Table of Contents

[Description 2](#_Toc533440705)

[Installation 3](#_Toc533440706)

[Deploy 3](#_Toc533440707)

[Working 4](#_Toc533440708)

[Components 4](#_Toc533440709)

[Admin 4](#_Toc533440710)

[School Admin 6](#_Toc533440711)

[Student 7](#_Toc533440712)

[CDN 9](#_Toc533440713)

# Description

The project aim is to provide a platform for schools to store transcripts and make it available to students and employers using blockchain which makes sure that the transcripts are genuine and not counterfeit.

The ERC721 token stores the student details like ID,DOB,first name,Hash,IPFS url and last name as meta data which can be later retrieved to access the transcript.

Currently the token is transferred from school admin to student but this can be modified easily using myApprove function in MyERC721.sol

The UI has made with help of Bootstrap and css elements.

**Challenges** faced:

1. To upload the transcript to ipfs without installing a node on the user pc
2. Saving and retrieving more details on the ERC721 Meta data token\_URI

**Improvements** over the one presented on 12/17/2018

1. During the presentation only one student and college could be executed – Now the students and schools can be expanded. Each student is tied to a school address which make it easy to track.
2. Before there was no verification by the school but now student needs to register and be verified by the school to login.
3. The meta data stored in the erc721 was expanded to not only store hash but only link to IPFS as well as student details
4. Student after login can now see not only hash but also the transcript image from IPFS , token No. where the URL of IPFS and token no were pulled from the meta data of the students token.
5. The UI was upgraded vastly using bootstrap and google material UI in the components like:
   1. Navbar
   2. Back ground image
   3. Buttons
   4. Forms
   5. Flow
6. More CDN’s are used instead of libraries required in the client machine to make it more portable.

# Installation

1. We need to install node js –  
   Download node.js package from <https://nodejs.org/en/download/> and install OR

npm install npm@latest -g

To check if installed npm -v OR node -v

1. We need to install ganche and truffle

npm install –g truffle  
npm install ‐g ganache-cli

Since My port in truffle.js is 7545 - I am using ganache Ui so install from the website <https://truffleframework.com/ganache> or else change the port accordingly for ganache-cli eg:8545

1. Initialize Truffle in the project directory

truffle init

npm init -y

1. Create the package.json

npm init -y

1. Edit the truffle.js file OR Copy the truffle.js to root folder which has set the compiler version to 0.4.24+commit.e67f0147.Emscripten.clang
2. Add the sol file MyERC721.sol under the contract folder in the root directory
3. Copy 2\_deploy\_contracts to the migrations folder under the root directory. Edit migrations folder to match the owner address which is first address of ganache in this case.
4. We need to import openzeppelin-solidity for ERC721 standard files.

npm install openzeppelin-solidity

1. Copy and replace the ERC721 folder under node\_modules\openzeppelin-solidity as it has been modified to suite the needs of this project
2. Our project doesn’t require meta mask
3. Web Server : node http server or Apache using xampp is used. Install xampp
4. Copy the font end files to C:\\xampp\htdocs\$Project\_name

# Deploy

1. Compile

truffle compile

1. Keep the ganache running
2. Migrate/Deploy

truffle migrate --reset

1. Copy the the 2\_deploy\_contracts.js contract address and paste into the index,js of the front end
2. Copy the abi from $Truffle\_Project\build\contracts\MyERC721.json to index.js
3. Now open localhost in the browser and go to the folder which will run the index.html of the front end.
4. Copy the required addresses from ganache in small case
5. Copy private key of the owner and schools selected from ganache in separate files say private.txt and private1.txt

# Working

Note: All addresses are in lower case.

1. Go to Admin page and login using the contract owner – Create wallet using private key say private.txt
2. Invite a school to participate in this program by adding the address of the school.
3. Go to the school page and login with the address. You will be asked to create a wallet for signing of transactions.
4. Upload an image to ipfs to get the link.
5. Enter the details for the meta data to be tied with the token and submit.
6. Go to student page and enter the address .
7. If not registered register with details.
8. Go to college page and login.
9. Verify the student which appears at the bottom of the page in a table.
10. Go to Student page and login with address which was verified .
11. Click give me transcript to get the image from ipfs, transaction hash of the token transfer and token ID.
12. The no of schools and students and be expanded and happens dynamically.

# Components

## Admin

The admin utilizes around two pages admin.html and admin\_Internal.html. The admin of the site is the owner of the contract. We login with this address which is the first address of ganache in this project. It uses myGetChairPerson function in solidity.

