**Semester Project Report**

**Software Development Team**

***Methods for Securing Blockchain-Based Transactive Energy Systems***

**Team Members**

Ismail Asorona

Gbolahan Afloabi

Mecahel Demissie

Glory Ezeogu

Boluwatife Shekoni

Emmanuel Yormesor

**Introduction**

**Purpose:** This research aims to develop a solution that will better manage the movement of energy systems between transactive energy systems.

**Solution:** We developed a program that will utilize blockchain to create a secure communication between the energy company and the end-user without the use of an intermediary.

**Real World Application:** Blockchain based Transactive Energy Systems will automate checks and balances of energy consumption within a network that is validated by everyone which makes it a decentralized ledger.

**Our Task**

This group will be writing a program for the blockchain The foremost language you can choose to step into the blockchain arena is Java. Python is fast becoming the most popular programming language in the world and if you are a new developer with experience in Python, then it might be the best Blockchain language for you.

**Goal**

* Successfully create a program that can safely secure the movement of energy systems between blocks.
* Create secure communication between the energy company and the end-user.
* Create a blockchain that accepts energy input from multiple sources and distributes it after.

**Transactive Energy**

A system in which anyone can trade electricity on the grid is referred to as transactive energy. Traditional power consumers can create electricity and sell their excess capacity back into the grid using this technique.

Transactive energy is an effective way to share and trade energy among peers. A transactive energy framework is composed of several integrated blocks such as an energy market, service providers, generation companies, transmission and [distribution networks](https://www.sciencedirect.com/topics/engineering/electric-power-distribution), prosumers, etc. The success of such a framework can be measured by analyzing the effectiveness of its major building blocks.

**Algorithm**

Methods= **Bold**

Import packages (hash, time, json)

Create block constructor class

Set block number, name, energy source,, location, energy input, timestamp, and present hash

**Create a function to get the previous block in blockchain list**

Return list of blockchain decremented by 1

**Create a function to add a block to blockchain list**

Verify if block is valid

If valid send to mine method

**Create function to calculate hash using sha256**

Get block string

Return hash to present hash

**Create a function to mine block to show record in blockchain list**

List the mine blocked, nonce, block number, the input of the user and timestamp

**Create function to print hashes**

Print previous and current hash

**Get blockchain length**

Get index of block

Determine if block is verified

**Create blockchain constructor class**

Create list of blockchain

Get previous hash

**Input state of the user**

If the user is not in GA, DC, MD, or VA, no access to program

Create bank of how much energy is stored

Input choice of buying, selling, or quit the program

Input name, location, energy source, and amount of energy input

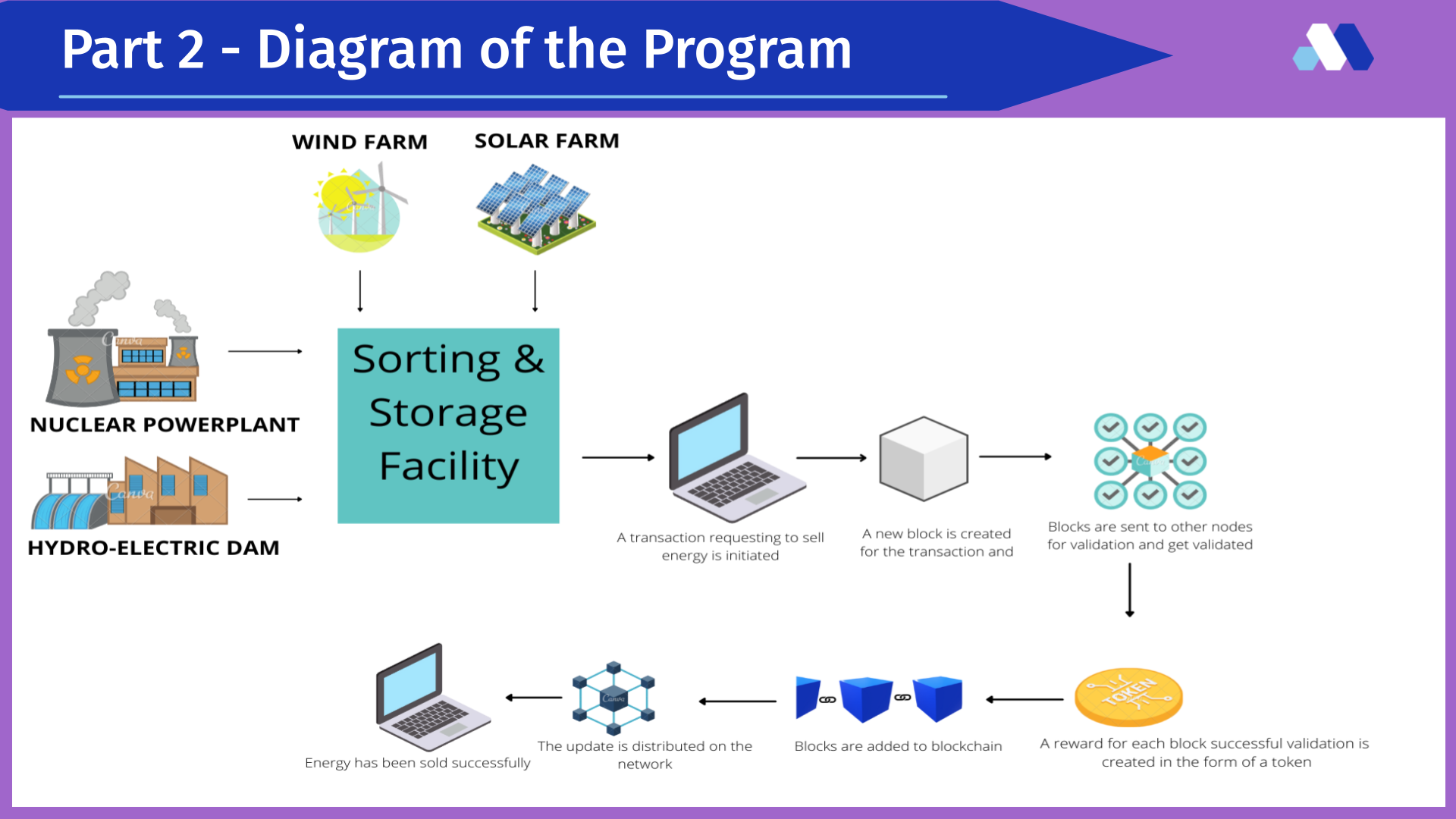
Print block record, and blockchain length

**Create default block with default values**

Calculate default block hash

Add default block to blockchain list

**Flowchart**



***Fig 1 - Team Block Program Model***

**Source Code**

The source code for this project was written using the python programming language. Python is a high-level, general-purpose programming language that is widely used. The Python programming language is utilized in web development, machine learning applications, and every other cutting-edge technology in the software industry such as blockchain. Due to the efficiency of python code in implementing blockchain, we chose to use it for this project. The testing environment in which this source code was written, debugged and tested is called replit. Repl.it is a free integrated development environment (IDE) that enables users to write their own programs and code in a variety of languages.

