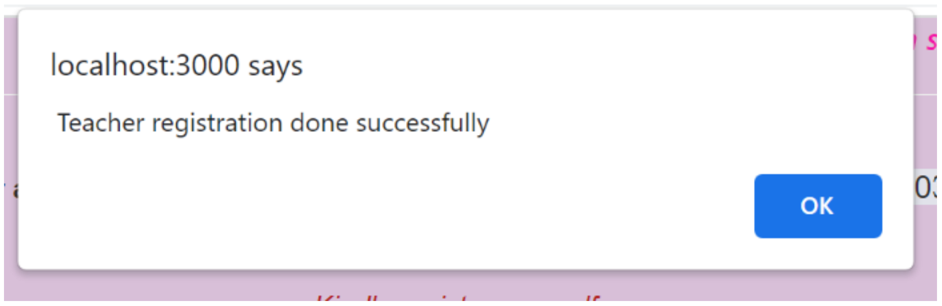
Student registration done successfully

2. **Teacher**
   Can register by entering amount of fees that will be charged to the student's taking their classes (considering 2 decimal places)


localhost:3000 says

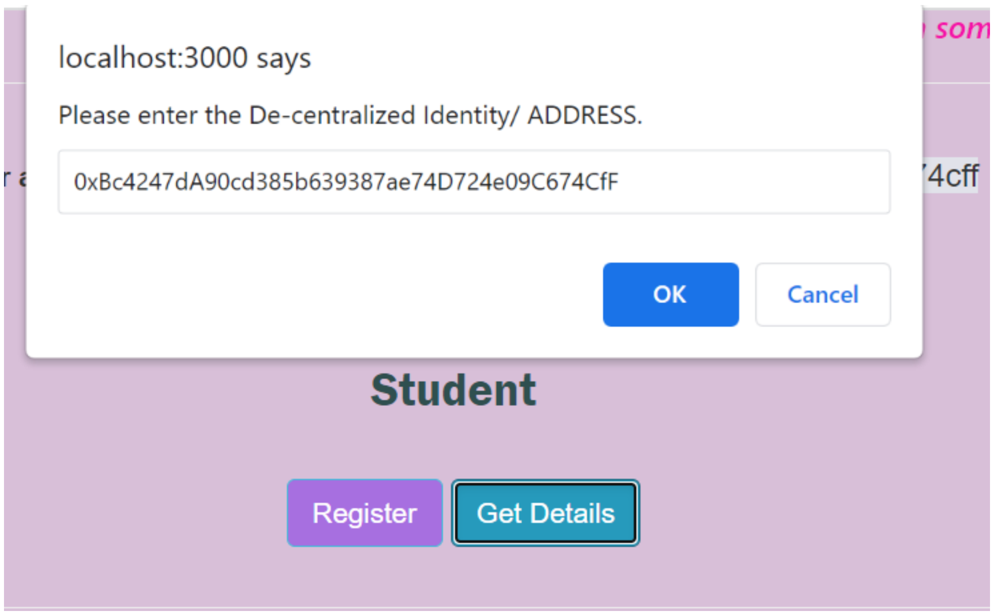
Please enter the Fees to be charged per class. (2 decimals value to be taken.)

1500


localhost:3000 says

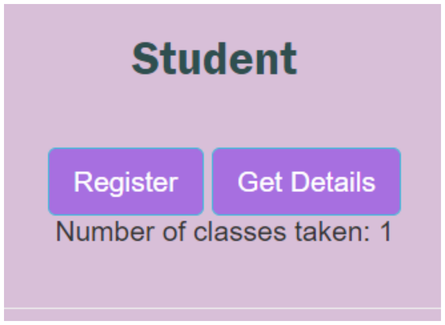
Teacher registration done successfully

getStudentDetails():student details can be fetched using their decentralized identity


localhost:3000 says

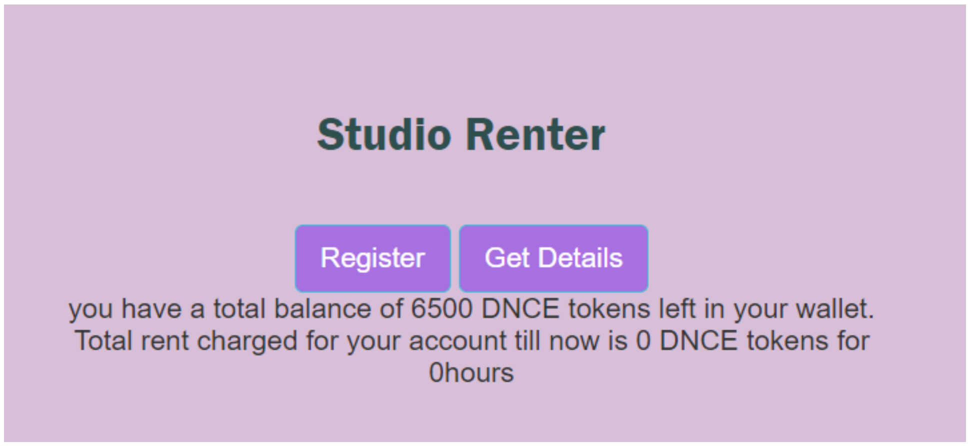
Please enter the De-centralized Identity/ ADDRESS.