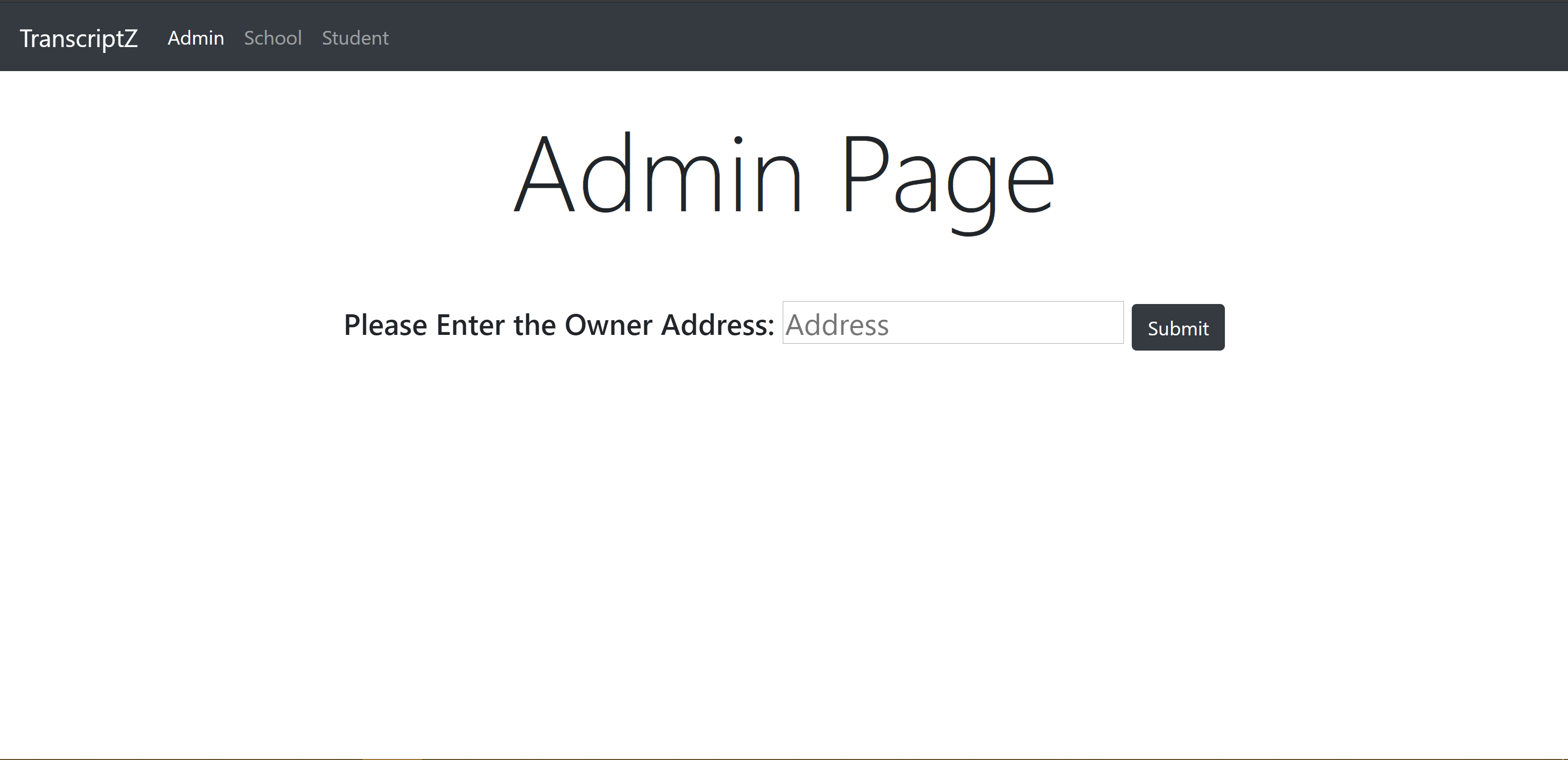
The admin has to create a wallet to sign the transactions. The wallet requires the private key and password to encrypt it. It stores it in the sessionstorage as per the documentation of web3js 1.0.

The admin has the function to invite a school address by calling the myIsActive function which sets the Boolean value.

