ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Các công nghệ mới trong phát triển phần mềm

Niên khóa 2021 – 2022 (HK2)

Blockchain

Giáo viên hướng dẫn:

Nguyễn Lê Hoàng Dũng

Ngô Ngọc Đăng Khoa

Trần Văn Quý

Sinh viên thực thiện:

18120362 - Trần Thanh Hiền

Mục lục

Tài liệu ứng dụng MyCoin – Blockchain

1 Thông tin cá nhân

Họ và tên: Trần Thanh Hiền

MSSV: 18120362

Sdt: 0934917187

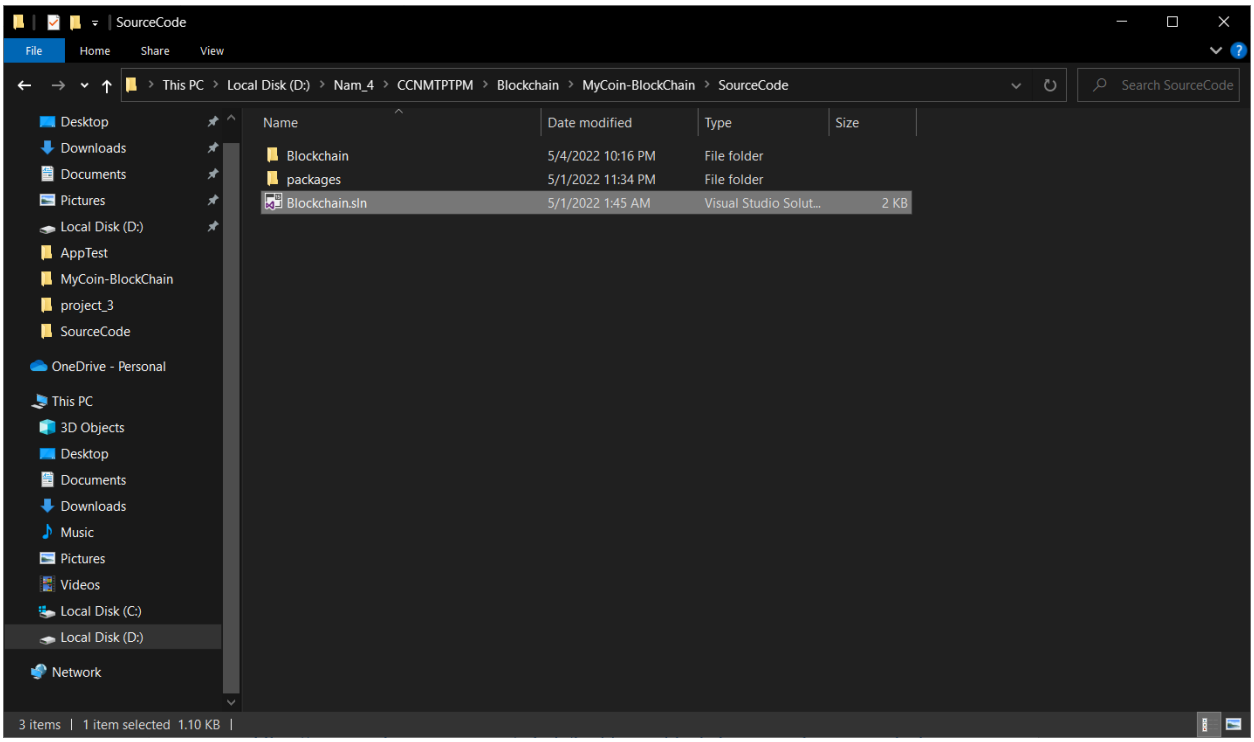
Mail liên lạc: tranthanhhien2017.kh@gmail.com hoặc 18120362@student.hcmus.edu.vn

Link Github Project: https://github.com/hientranHT/MyCoin-BlockChain.git

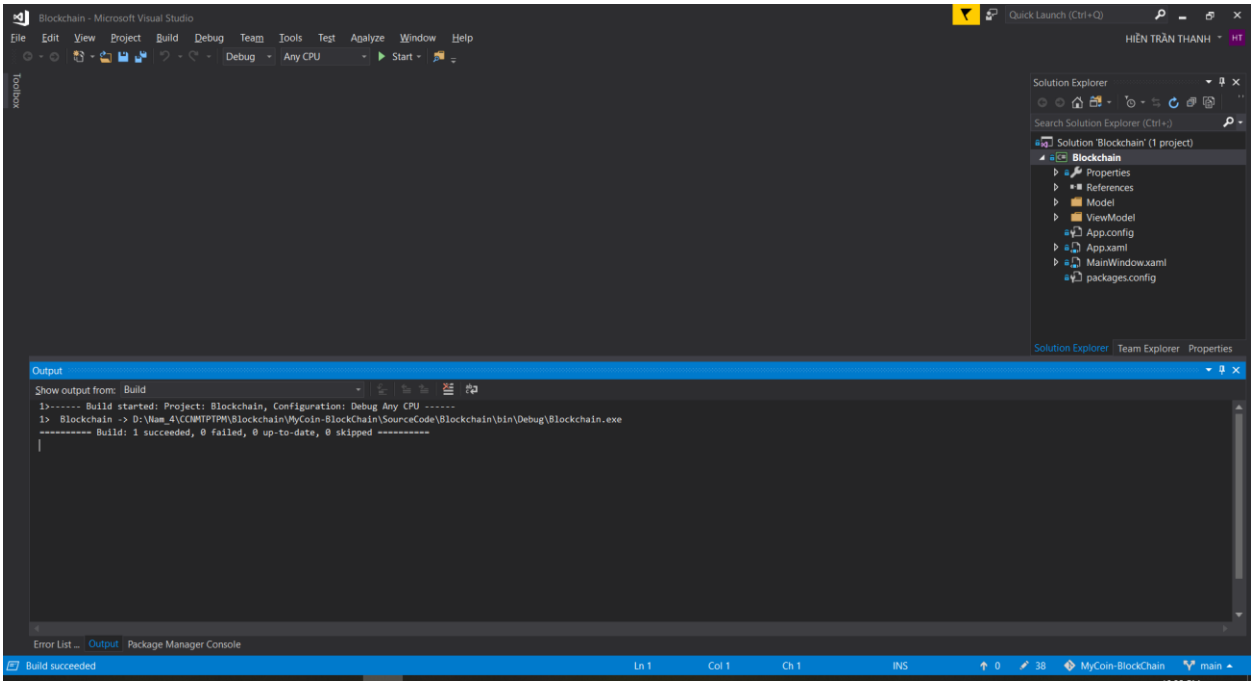
Link Video Project: https://youtu.be/93OVl34NXYc

