Information Security

Project Report

Title: Academic Application Tracker

Members: 19K-1057 Muhammad Khizer Jilani

19K-1116 Sarmad Jamal

19K-1114 Mansoor Butt

Section: SE-A

Submitted to: Fahad Samad

Dated: 10-Dec-2022

Overview

In this project we have made an academic Application Tracker. The System allows the student of a University to upload his/her application related to any University Issue as soon as the student uploads the application on our portal the application is stored on the IPFS. IPFS is a distributed peer to peer file system just like torrent. As the application is uploaded on the IPFS, it will generate a unique hash, this hash will be stored on our Smart Contract, the Smart Contract will be deployed on a blockchain network. The reason to use Smart Contracts and Blockchain in this project is to make our System more Secure, also to allow a Consensus Mechanism which will be implemented in the later process.

The other concept we are using is RBAC, where Teacher, HOD and Director will have a separate portal and all of them will have the access of the same application uploaded by the student. All 3 stakeholders' approvals are required to approve the application depending on the nature of application means how critical it is.

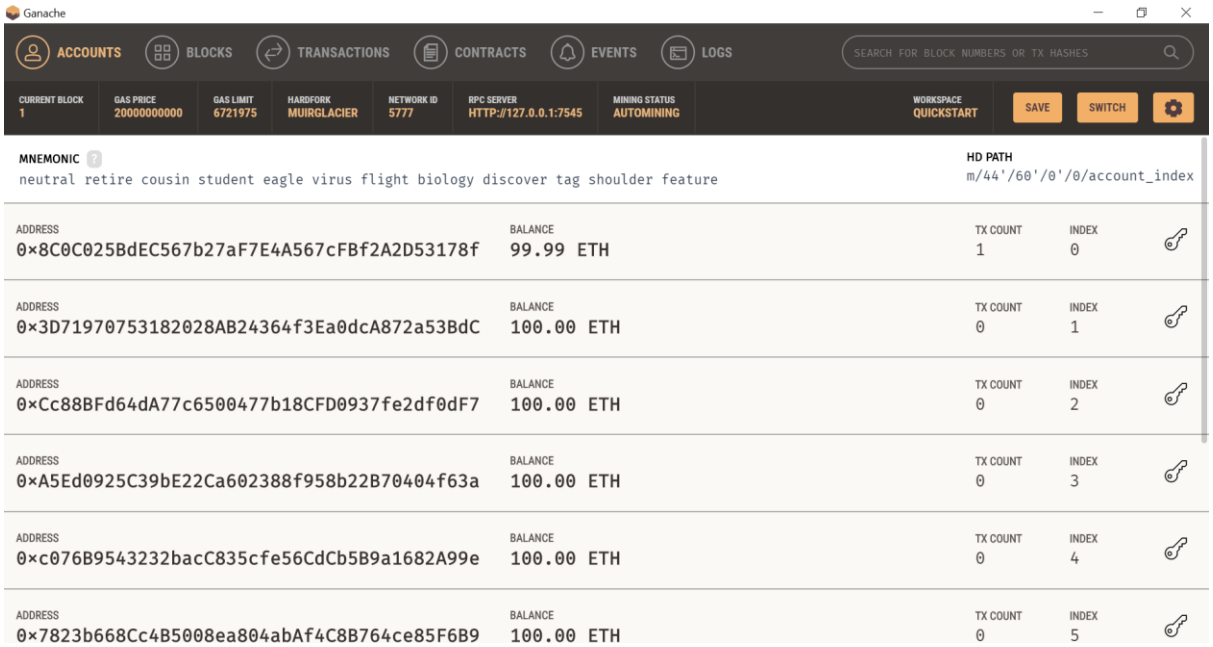
All the stakeholders have their own digital signatures, this feature will ensure that no unauthorized entity can access the application or perform any kind of operations which are not authorized for it to perform.