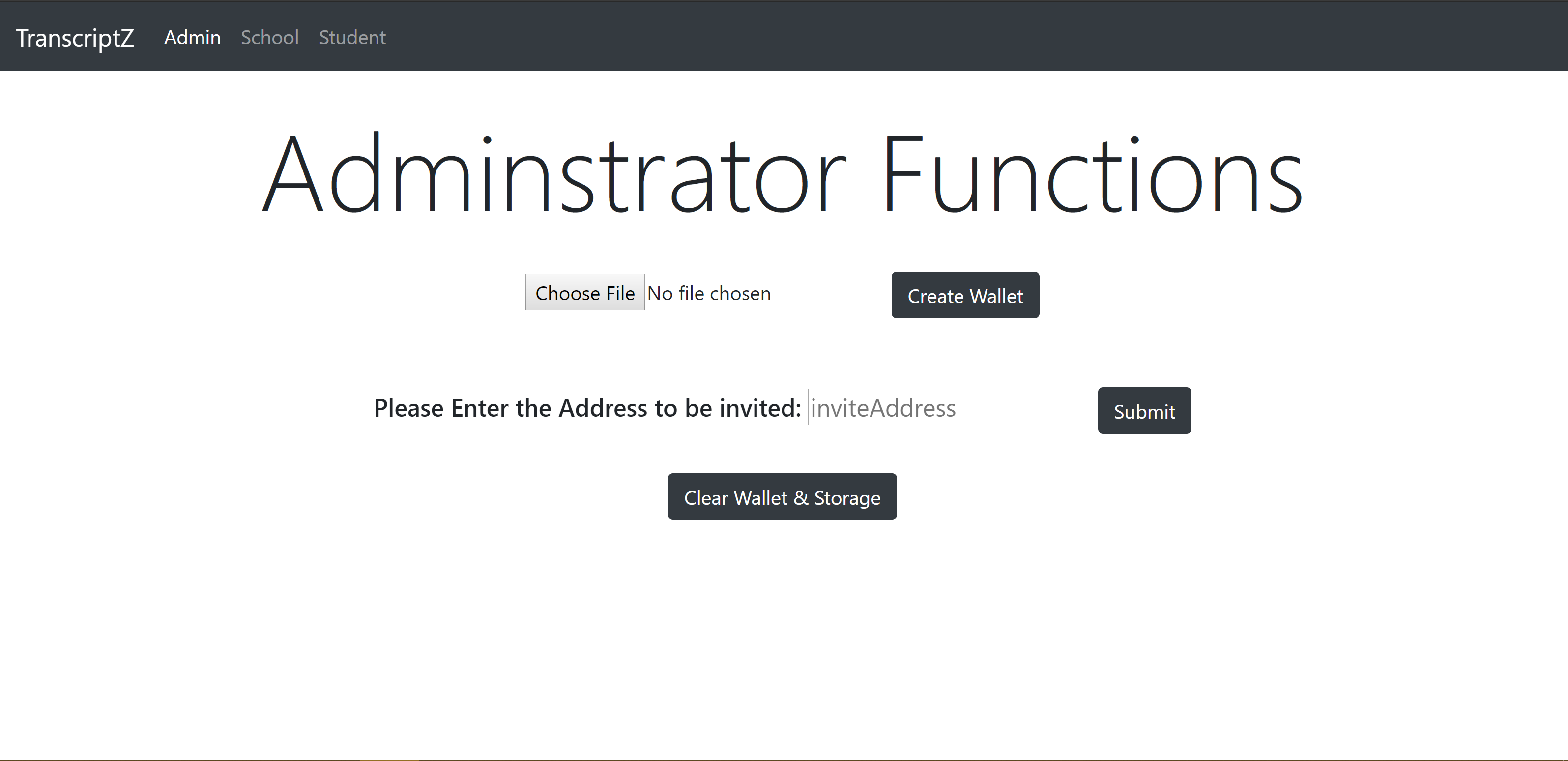
The clear All button clear the local storage, session storage and global variables.