2 Hướng dẫn cài đặt

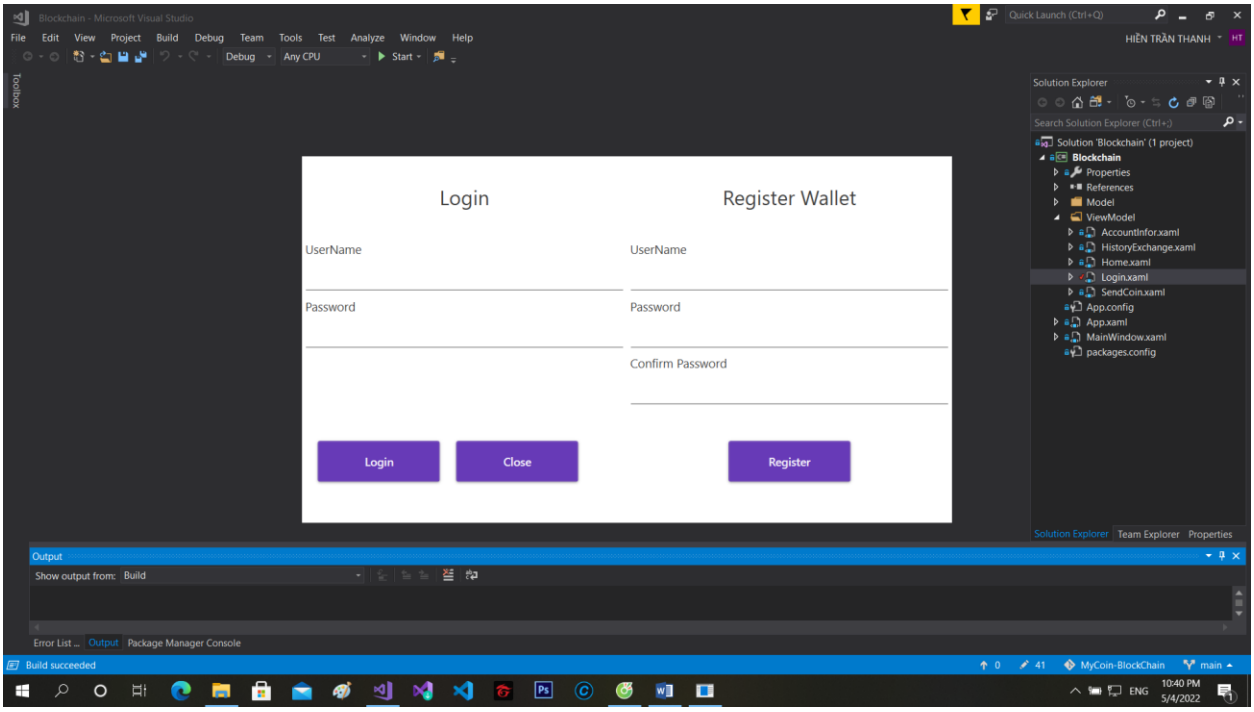
1. Cài đặt Visual Studio của Microsoft tại link trang chủ của Microsoft (https://visualstudio.microsoft.com/)
2. Mở Project Blockchain



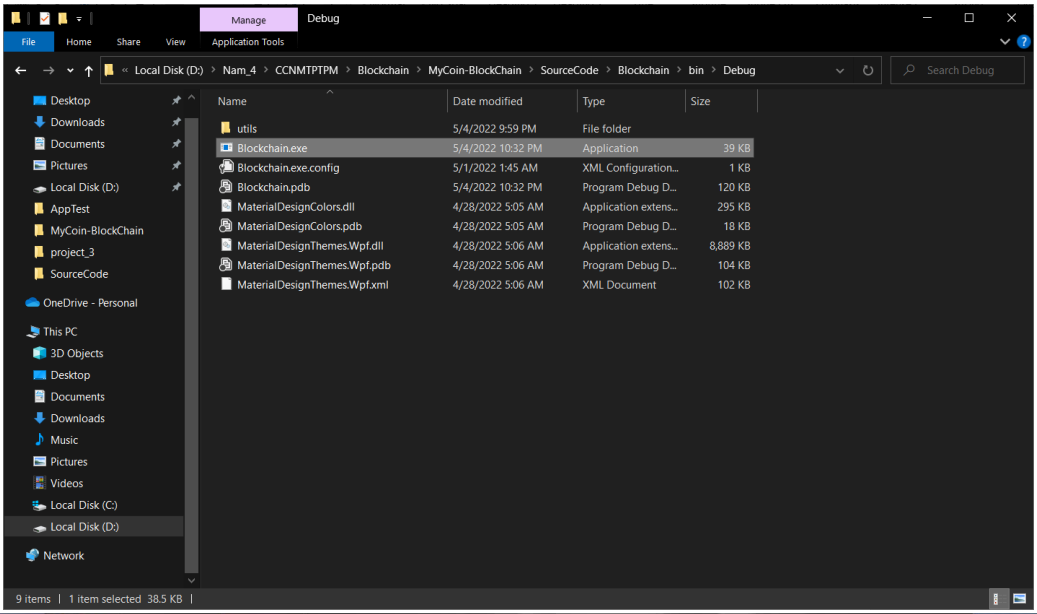
3. Build Project bằng tổ hợp phím Ctrl + Shift + B