Role Based Access Control

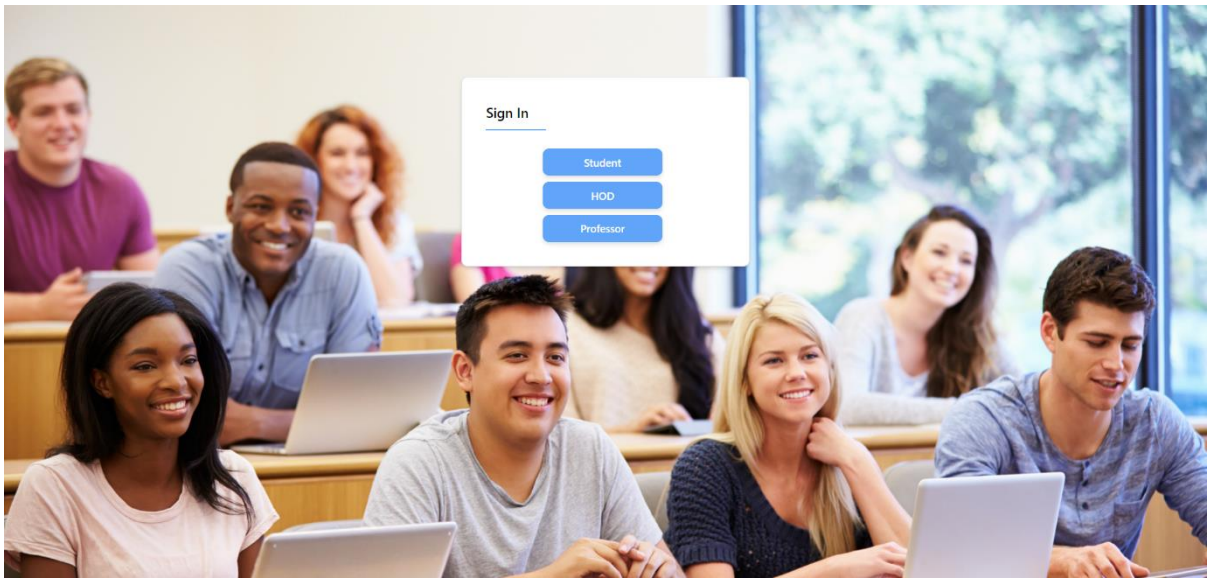
Step 1:

we are using Ganache which is a software which provides a test RPC, the RPC is connected to Ganache's locally deployed blockchain

network also it provides us with 10 registered addresses along with their private keys. We are using the first three accounts addresses which will be assigned to the Professors and HOD