**Index.html**



**Admin.html**



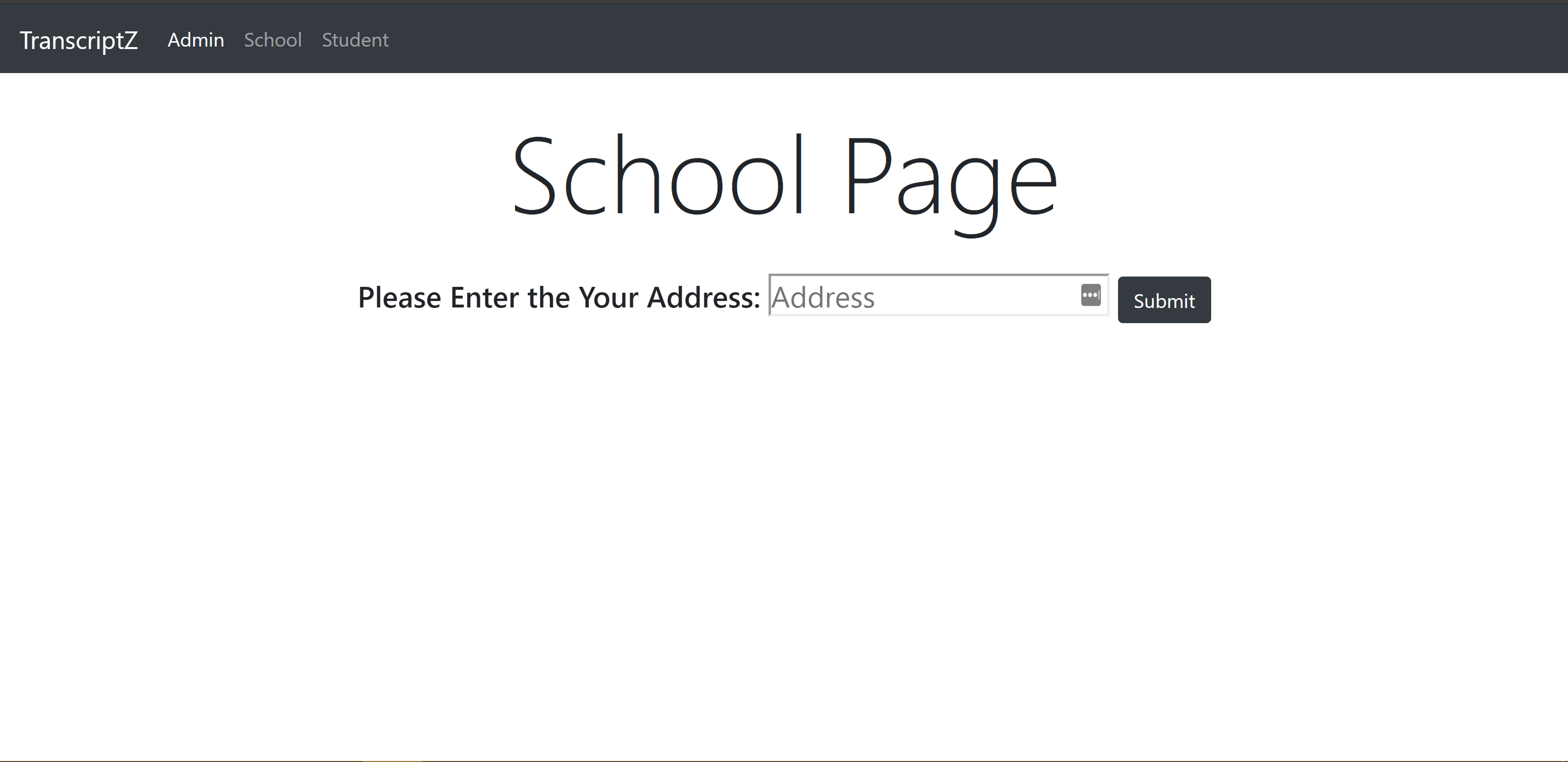
**Admin\_Internal.html**

## School Admin

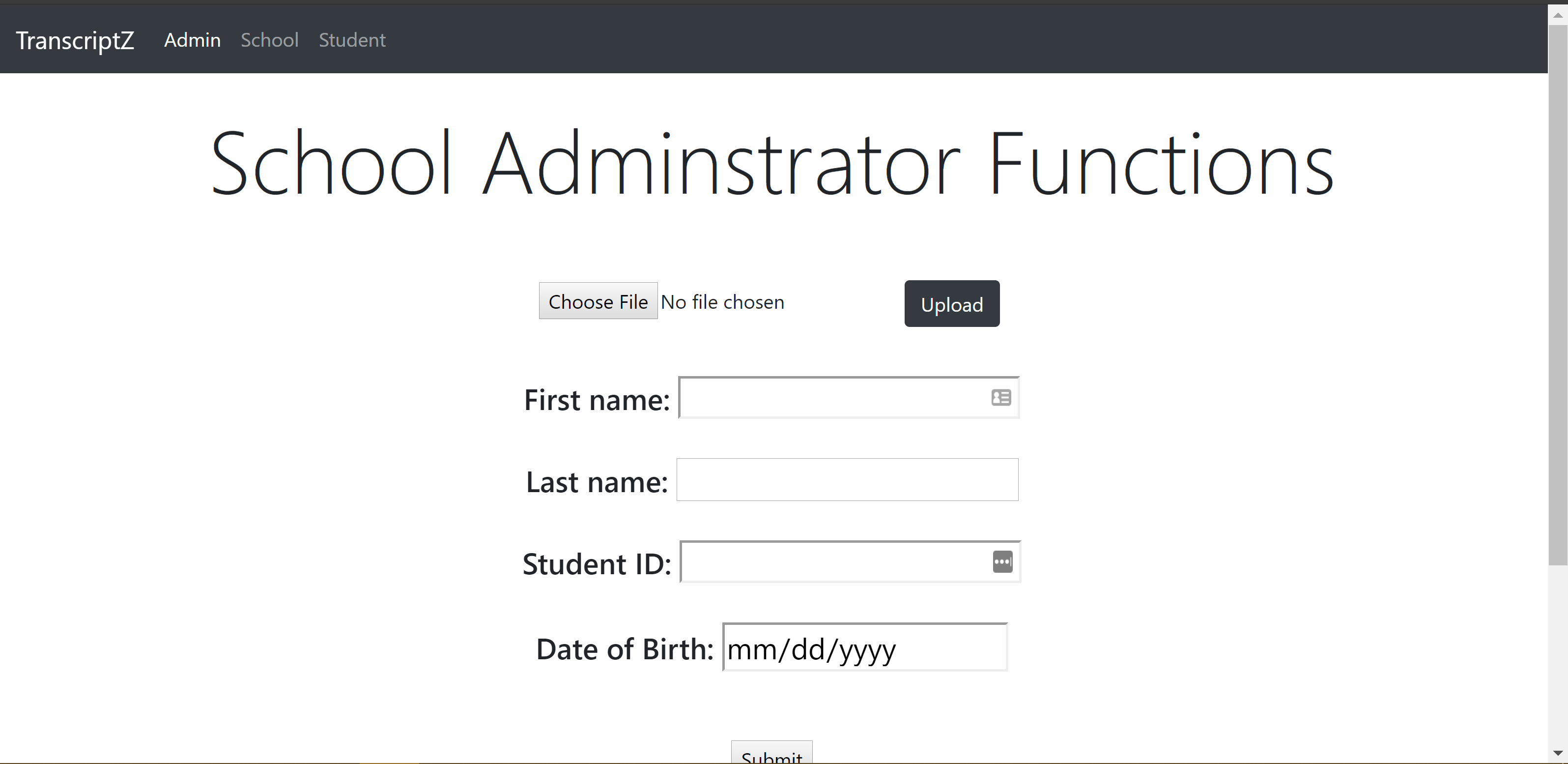
The school utilizes around two pages college.html and college\_Internal.html. The school admin must be first invited which calls to myGetBool to check it. The school admin also must create a wallet to sign the transactions.