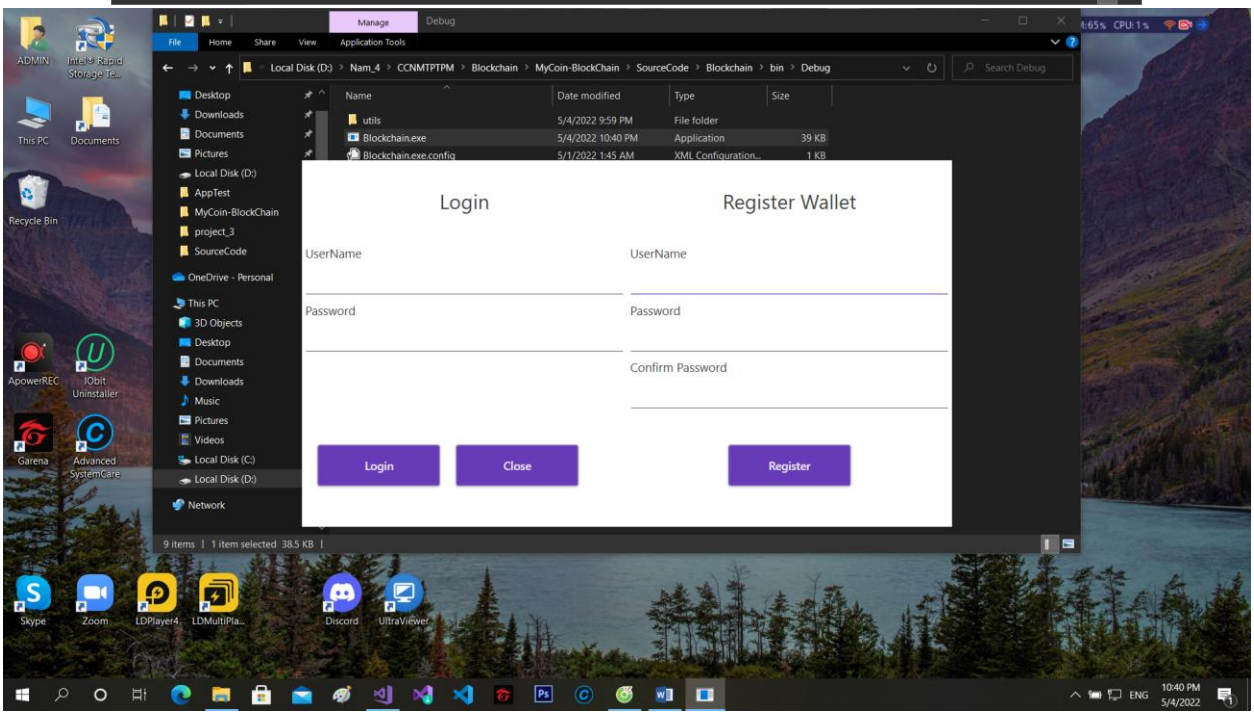


4. Chạy run bằng tổ hợp phím Ctrl + F5 để chạy ứng dụng



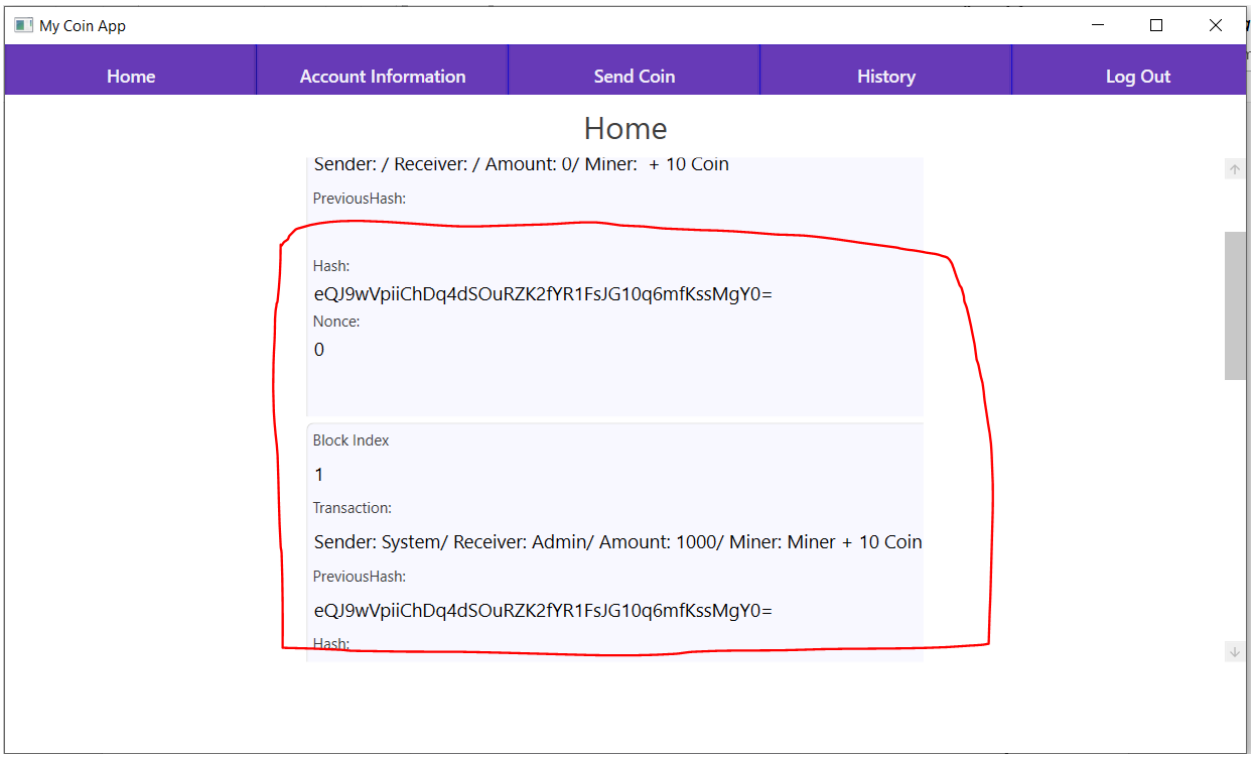
Hoặc mở file Blockchain.exe theo đường dẫn sau

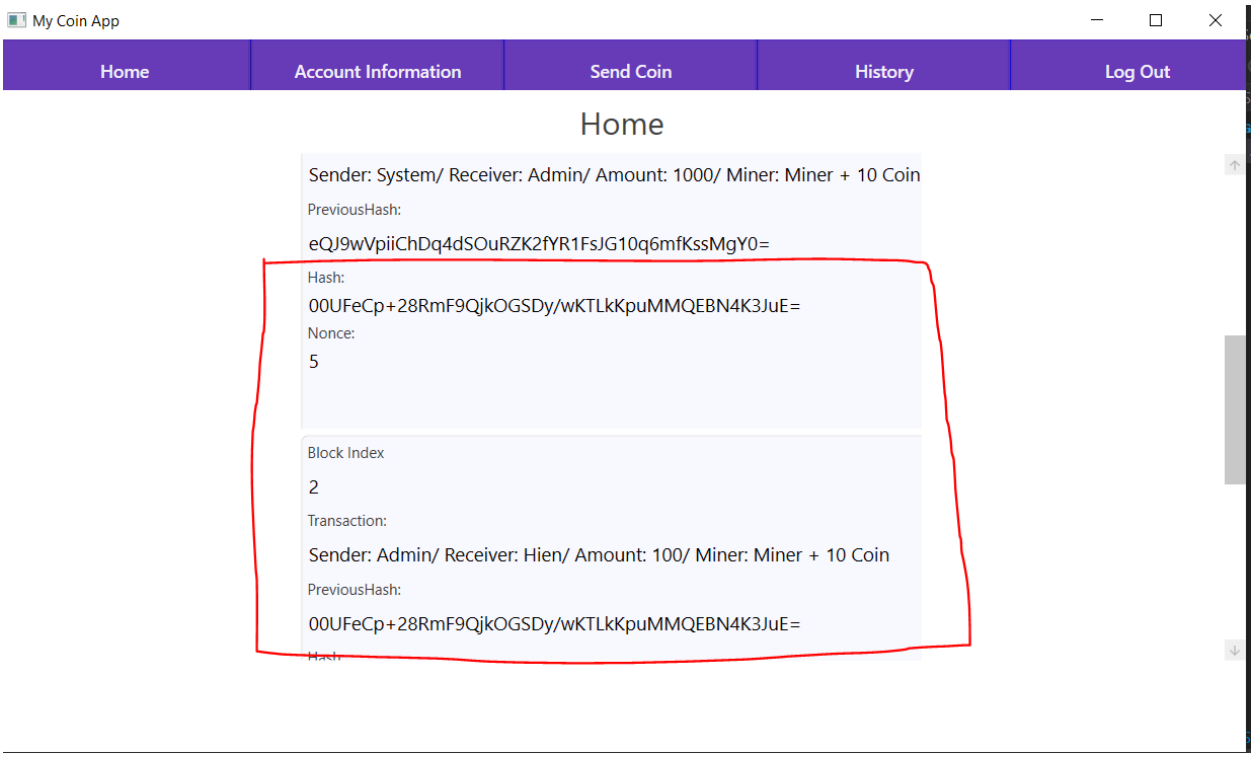




3 Hướng dẫn sử dụng chức năng

Hash của Block i trùng với PreviousHash của Block i+1 hợp lệ với quy tắc Blockchain