In this section, we will explain the functions of the various code used and how they were used in the program.

The first step is to import necessary libraries that can help implement other functions such as hashlib, json and time. hashlib, json and time are necessary libraries in python that help with length sequence, handle dataflow & real time value.

**SourceCode**

class Block:

def \_\_init\_\_(self, block\_number, data, location, device\_name, energy, nonce=0,

prev\_hash=" ", timestamp=None):

self.block\_number = block\_number

self.data = data

self.timestamp = timestamp or time.time()

self.nonce = nonce

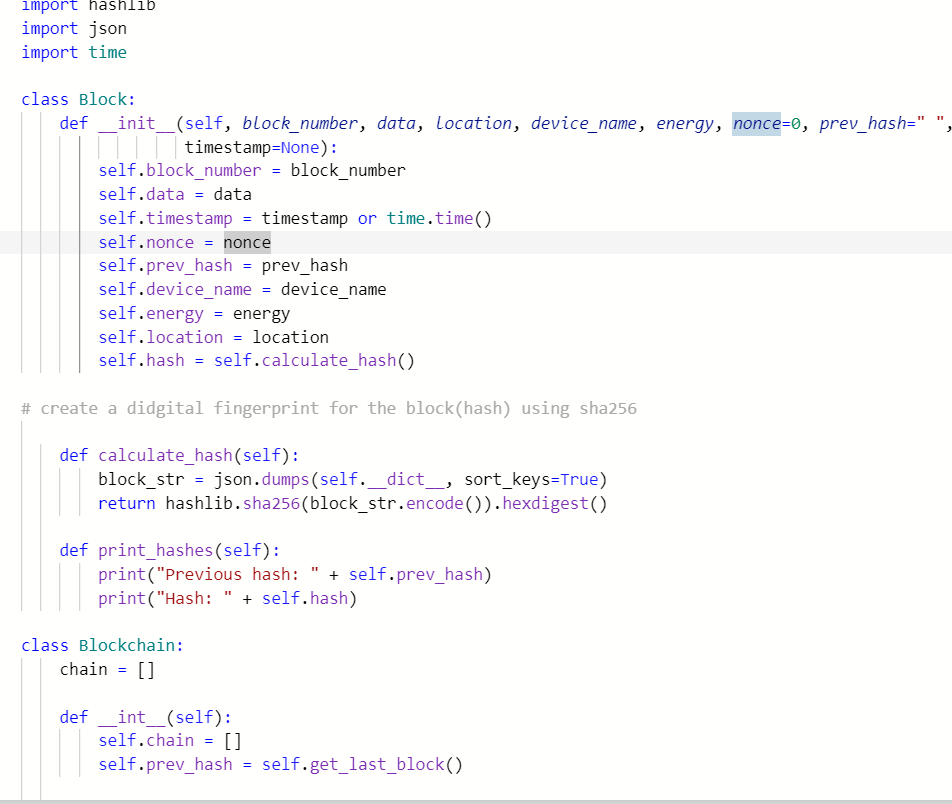
self.prev\_hash = prev\_hash

self.location = location

self.device\_name = device\_name

self.energy = energy

self.hash = self.calculate\_hash()



***Fig 2 - Screenshot of the source code - definition of variables***

The first screenshot shows some very important sections of the code which is to import the necessary libraries such as hashlib, json and timestamp.

Hashlib -

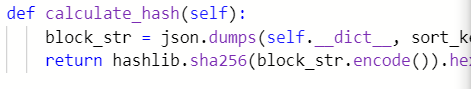
Json -

Timestamp - imports functions that record transactions in real time.

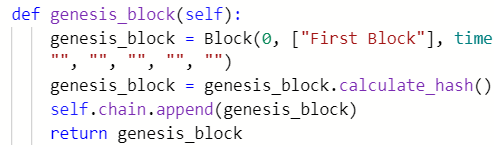
Next is to create a block class where all variables (transactions that are to be recorded in the Team block chain) are initialized.

The transactions includes

* Block number - this calculates the number of blocks in a chain. When different transactions occur this calculates the total chain created and informs teamblock. Example 5 equals 5 blocks.
* Data - this includes the details of the transaction such as the buyer or seller name
* Location - this variable is used to validate location of transaction (teamblock can only transact with 4 states and they include DC, MD, VA and AT) and further used to calculate the amount of energy using the zip code.
* Device name - This stores the type of power source. Here the user enters the power source they are interested in either buying or selling.
* Energy - this stores the amount of energy to be bought or sold and then calculates the recent amount of energy in stock after a transaction is made of.
* Nonce -
* Previous hash - this calculates the previous hash and is used to create the present hash.