The school Admin has ability to upload the file to IPFS and upload the meta data while minting the coin.

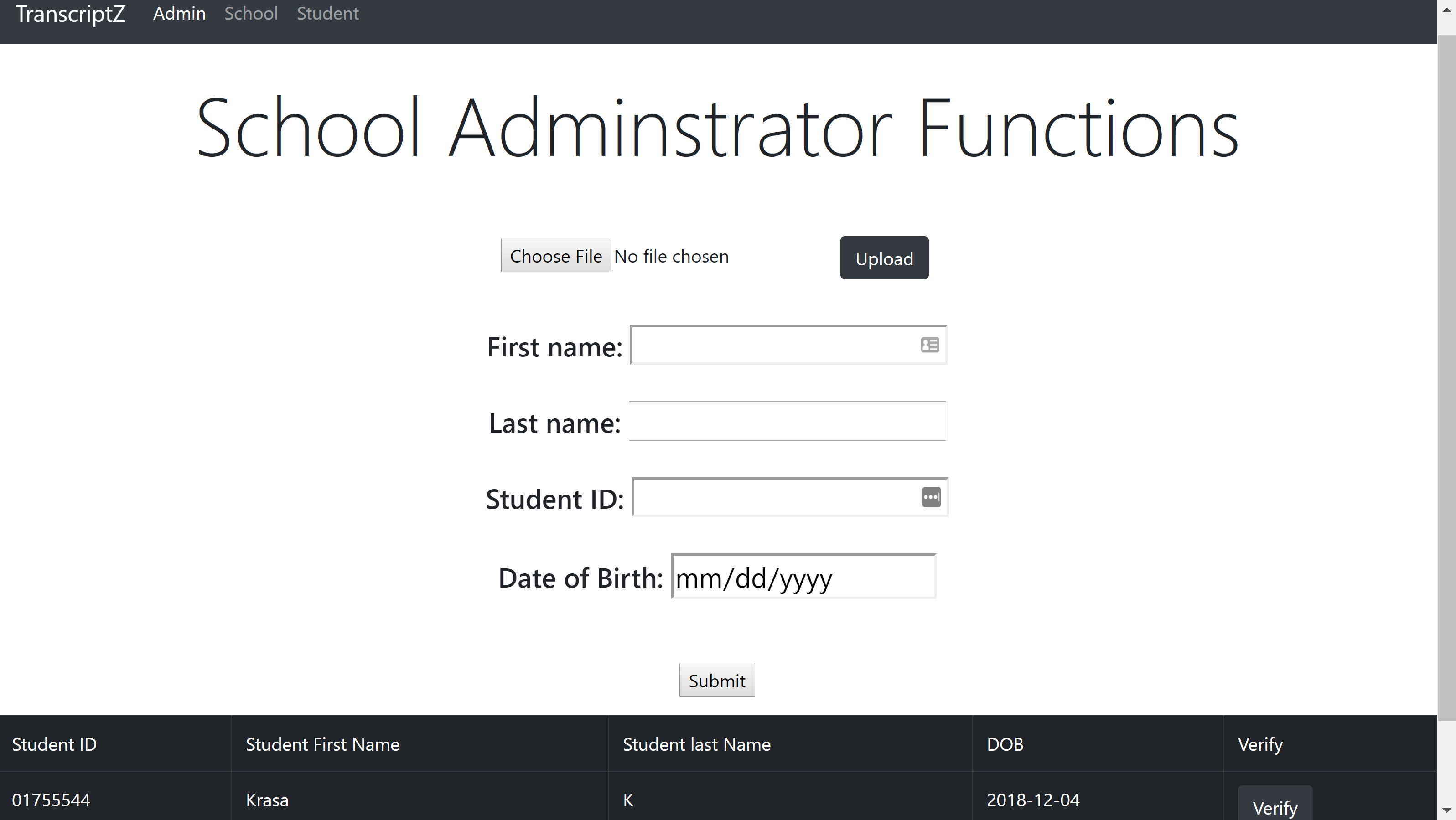
When the student registers the student appears in the school admin page which can verify and transfer the owner ship of token to the student by the calling the MyTransferFrom function in MyERC721.sol. To transfer the token the owner must submit the request.