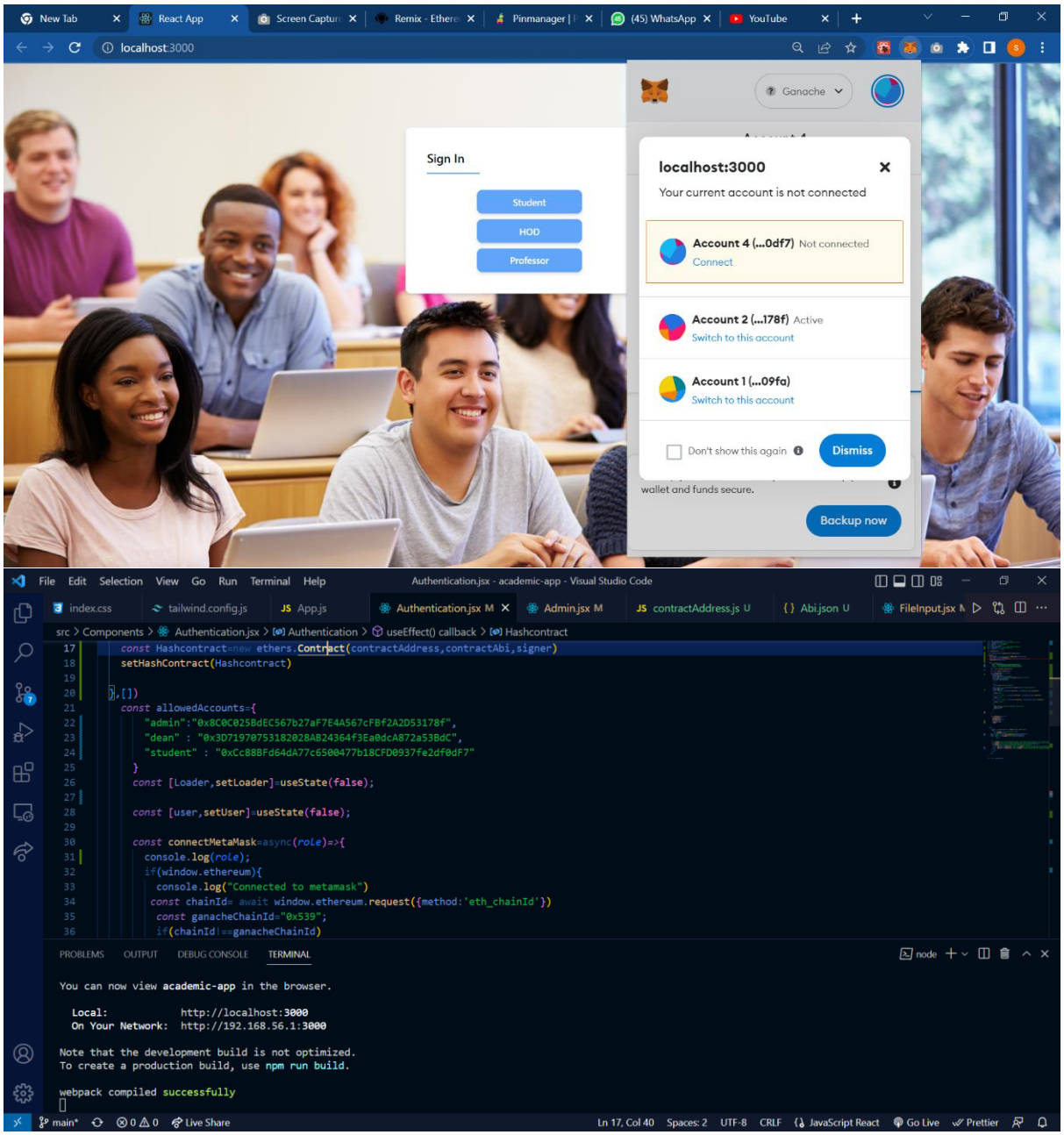


Portal



Step 2

We are authenticating login accounts with MetaMask. MeteMask is a digital wallet which provides every user with their custom private keys and hash addresses

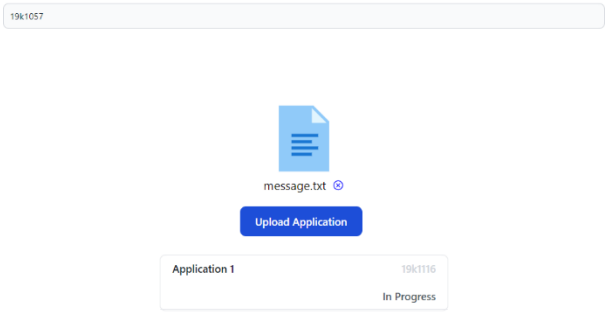


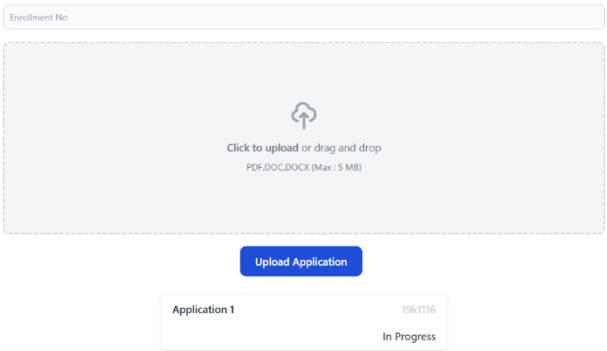
Step 3:

Student logins via the portal the MetaMask verifies the student by communicating with the smart contract

Step 4:

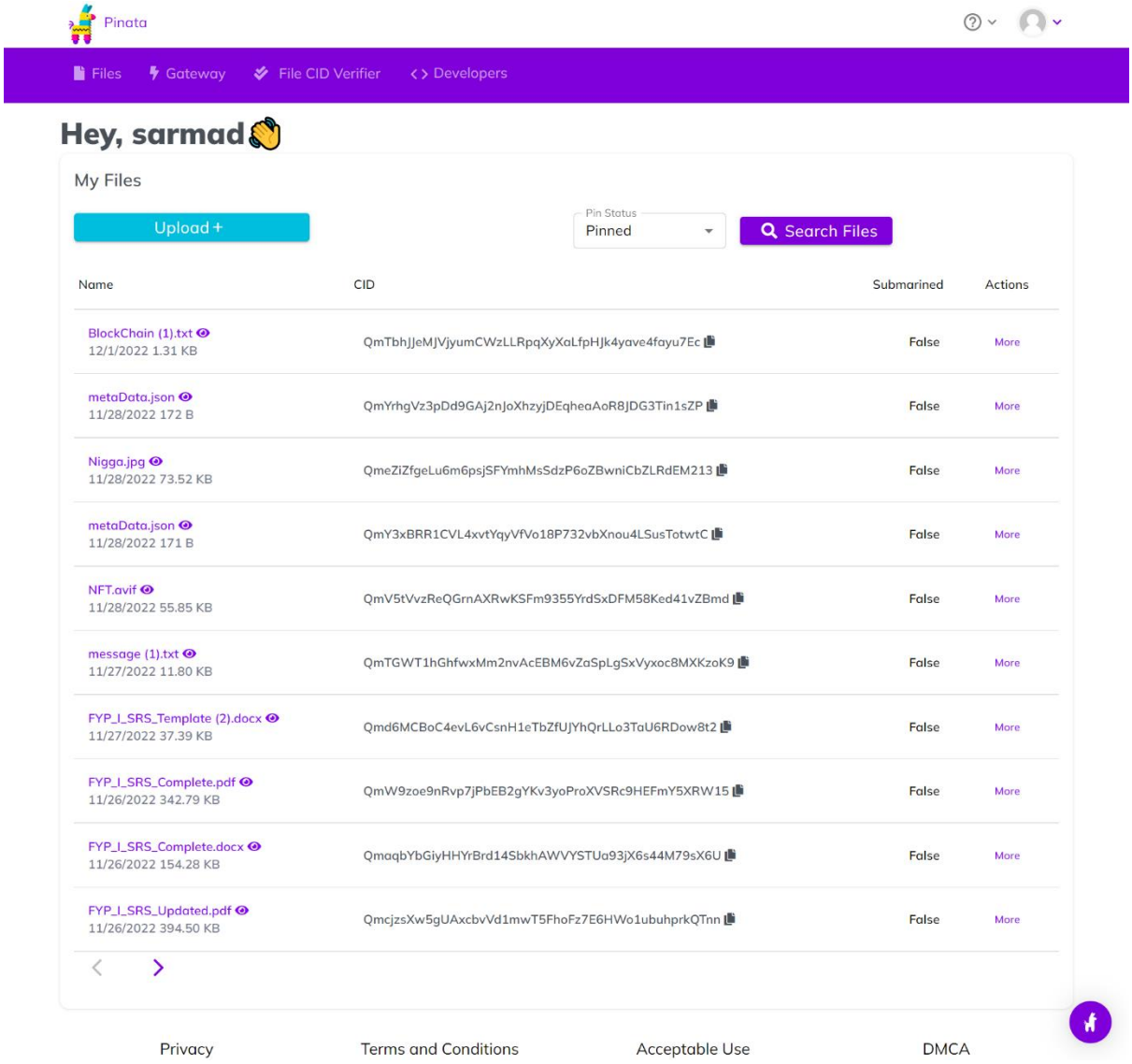
Student uploads the application on the portal





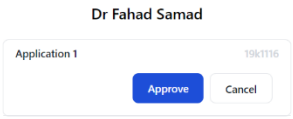
Step 5:

The Application will be stored on Pinata which is an IPFS hosting platform also we can access hash of that Application