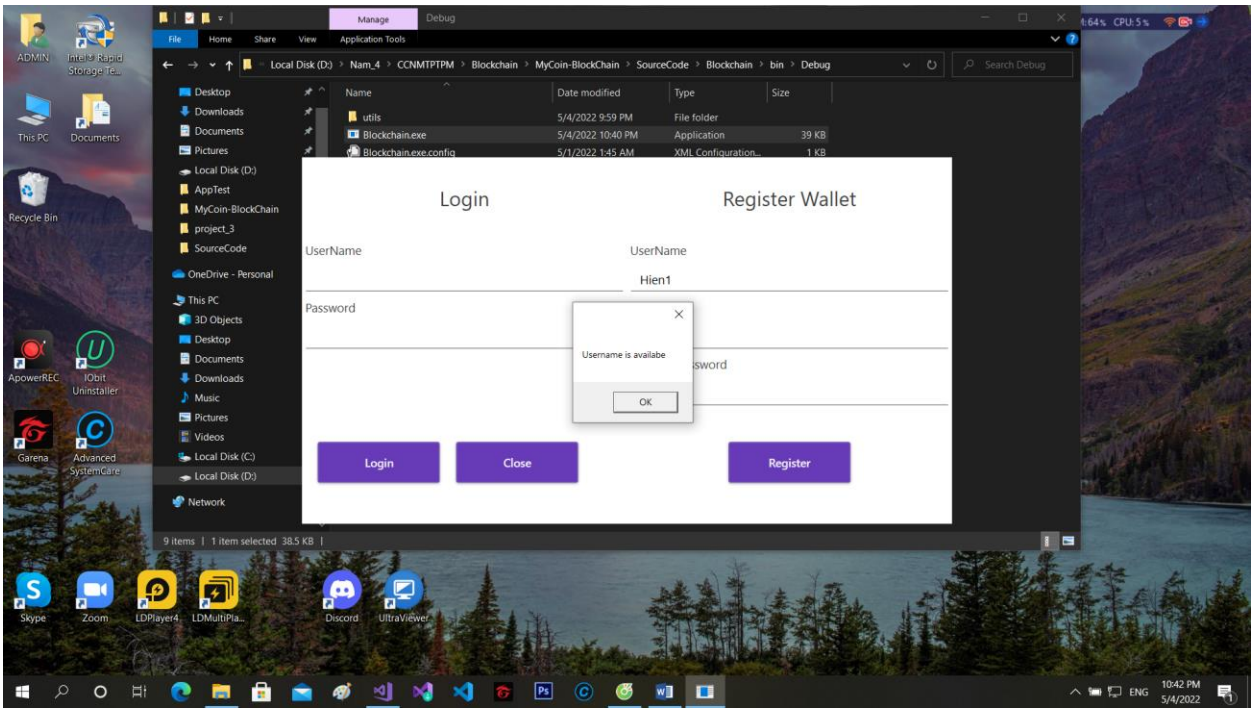




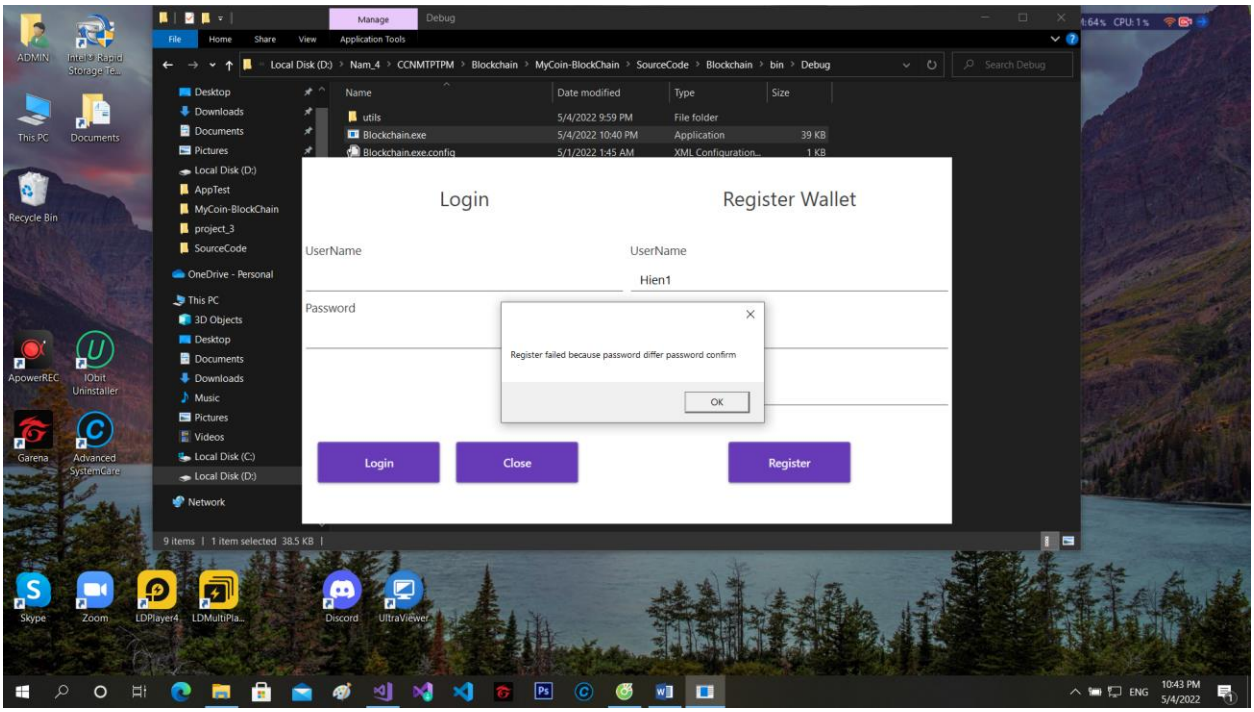
3.1 Tạo ví Wallet

Nhập vào Username, password, password confirm

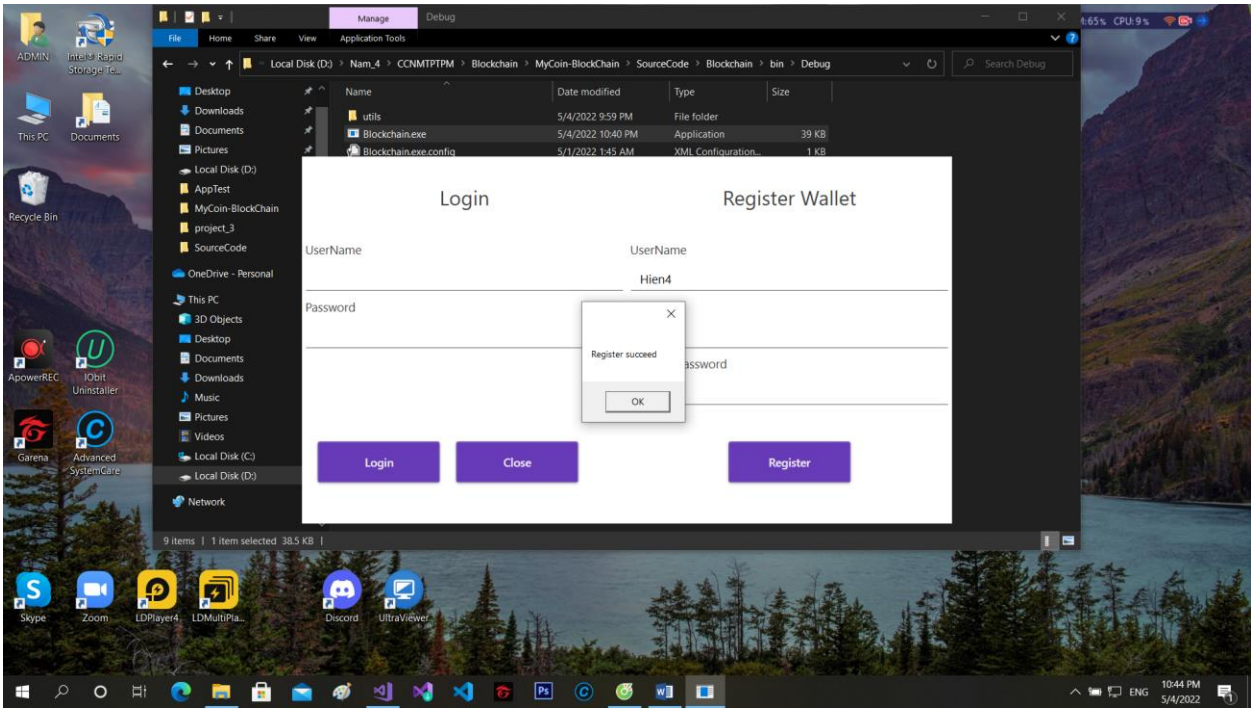
- Nếu Username đã được người khác sử dụng thì sẽ thông báo