**College.html**



**College\_Internal.html**

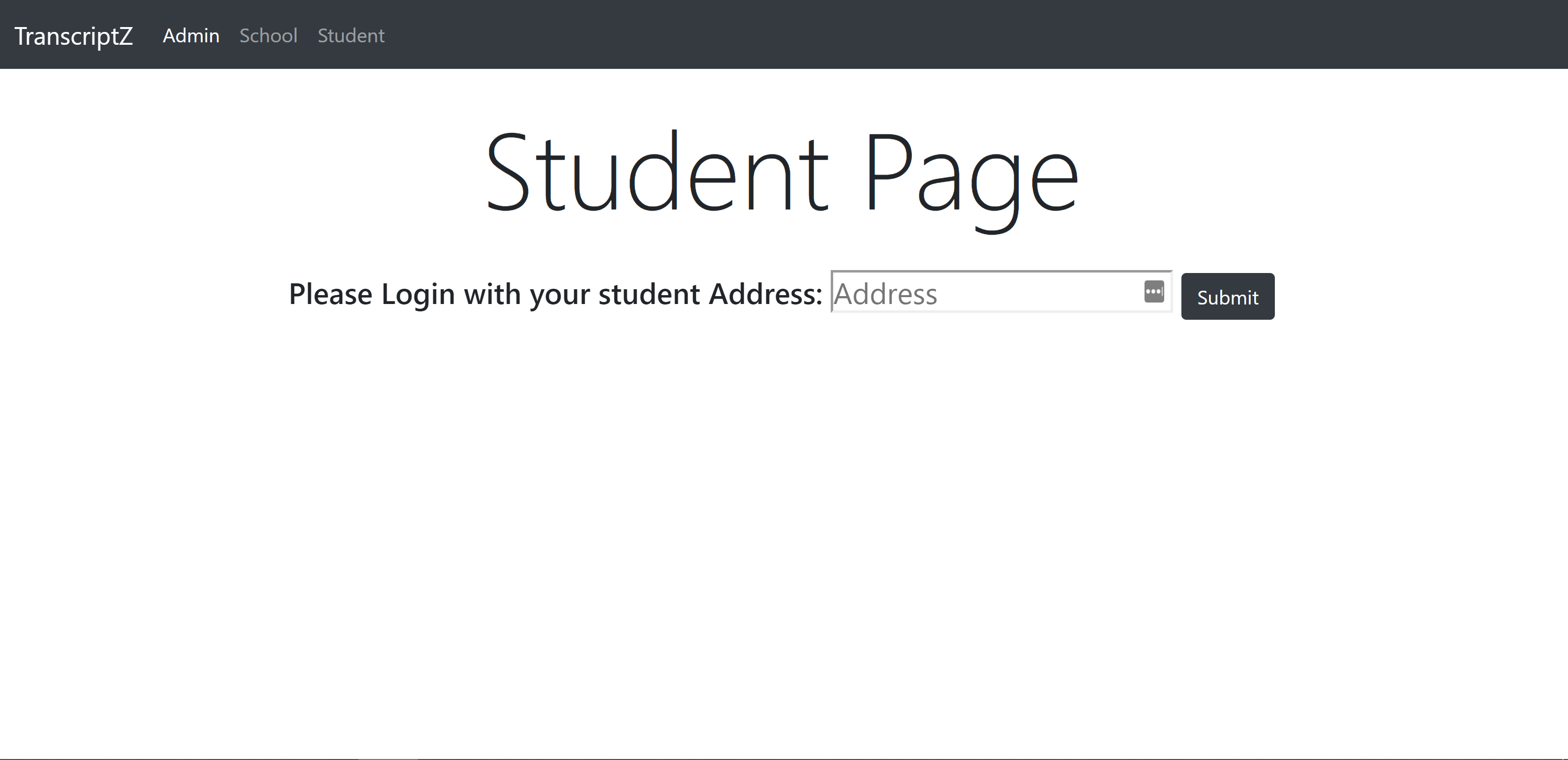


**Verification process in college\_internal.html**

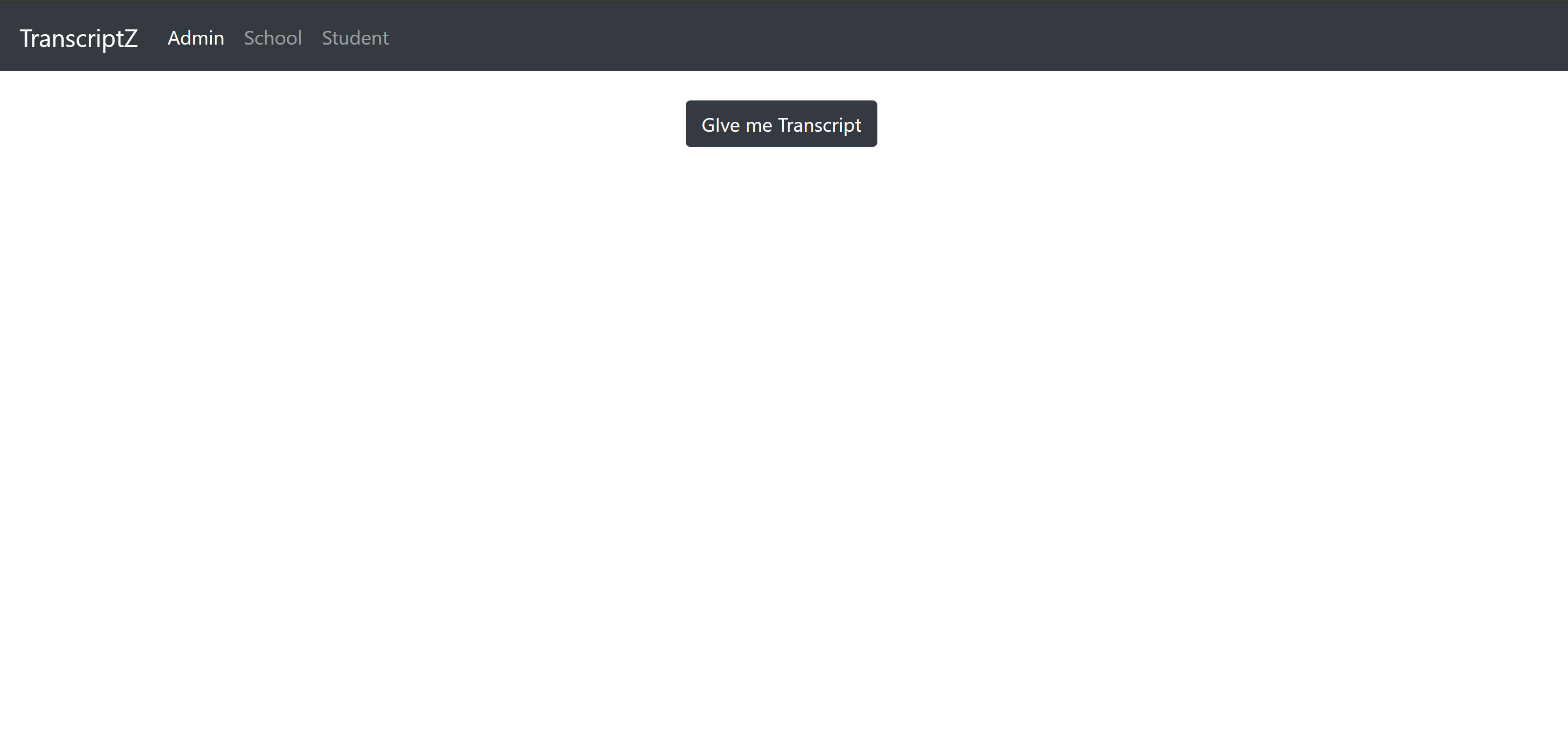
## Student

The student has the ability to register with Id, name, dob and login using its address after the verification by the school admin