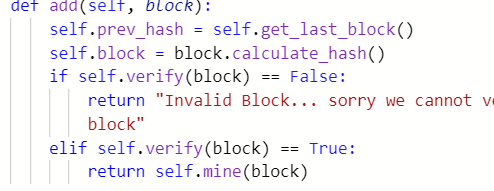


a method will be created to calculate and generate hash using sha256

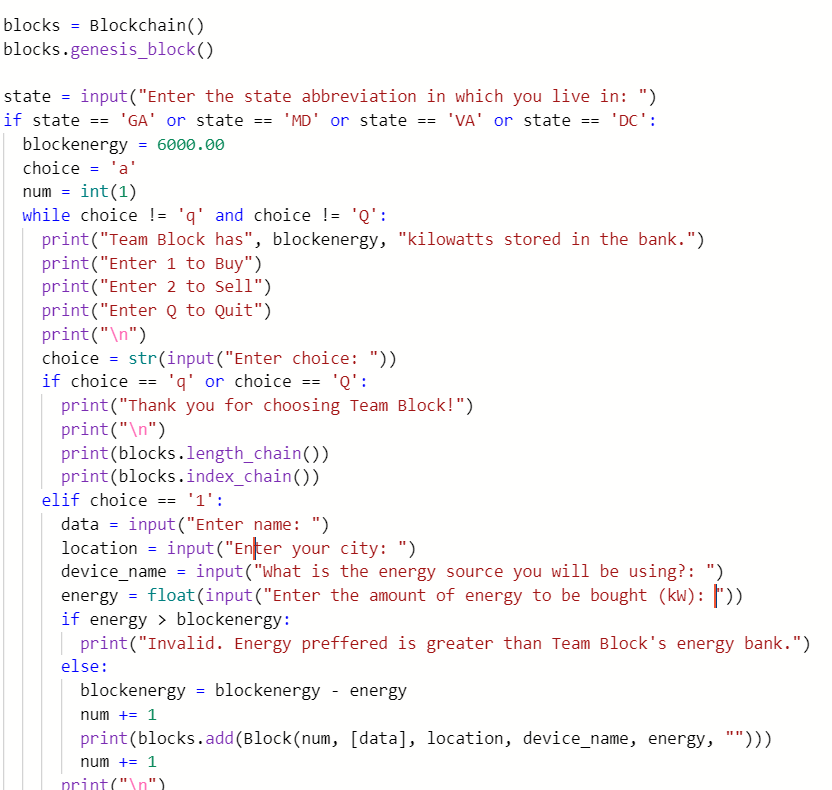


Creates the genesis block

genesis block method used to build initial block to the chain and generate the hash for first block



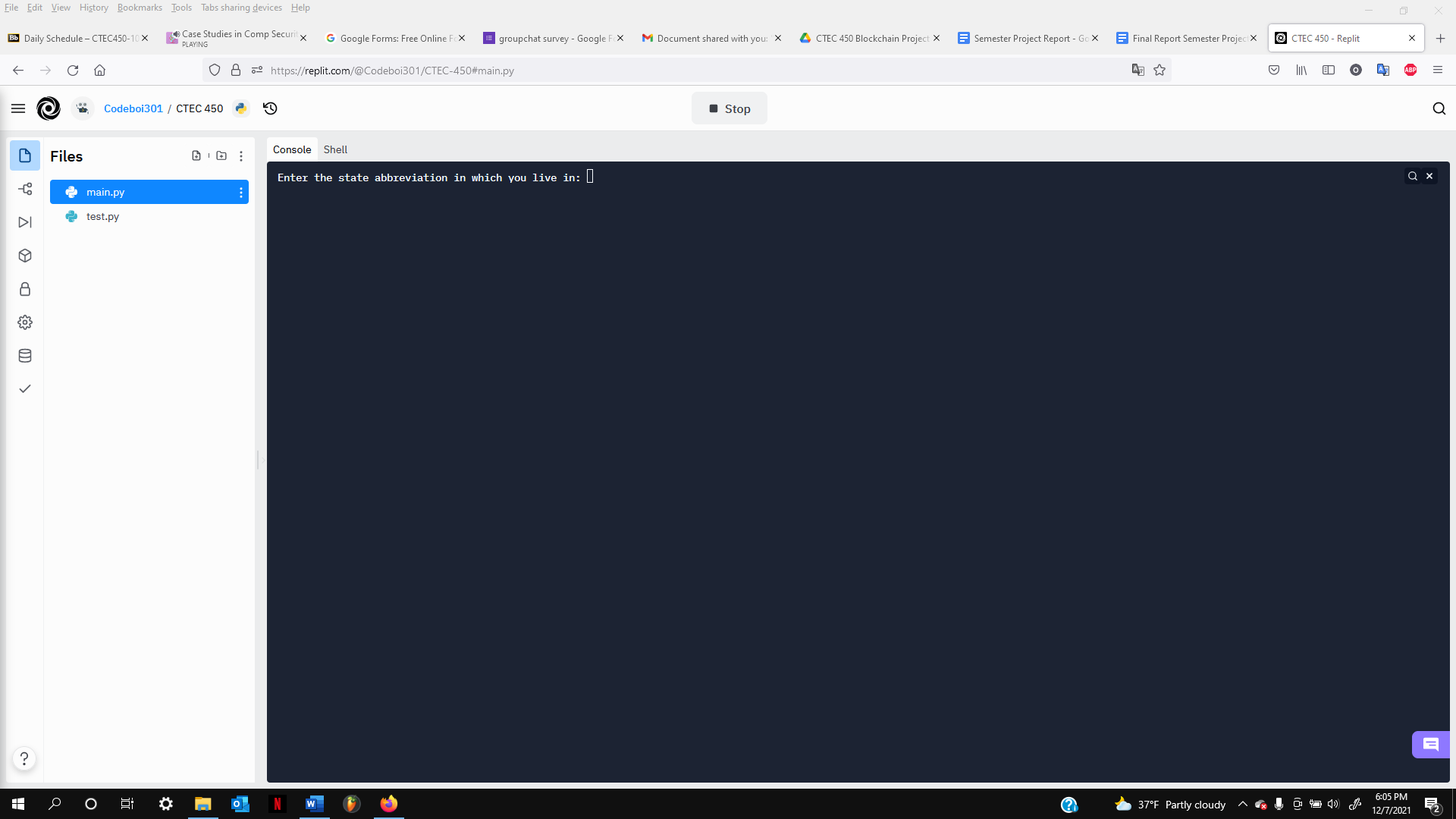
used to add new block and verify the last block before it uses it to create a new hash and chain