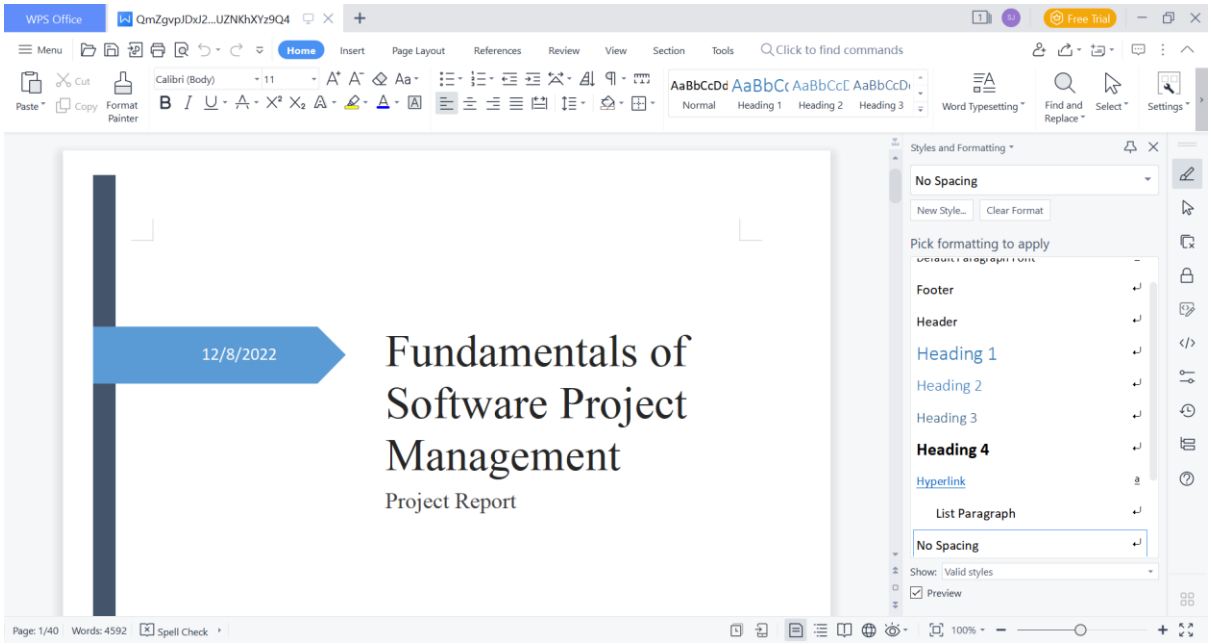


Step 6:

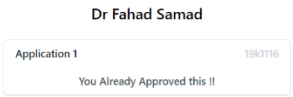
Now the Application will be available to be viewed for all stakeholders including Professors and HOD on their relevant portal.



Now the Professor Doctor Fahad Samad will view the application by clicking on the tag Application 1 and the file will be downloaded



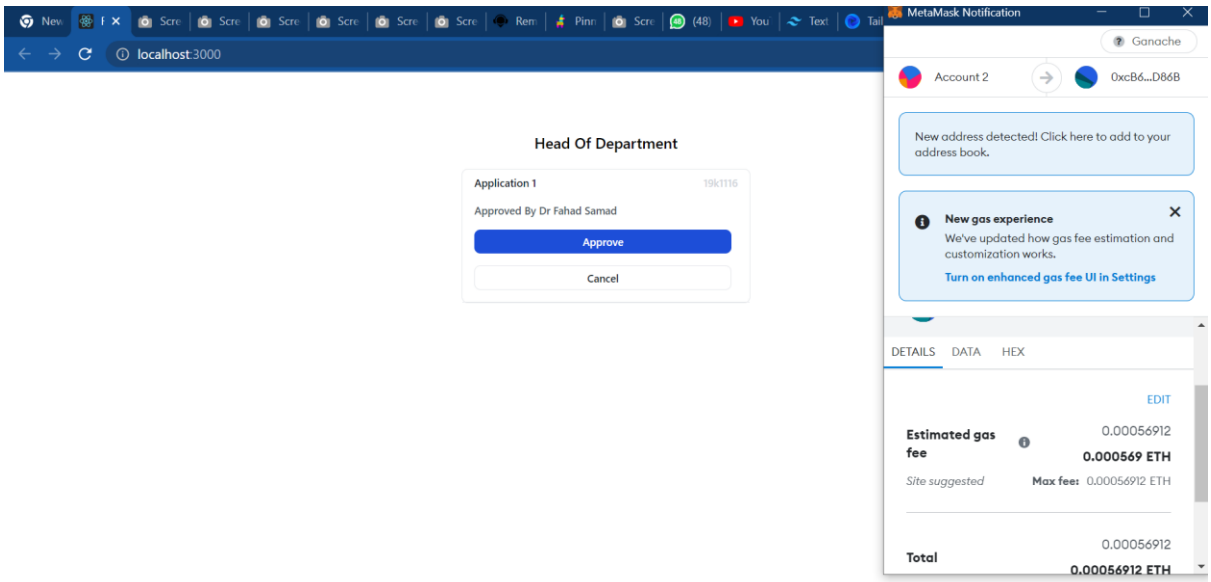
For instance, we have uploaded this document from student profile and the file is now downloaded and can be viewed