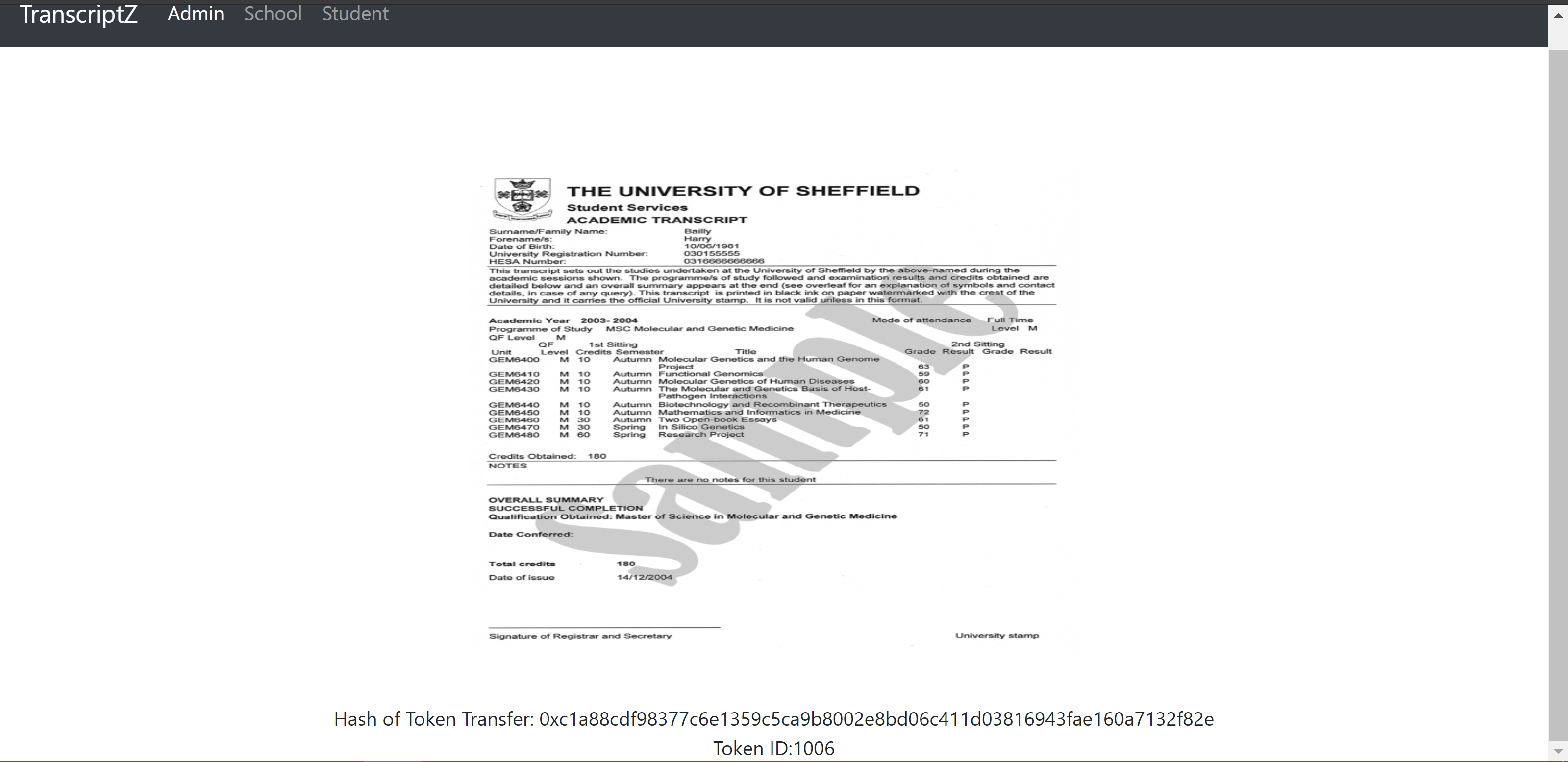
The student can see the transcript as well the token id and the hash of the transaction.



**student.html**



**student\_Internal.html**



**student\_Internal.html with transcript**

To change the ownership of the contract we need to inherit Ownable.sol

# CDN

In this project we use mainly CDN sources to avoid dependency of client-based libraries as possible.

The CDN’s used here are:

<script src="https://code.**jquery**.com/jquery-3.3.1.js" integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60=" crossorigin="anonymous"></script>

<script src="https://**ajax.googleapis**.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js"></script>

<script src="https://**cdnjs.cloudflare**.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>

<script src="https://stackpath.**bootstrapcdn**.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/gh/ethereum/**web3.js@1.0.0-beta.36**/dist/web3.min.js" integrity="sha256-nWBTbvxhJgjslRyuAKJHK+XcZPlCnmIAAMixz6EefVk=" crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/**ipfs**/dist/index.js"></script>

<script src="https://wzrd.in/standalone/**buffer**"></script>