- Nếu password không trùng khớp password confirm thì sẽ thông báo

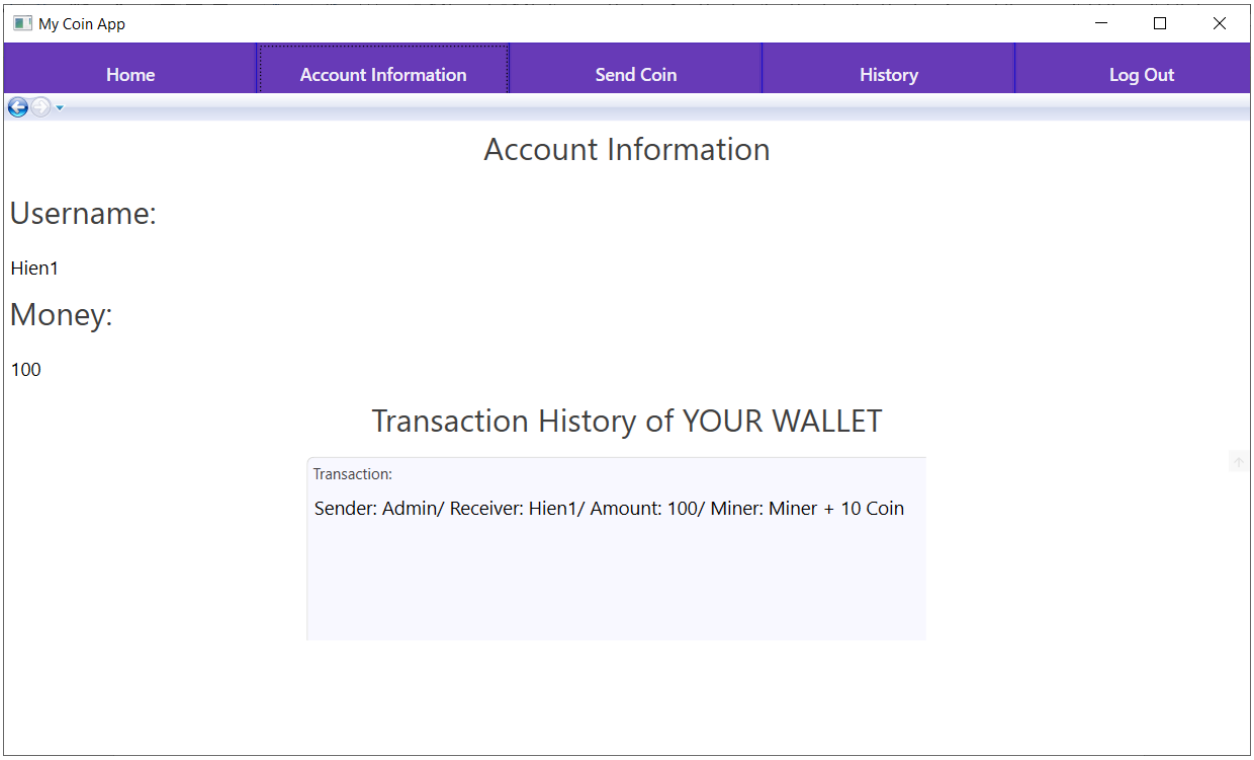


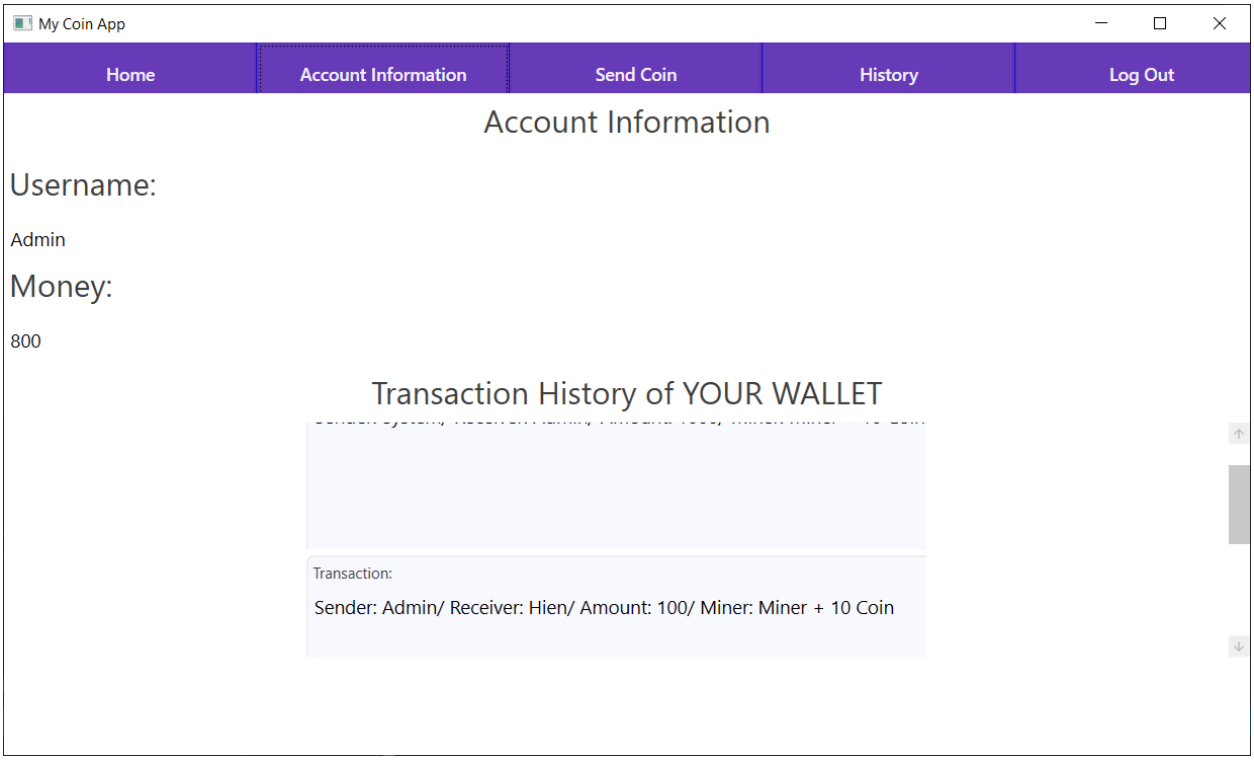
- Đang ký hợp lệ



3.2 Xem thống kê tài khoản

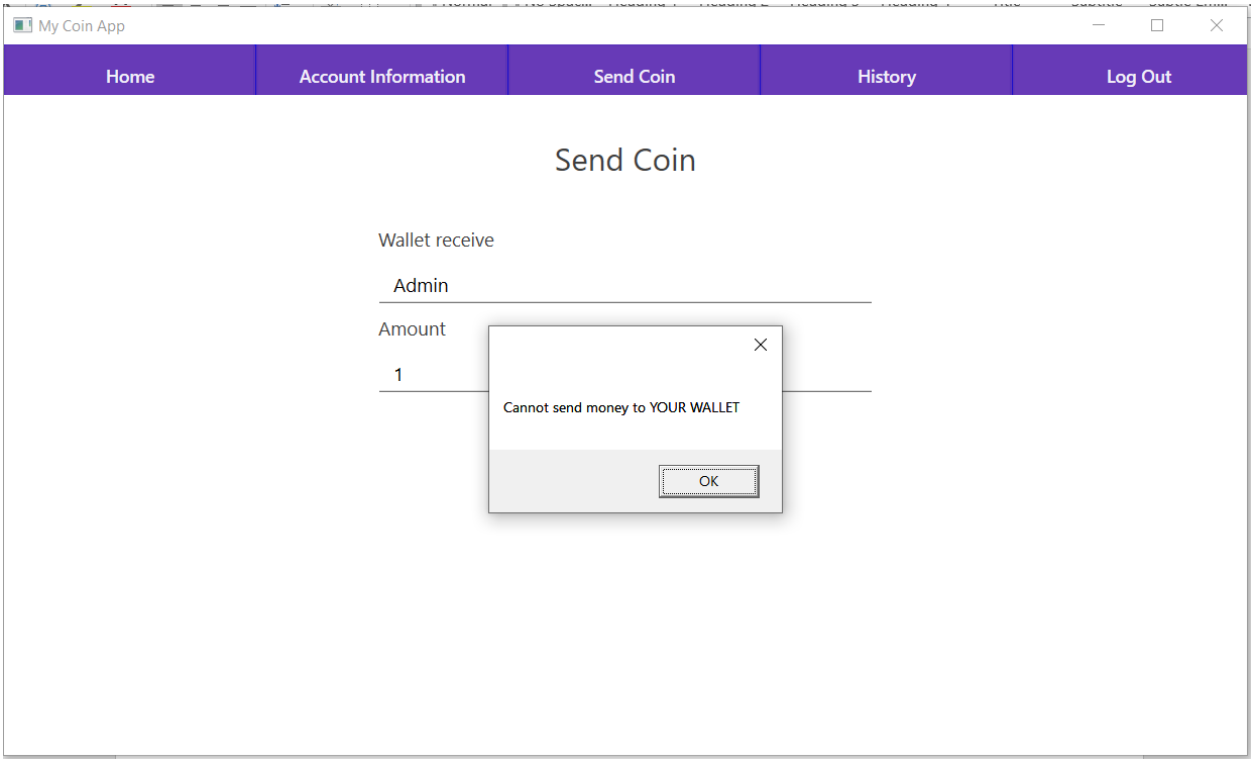
Hiển thị thông tin ví của người dùng gồm username, money số tiền hiện tại, những giao dịch mà người dùng đã tham gia, có mặt