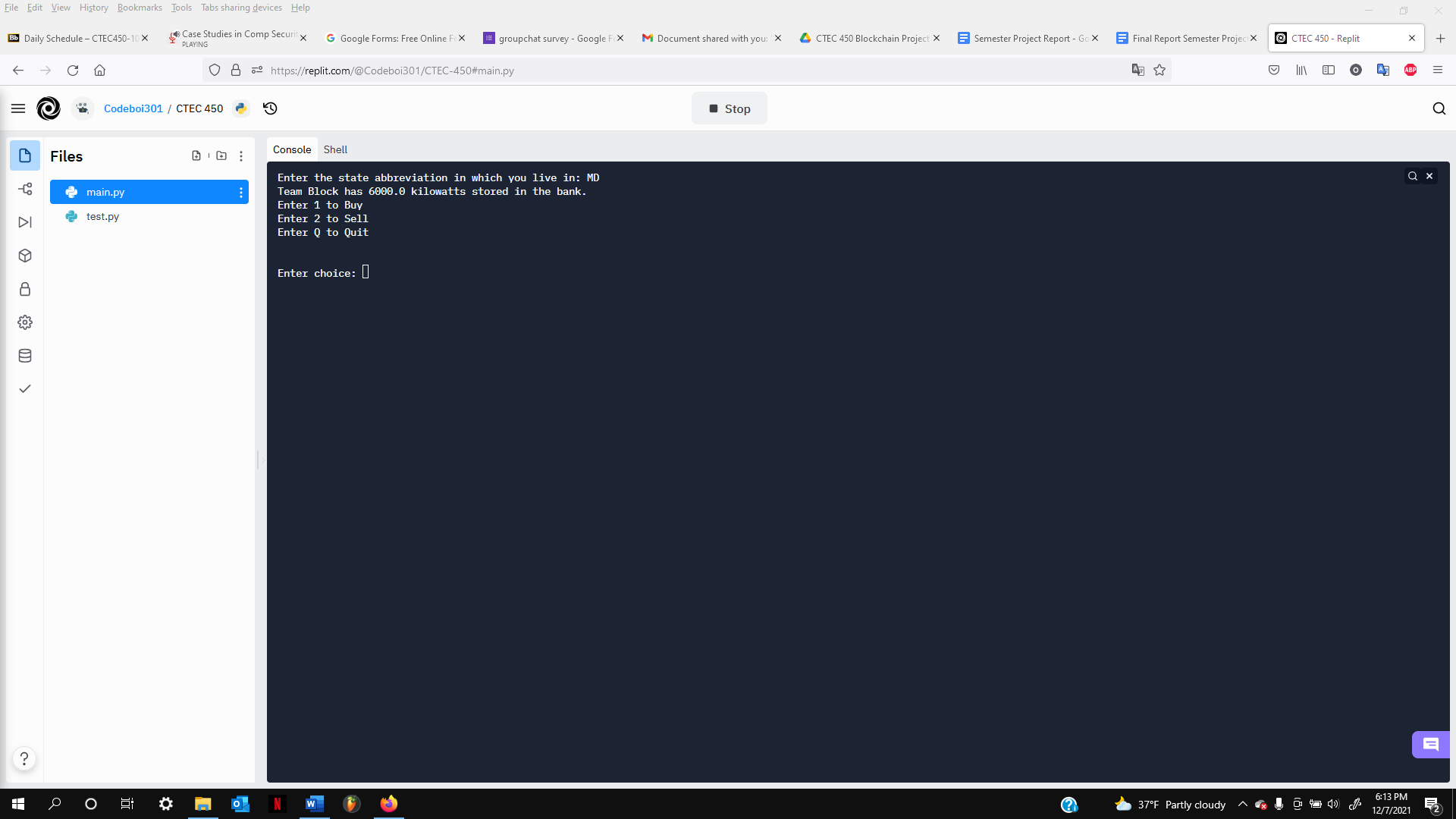
***Img 2 - Screenshot of the expected input and output***

The screenshot above applies the use of conditional if statement that validates users entry. First before any transaction occurs users are prompted to enter their state where translation will take place. If state is not equal to any of the program conditions(MD< AT<VA<DC) then access will be denied and the program will exit. However if any of the 4 states mentioned above is entered, access will be allowed and users can proceed to buy or sell. Depending on their choice they will be prompted to enter their details such as how much to buy or sell, the type of energy they intend to transact and the amount. Finally it prints out the hash result and available balance of the Team Block energy reserve.