Dr Fahad Samad approved this application as we can see

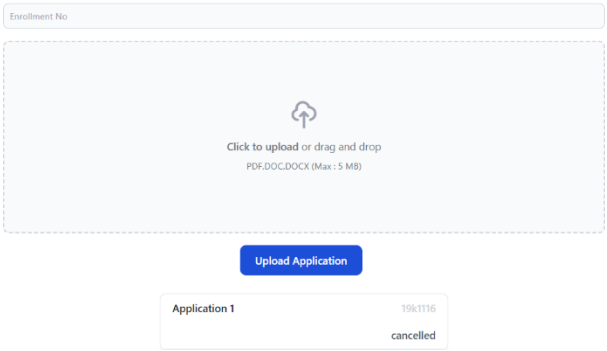
Step 7:

Now all the stakeholders will either approve or reject the application, the application will only be accepted if all the stakeholders approve it else it will be rejected



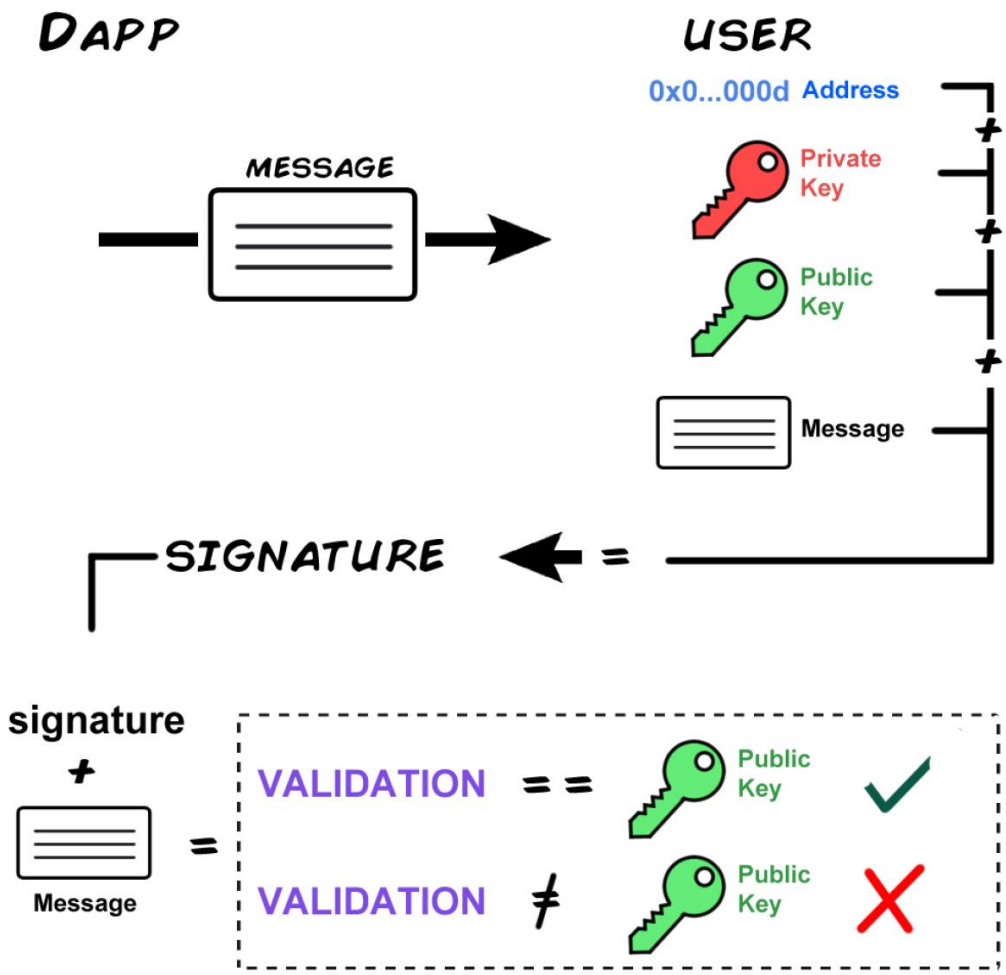
Step 8:

The student on its portal can track the status of the application that whether It was accepted or rejected.



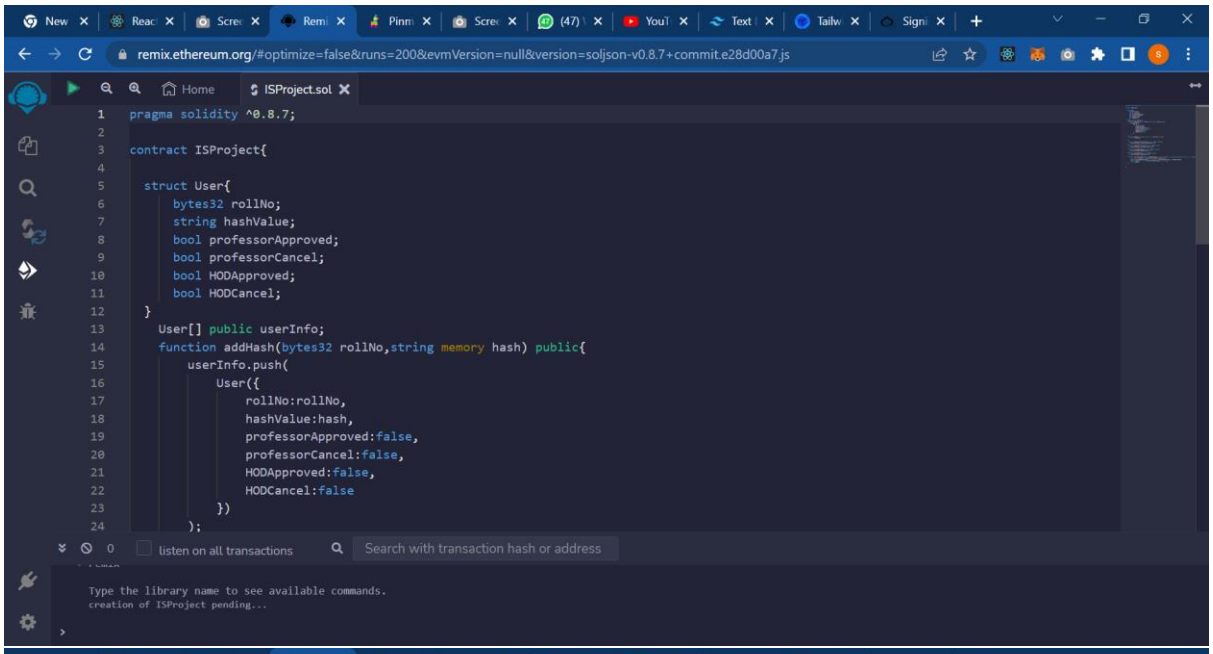
Digital Signature

Here we are encrypting our digital account with MetaMask private key to ensure strong security mechanism every individual user get its private key via MetaMask which ensures the authentication helps in creating the digital signature of individual User so this mechanism ensures that sender can't set stance of denial on sent message.