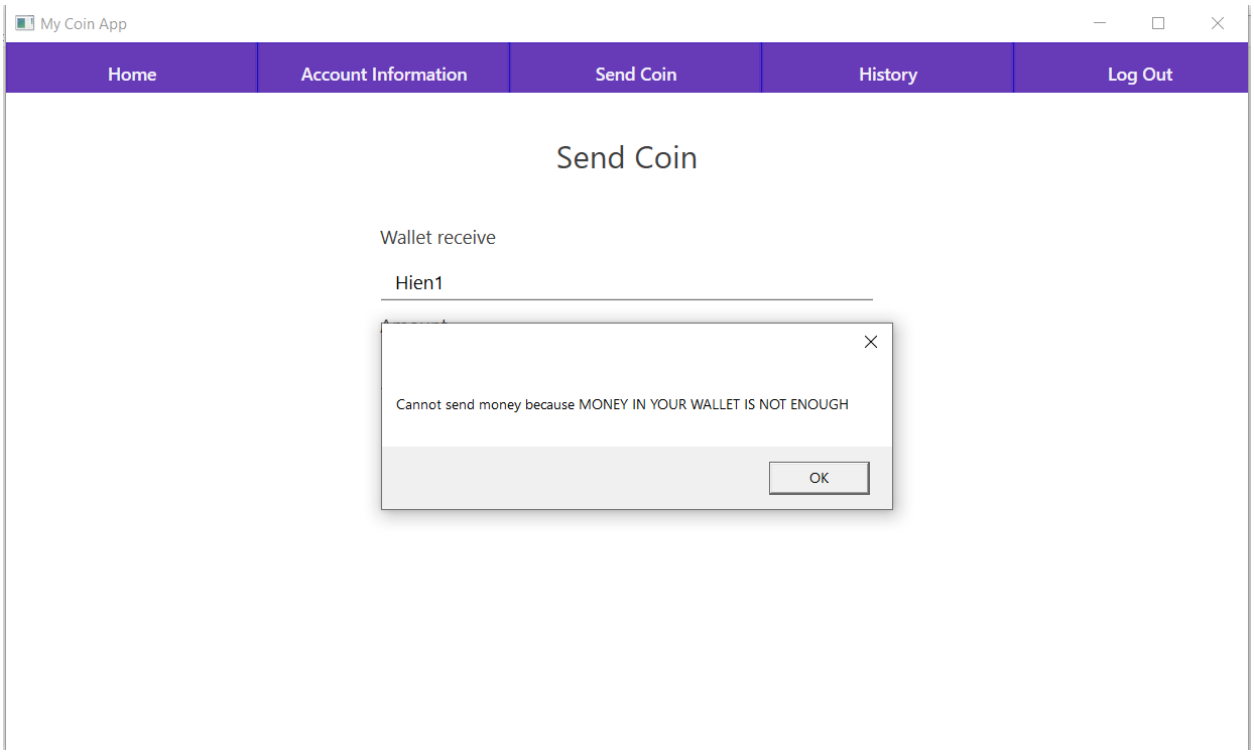


3.3 Gửi Coin cho 1 địa chỉ khác

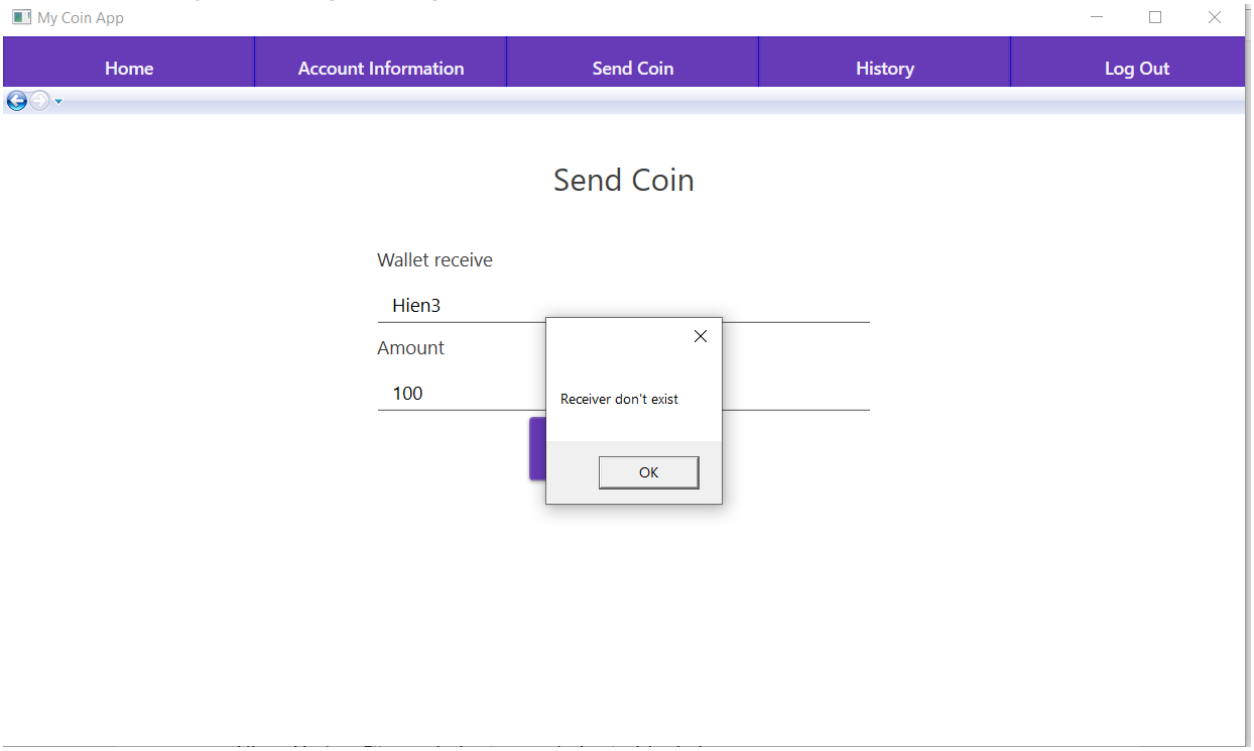
- Nếu wallet gửi coin cho chính wallet đó thì gửi về thông báo



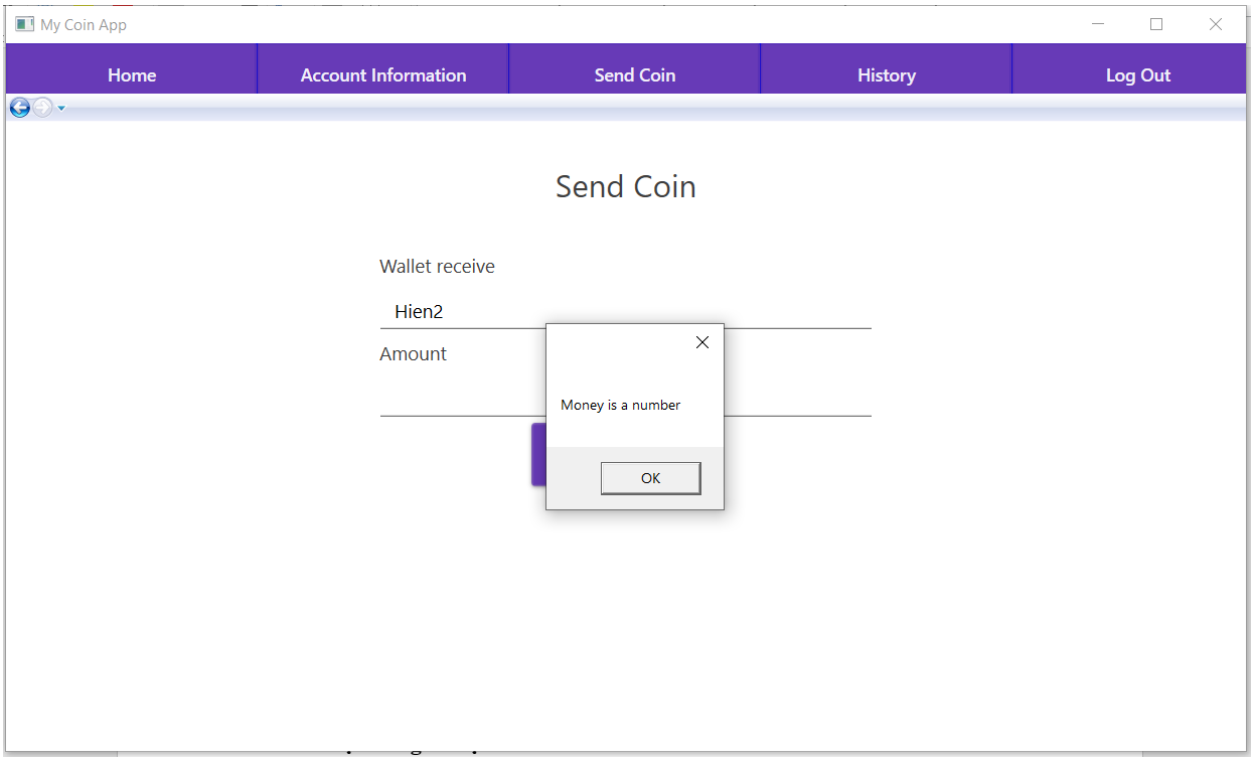
- Nếu số tiền gửi coin vượt quá số tiền trong ví gửi về thông bá