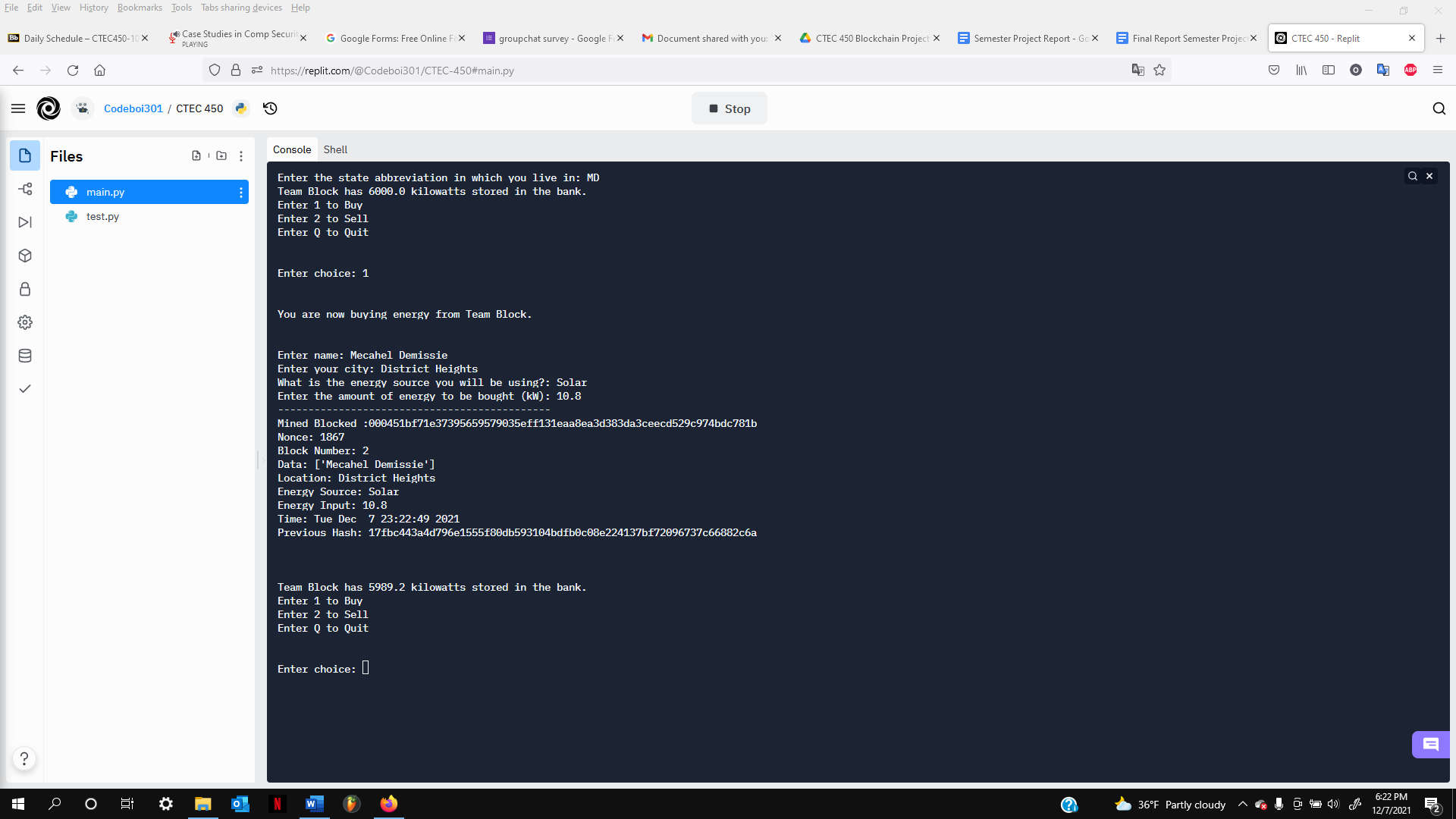
**Program Results**



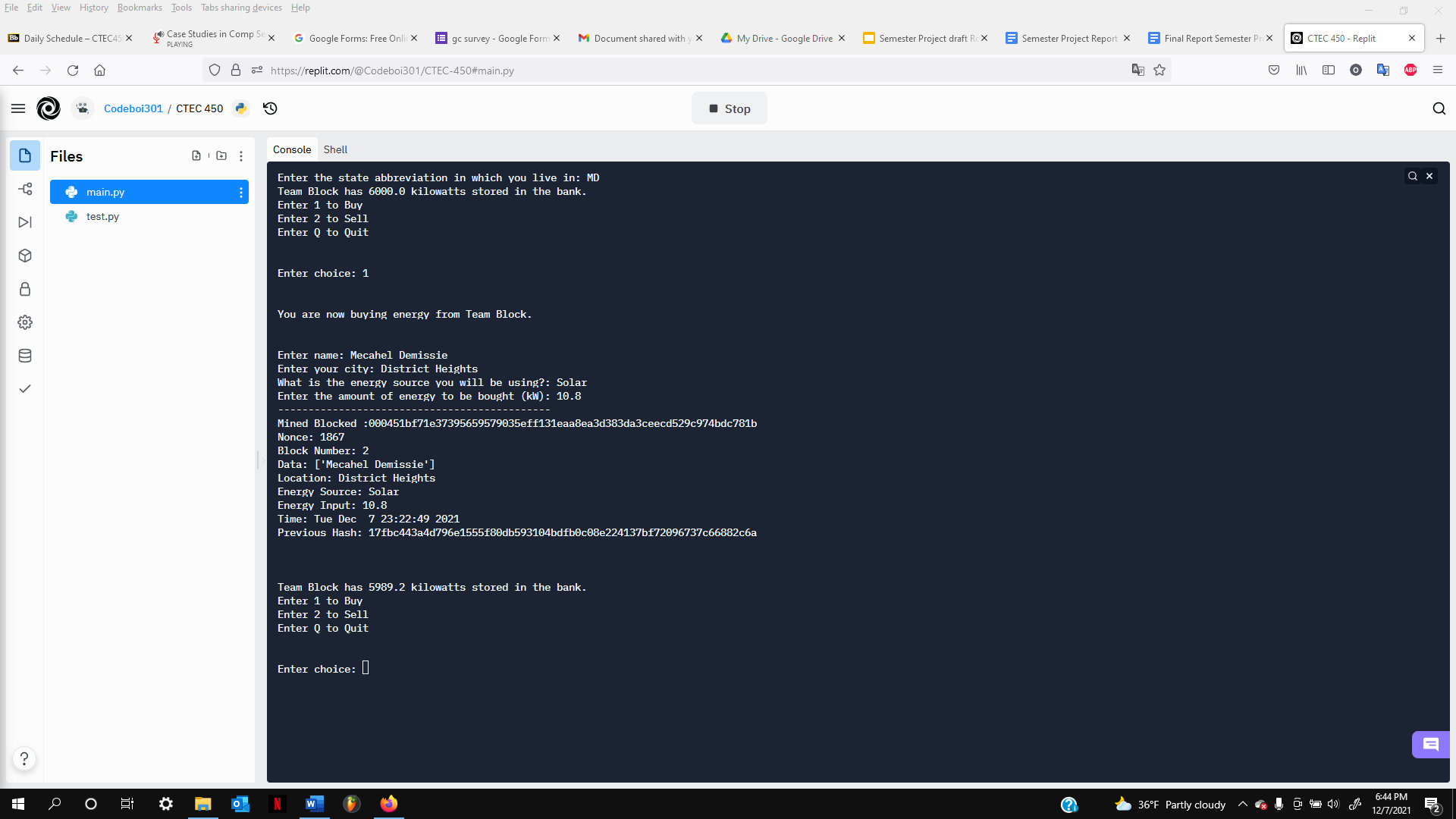
The program starts with asking the user which state they live in. This helps secure access as the only valid states are Georgia, Maryland, Virginia, and the city Washington D.C. Any other input would deny access to the blockchain system. The reason for that is Team Block has buildings that are located in the valid entries.