0xBc4247dA90cd385b639387ae74D724e09C674CfF



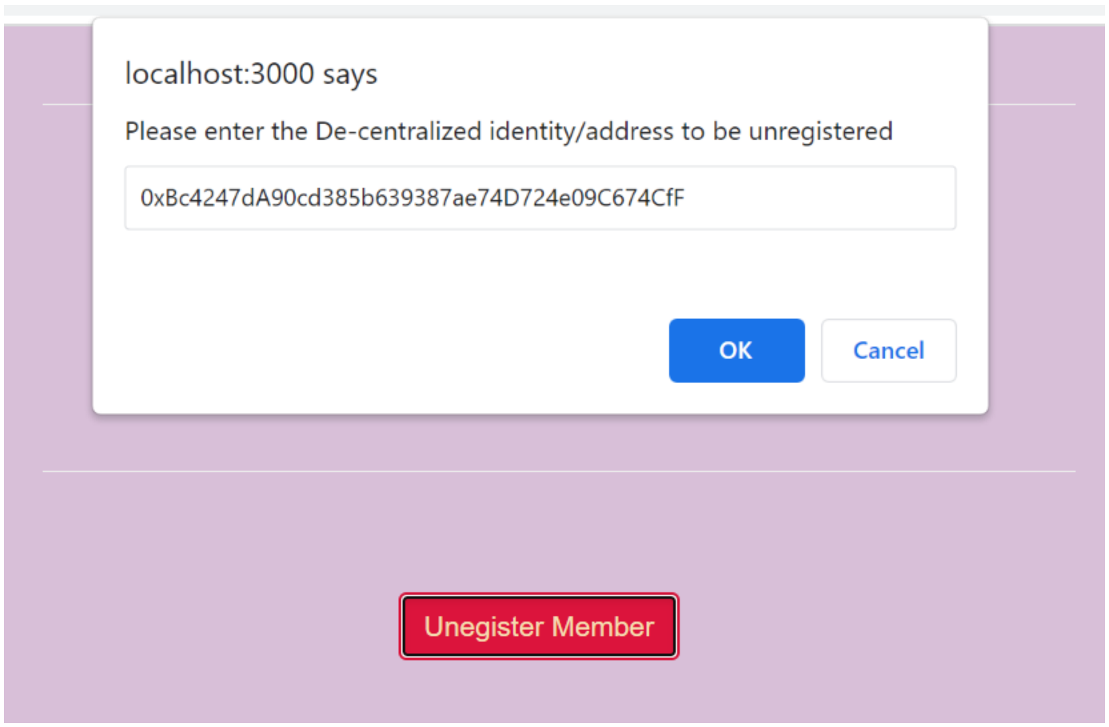
Number of classes taken: 1

getStudioRenterDetails(): wallet balance of the user can also be viewed using get details function



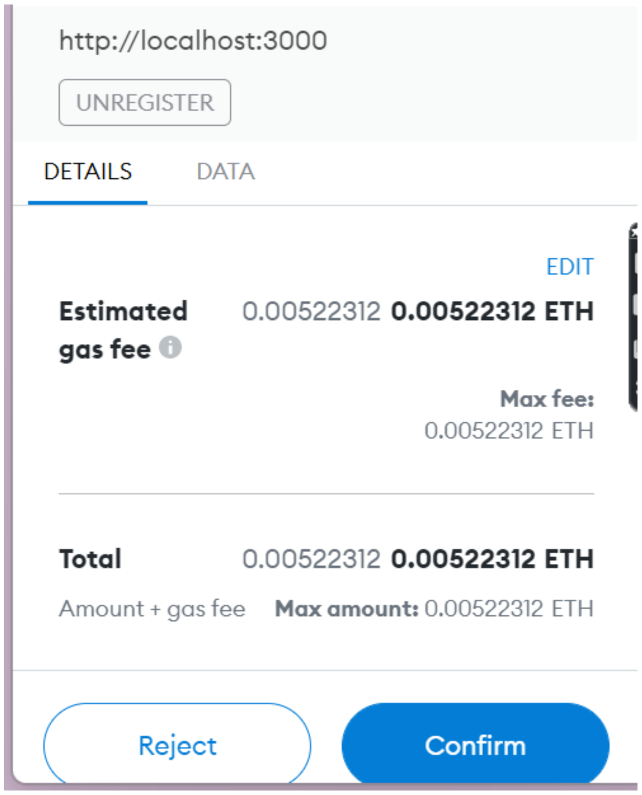
you have a total balance of 6500 DNCE tokens left in your wallet. Total rent charged for your account till now is 0 DNCE tokens for 0hours

unregister(): admin can unregister any member from the system by inputing their decentralized identity


localhost:3000 says

Please enter the De-centralized identity/address to be unregistered

0xBc4247dA90cd385b639387ae74D724e09C674CfF


http://localhost:3000

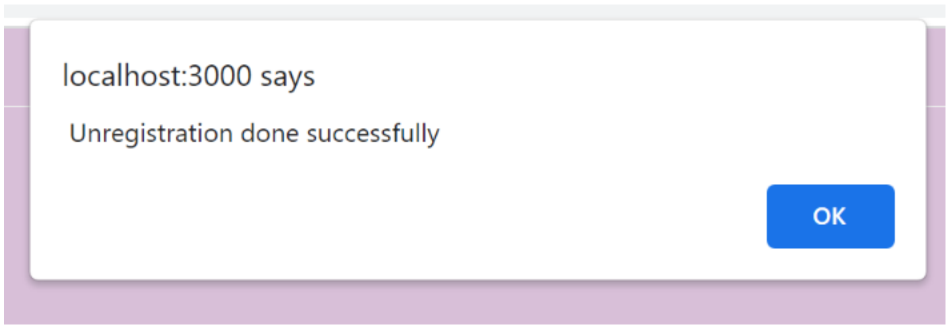
UNREGISTER

Estimated gas fee 0.00522312 **0.00522312 ETH**

Max fee: 0.00522312 ETH

Total 0.00522312 **0.00522312 ETH**

Amount + gas fee **Max amount:** 0.00522312 ETH


localhost:3000 says

Unregistration done successfully