- Nếu người dùng không tồn tại

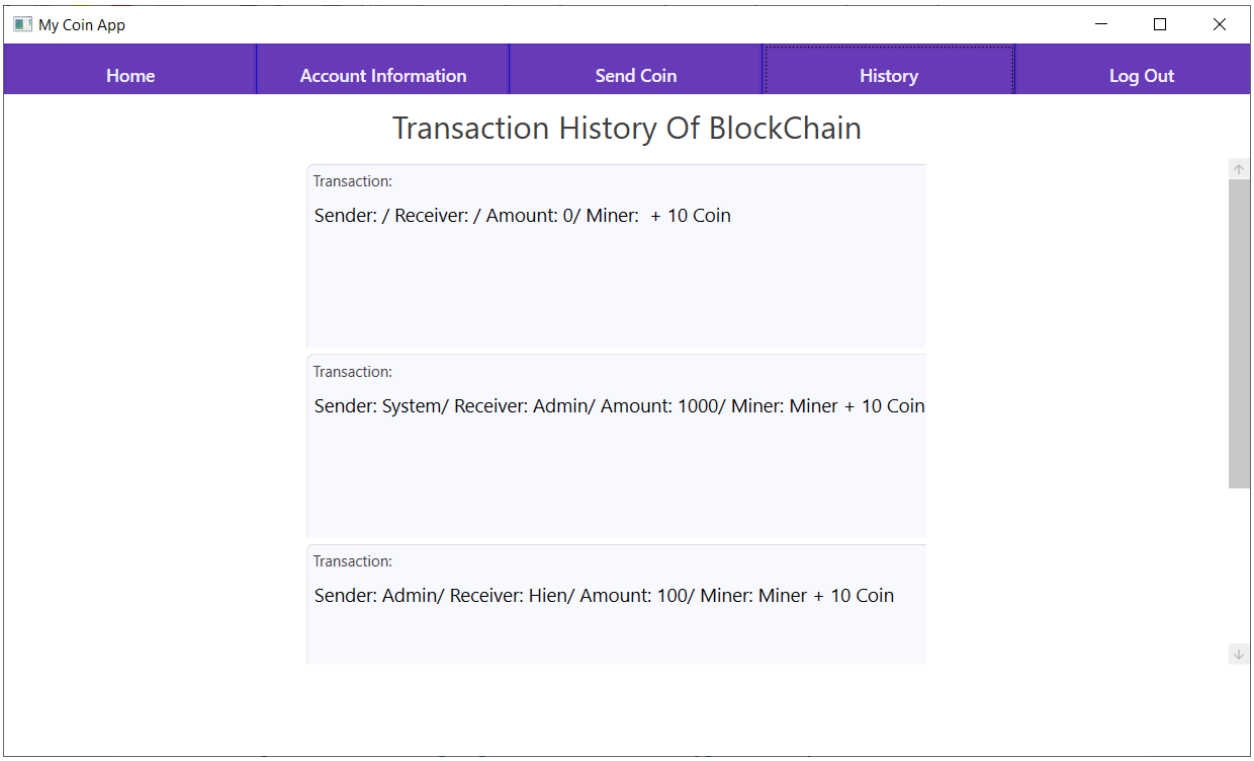


- Nếu số tiền bị rỗng

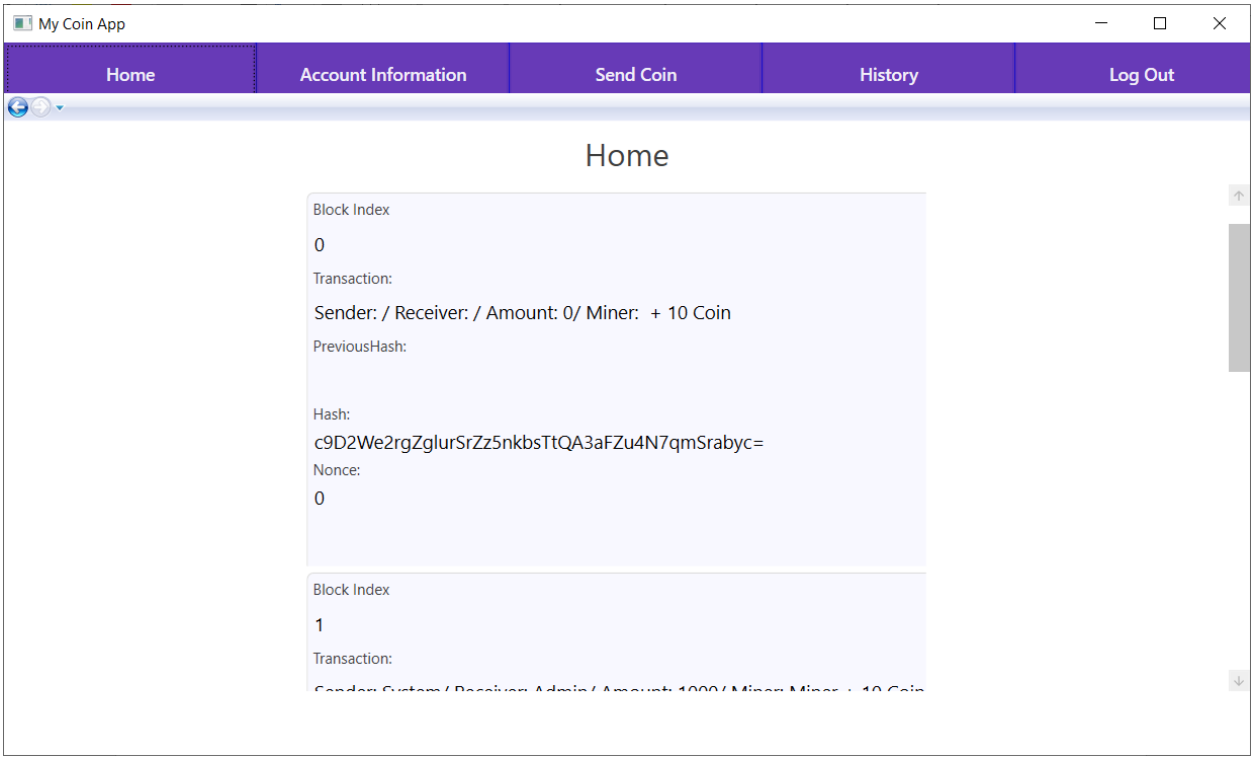


3.4 Xem lịch sử giao dịch của blockchain

Chọn Histoy để xem lịch sử giao dịch của blockchain



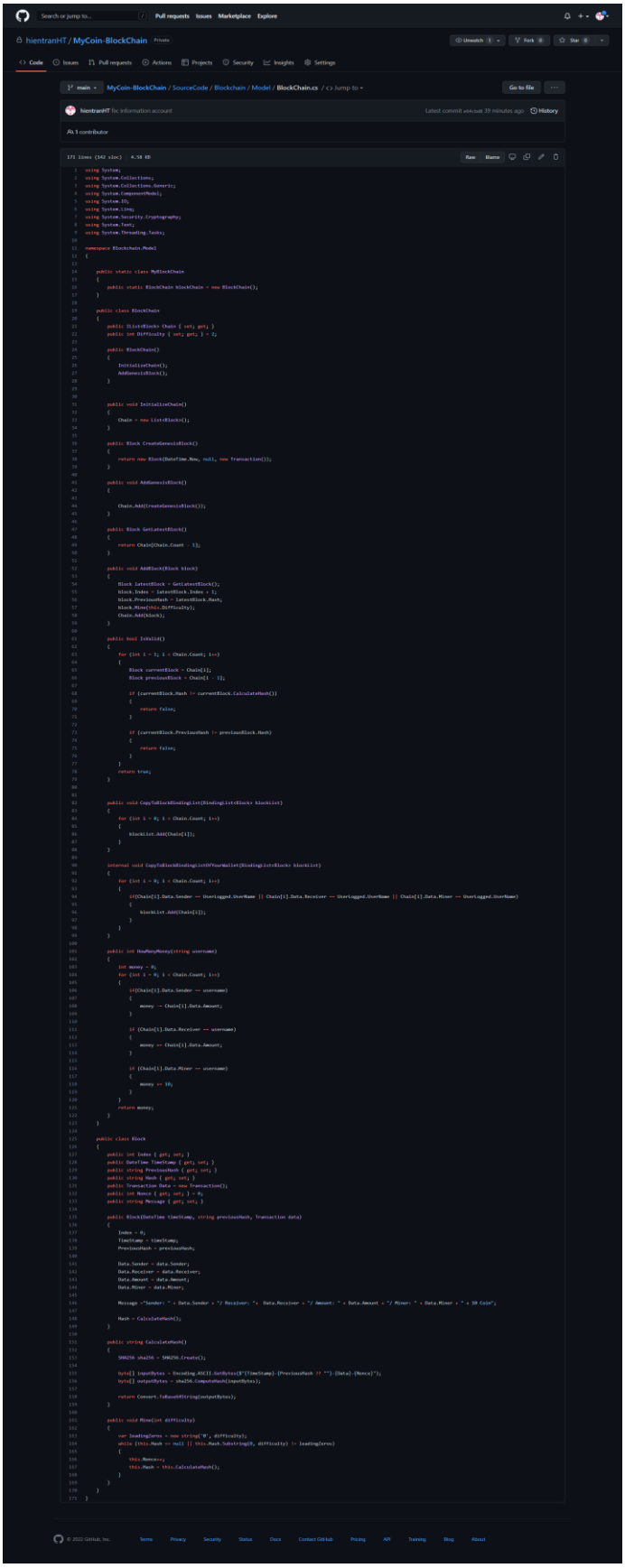
Chọn Home để theo dõi chi tiết Blockchain



3.5 Thuật toán Proof of Work

Vào thư mục Model -> BlockChain.cs

Dùng thuật toàn Proof of Work để tạo độ khó difficulty cho các Miner khi khởi tạo, tìm ra các block phù hợp



hientranHT / MyCoin-BlockChain

Private

Code Issues Pull requests Actions Projects Security Insights Settings

main · MyCoin-BlockChain / SourceCode / Blockchain / Model / BlockChain.cs

Go to file

hientranHT fix: information account

Latest commit e64ca88 39 minutes ago · History

1 contributor

171 lines (142 sloc) 4.58 KB

Raw Blame

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;

namespace Blockchain.Model
{

    public static class MyBlockChain
    {
        public static BlockChain blockChain = new BlockChain();
    }

    public class BlockChain
    {
        public IList<Block> Chain { get; get; }
        public int Difficulty { set; get; } = 2;

        public BlockChain()
        {
            InitializeChain();
            AddGenesisBlock();
        }


        public void InitializeChain()
        {
            Chain = new List<Block>();
        }

        public Block CreateGenesisBlock()
        {
            return new Block(DateTime.Now, null, new Transaction());
        }

        public void AddGenesisBlock()
        {

            Chain.Add(CreateGenesisBlock());
        }

        public Block GetLatestBlock()
        {
            return Chain[Chain.Count - 1];
        }

        public void AddBlock(Block block)
        {
            Block latestBlock = GetLatestBlock();
            block.Index = latestBlock.Index + 1;
            block.PreviousHash = latestBlock.Hash;
            block.Mine(this.Difficulty);
            Chain.Add(block);
        }

        public bool IsValid()
        {
            for (int i = 1; i < Chain.Count; i++)
            {
                Block currentBlock = Chain[i];
                Block previousBlock = Chain[i - 1];

                if (currentBlock.Hash != currentBlock.CalculateHash())
                {
                    return false;
                }

                if (currentBlock.PreviousHash != previousBlock.Hash)
                {
                    return false;
                }
            }
            return true;
        }


        public void CopyToBlockBindingList(BindingList<Block> blocklist)
        {
            for (int i = 0; i < Chain.Count; i++)
            {
                blocklist.Add(Chain[i]);
            }
        }

        internal void CopyToBlockBindingListOfYourWallet(BindingList<Block> blocklist)
        {
            for (int i = 0; i < Chain.Count; i++)
            {