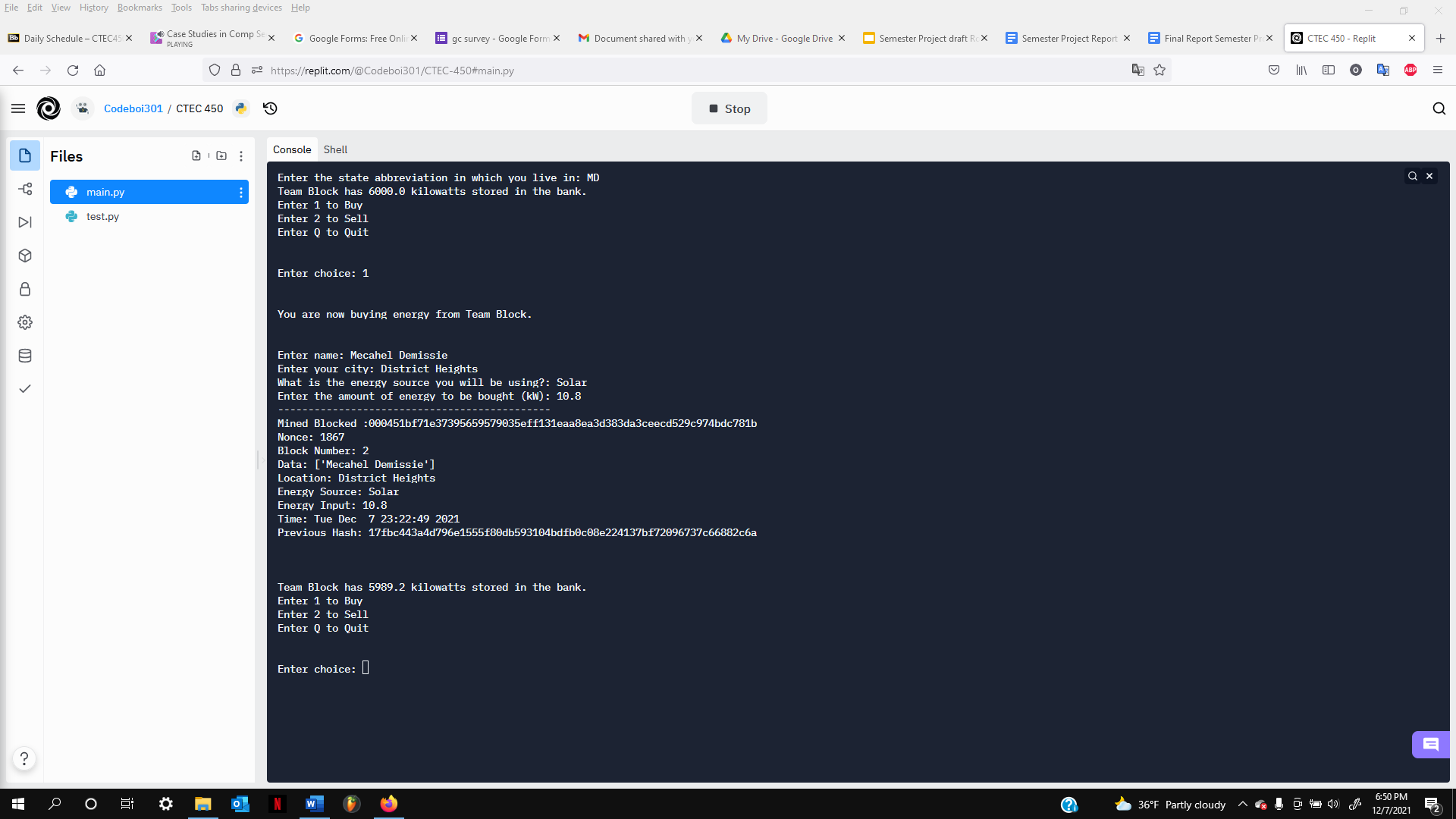
After entering the state, the user is displayed the total amount of kilowatts Team Block has stored in the back. The user is given three options where they can either buy, sell, or quit.



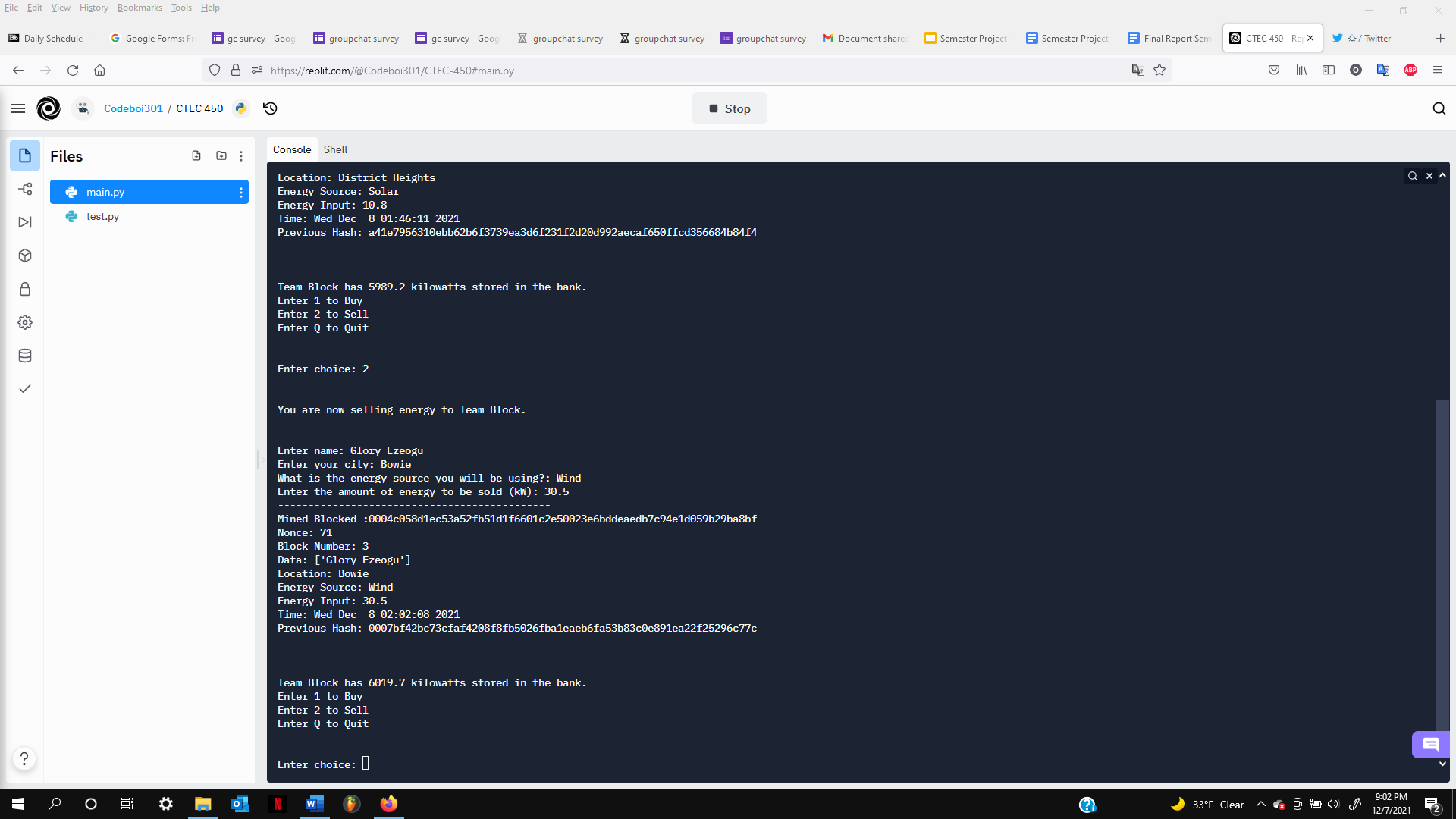
If the user enters “1”, they choose the option to buy. This prompts them to enter their credentials such as their name, city they live in, the energy source used to transfer energy, and the amount of energy they want to buy.