For instance as define in this picture we are taking private key of user and a message which we are sending to our smart contract and it is returning a public key and we are comparing MetaMask public

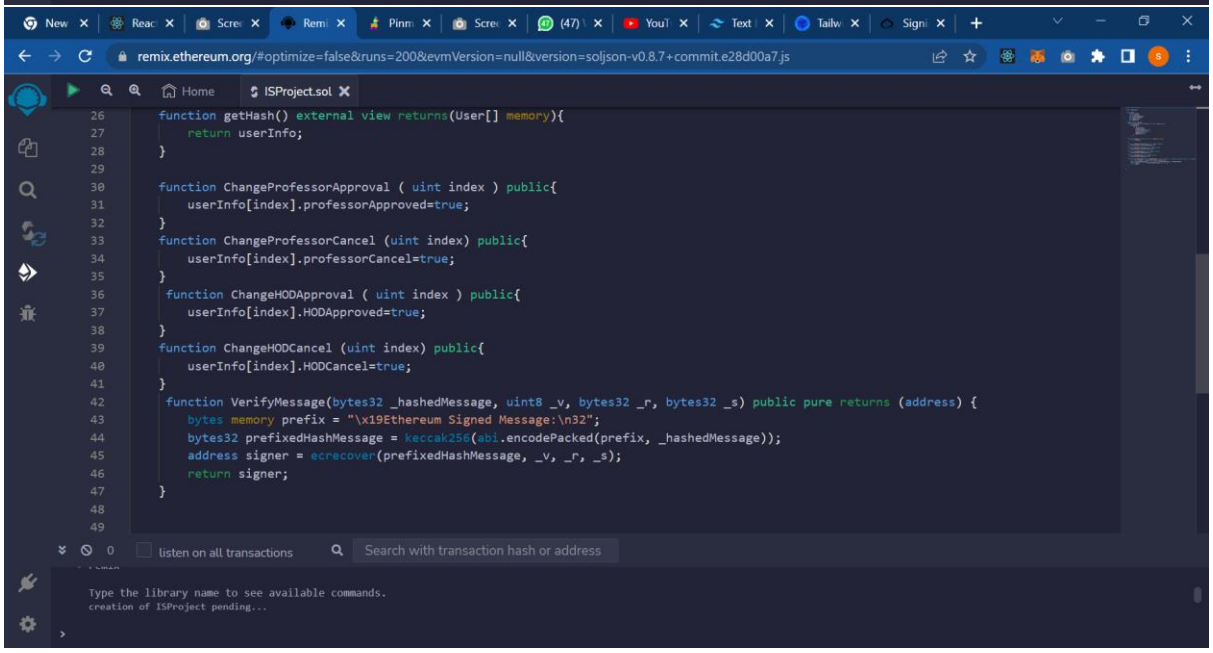
address of the user account with the return value if this condition satisfies we are allowing user to approve and cancel our application




```
pragma solidity ^0.8.7;

contract ISProject{

  struct User{
      bytes32 rollNo;
      string hashValue;
      bool professorApproved;
      bool professorCancel;
      bool HODApproved;
      bool HODCancel;
  }
    User[] public userInfo;
    function addHash(bytes32 rollNo,string memory hash) public{
        userInfo.push(
            User({
                rollNo:rollNo,
                hashValue:hash,
                professorApproved:false,
                professorCancel:false,
                HODApproved:false,
                HODCancel:false
            })
        );
```




```
    function getHash() external view returns(User[] memory){
        return userInfo;
    }

    function ChangeProfessorApproval ( uint index ) public{
        userInfo[index].professorApproved=true;
    }
    function ChangeProfessorCancel (uint index) public{
        userInfo[index].professorCancel=true;
    }
     function ChangeHODApproval ( uint index ) public{
        userInfo[index].HODApproved=true;
    }
    function ChangeHODCancel (uint index) public{
        userInfo[index].HODCancel=true;
    }
     function VerifyMessage(bytes32 _hashedMessage, uint8 _v, bytes32 _r, bytes32 _s) public pure returns (address) {
        bytes memory prefix = "\x19Ethereum Signed Message:\n32";
        bytes32 prefixedHashMessage = keccak256(abi.encodePacked(prefix, _hashedMessage));
        address signer = ecrecover(prefixedHashMessage, _v, _r, _s);
        return signer;
    }
```

Acronyms

IPFS = interplanetary File System

RBAC = Role Based Access Control

HOD = Head of Department

RPC = Remote Procedure Call