                if(Chain[i].Data.Sender == UserLogged.UserName || Chain[i].Data.Receiver == UserLogged.UserName || Chain[i].Data.Miner == UserLogged.UserName)
                {
                    blocklist.Add(Chain[i]);
                }
            }
        }

        public int HowManyMoney(string username)
        {
            int money = 0;
            for (int i = 0; i < Chain.Count; i++)
            {
                if(Chain[i].Data.Sender == username)
                {
                    money -= Chain[i].Data.Amount;
                }

                if (Chain[i].Data.Receiver == username)
                {
                    money += Chain[i].Data.Amount;
                }

                if (Chain[i].Data.Miner == username)
                {
                    money += 10;
                }
            }
            return money;
        }
    }

    public class Block
    {
        public int Index { get; set; }
        public DateTime TimeStamp { get; set; }
        public string PreviousHash { get; set; }
        public string Hash { get; set; }
        public Transaction Data = new Transaction();
        public int Nonce { get; set; } = 0;
        public string Message { get; set; }

        public Block(DateTime timeStamp, string previousHash, Transaction data)
        {
            Index = 0;
            TimeStamp = timeStamp;
            PreviousHash = previousHash;

            Data.Sender = data.Sender;
            Data.Receiver = data.Receiver;
            Data.Amount = data.Amount;
            Data.Miner = data.Miner;

            Message = "Sender: " + Data.Sender + "/ Receiver: "+ Data.Receiver + "/ Amount: " + Data.Amount + "/ Miner: " + Data.Miner + " + 10 Coin";

            Hash = CalculateHash();
        }

        public string CalculateHash()
        {
            SHA256 sha256 = SHA256.Create();

            byte[] inputBytes = Encoding.ASCII.GetBytes($"{TimeStamp}-{PreviousHash ?? ""}-{Data}-{Nonce}");
            byte[] outputBytes = sha256.ComputeHash(inputBytes);

            return Convert.ToBase64String(outputBytes);
        }

        public void Mine(int difficulty)
        {
            var leadingZeros = new string('0', difficulty);
            while (this.Hash == null || this.Hash.Substring(0, difficulty) != leadingZeros)
            {
                this.Nonce++;
                this.Hash = this.CalculateHash();
            }
        }
    }
}
```

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

4 Tài liệu, trang ứng dụng tham khảo

Link tham khảo bài tập cá nhân Blockchain:

https://www.smashingmagazine.com/2020/02/cryptocurrency-blockchain-node-js/?fbclid=IwAR13mQOF1k1aWgxbtsPwQjwk0kpGYkj6ADw6YjS4-5ZLjJWMTpG93My3xDI

https://www.myetherwallet.com/wallet/create

https://www.c-sharpcorner.com/article/blockchain-basics-building-a-blockchain-in-net-core/

https://www.c-sharpcorner.com/article/building-a-blockchain-in-net-core-proof-of-work/