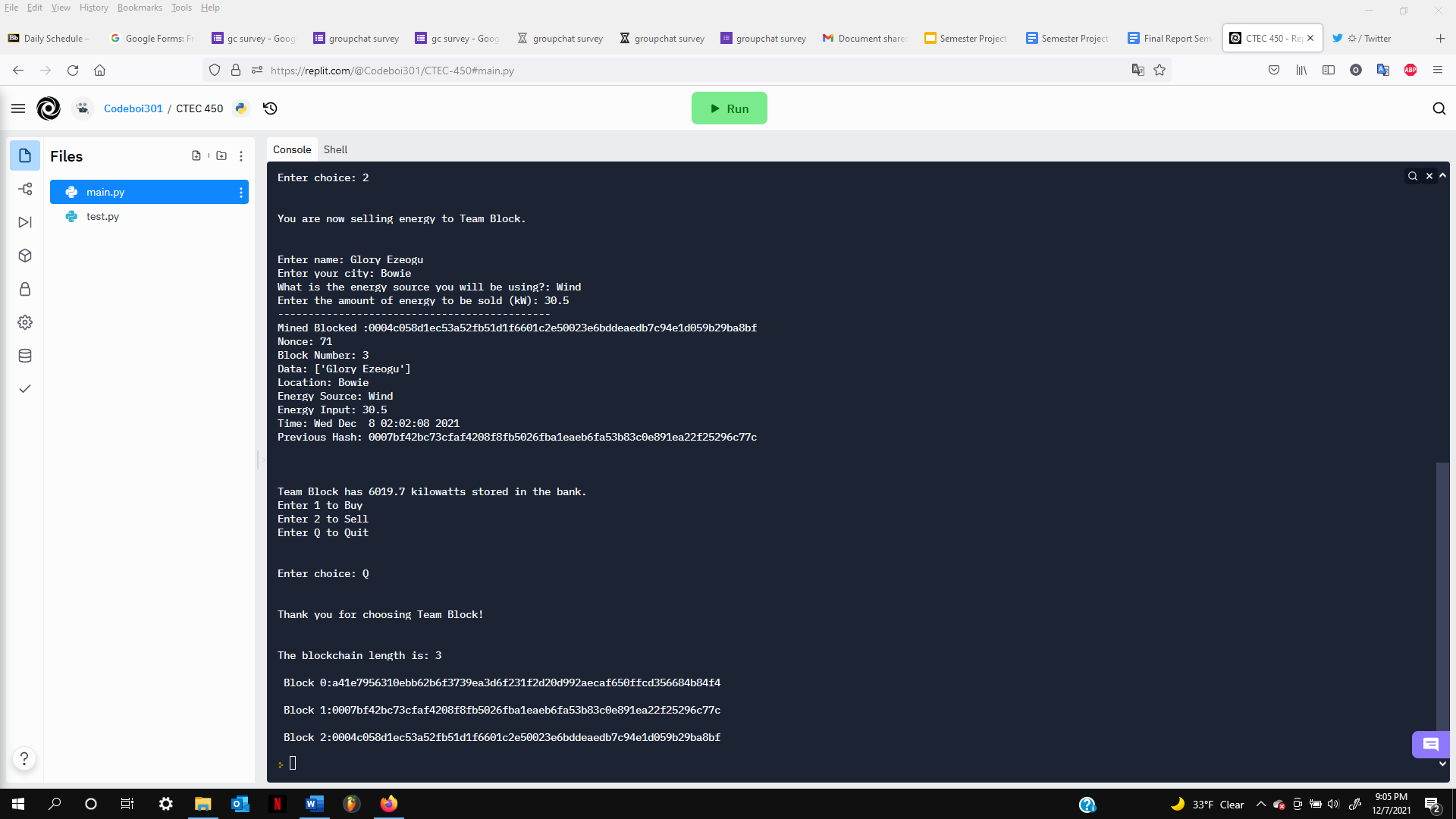
After entering the information, the record is recorded and stored into the blockchain system. Time, hash, and nonce has also been recorded at the time. The previous hash is also displayed.



The energy bank is then updated showing the current kilowatts and the user is prompted to enter a choice again.



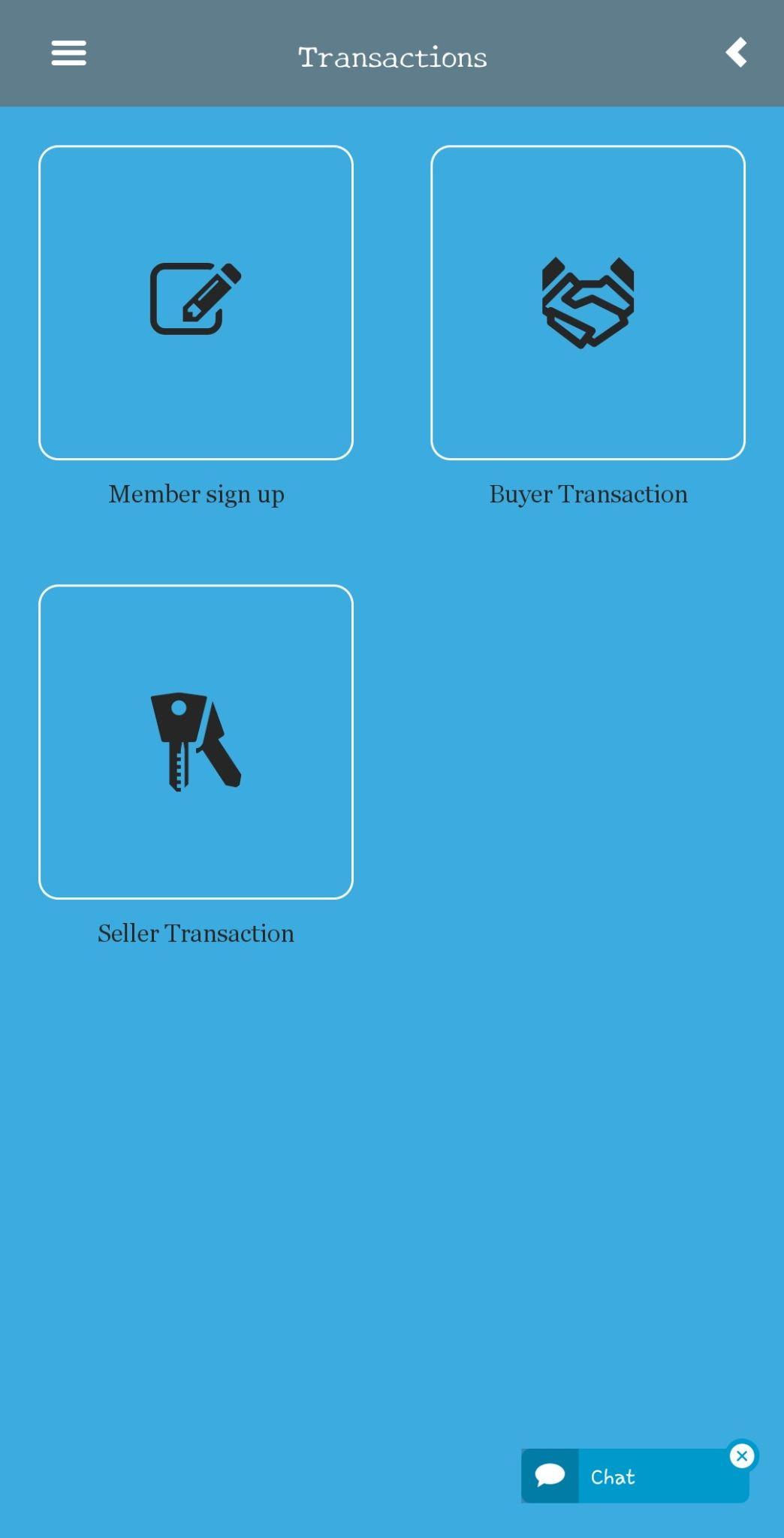
When the user chooses to sell, the same credentials are prompted but now asks how much the user wants to sell to Team Block’s energy bank. This adds energy back into the bank and successfully records the transaction in the system.



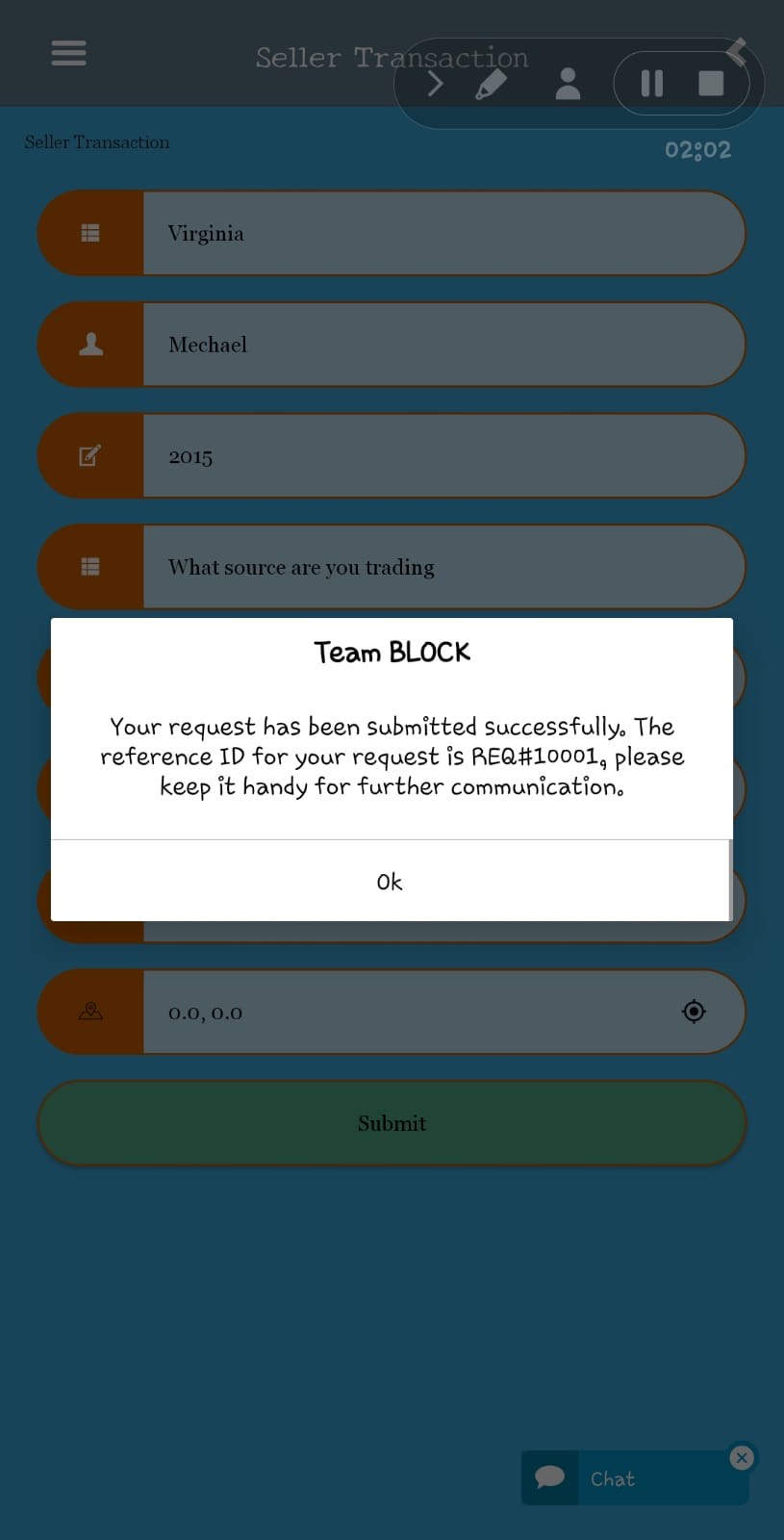
When the user finally wants to exit the program, the system stops and displays the current blockchain.

**Graphical User Interface**

This is the mobile application of Team block. It has three sections as we can see below in fig 1. The first section is for intending members or existing members to sign up and login respectively. The second is for interested buyers and the third is for interested sellers. Upon installation and use of this app, a notice for GPS enablement will be turned on. This is to confirm that your area is one of the area that Team Block energy company provides or buys energy from (the authorized areas are DC, MD, VA and AT).

 ****

***Fig 1- Mobile App Home Screen*** ***Fig 2- Enabled GPS setting***

** **

***Fig 3 - Buyer Transaction Details Fig 4- Confirmation Message***

**References**

<https://www.geeksforgeeks.org/create-simple-blockchain-using-python/amp/>

<https://www.sciencedirect.com/science/article/pii/S2352484721003164#fig1>

<https://www.ricoh.com/technology/tech/089_blockchain>

<https://replit.com/join/cnnuypduxd-codeboi301>