Ethereum Smart Contract Deployment Using Solidity

Bala Subramanyam Duggirala
School of Computing
University of Nebraska - Lincoln
Lincoln, USA
bduggirala2@huskers.unl.edu

Nhien Nguyen
School of Computing
University of Nebraska – Lincoln
Lincoln, USA
nguyen102@huskers.unl.edu

Abstract—A smart contract, which we will be implementing [6] [7] as part of this project, is a computer protocol using which the negotiation as well as the implementation of digital contracts can be facilitated, verified, and enforced, without a central authority [1]. After they were first introduced by Szabo in the 1990s [2], they did not see light until blockchain had emerged as a popular technology in which the Distributed Ledger Technology (public and append-only) and consensus mechanism play a key role in helping implement them in their true sense [1]. Although Bitcoin [3] is popular for being the first cryptocurrency to have supported basic smart contracts, Ethereum [4] takes credit to be the first public blockchain platform to support advanced contracts by using a Turing-complete virtual machine, a.k.a. Ethereum Virtual Machine (EVM) [1]. For this project, we plan to implement a simple Ethereum smart contract using Solidity [5], an object-oriented language developed for creating Ethereum smart contracts.

Keywords—Blockchain, Ethereum, Smart Contract, Solidity

1. INTRODUCTION

A contract is the traditional method of formalizing a relationship, and is a collection of promises agreed-to during a "meeting of the minds" [2]. A self-executing contract, also known as a "smart contract" is a computer program. The terms of an arrangement are directly encoded into lines of code in this program [8]. This smart contract is deployed on a distributed, decentralized blockchain network [9]. Szabo, a business scientist, and expert in cryptography, originally used the term "smart contract" in the middle of the 1990s. He characterized it as "a series of promises, written in digital form, including protocols within which the parties fulfill on these promise [1]." The fundamental principle of Bitcoin—sending and receiving money without a "trusted intermediary" like a bank in the middle—is expanded upon by smart contracts.

2. BACKGROUND

We now discuss the background about the smart contracts and their Blockchain, i.e., Ethereum, and also about a sibling Blockchain, Bitcoin.

2.1 Blockchain

A blockchain is a shared, publicly accessible database that is shared among numerous computers in a network. Data and information are kept in units called "blocks," which are grouped together in a sequential order. A digital currency transaction information is what is usually included in a block. The term "chain" describes how each block contains a cryptographic reference to its parent. Blocks are therefore connected. A block's data cannot be changed without also altering all blocks that come after it, which would require network-wide agreement. Each new block and the chain must be approved by every computer in the network. "Nodes" are the name given to these computers. Nodes guarantee that everyone using the blockchain has access to the same data. A consensus mechanism is required for blockchains to implement this distributed agreement [13]. Fig 2.a below gives the flow diagram of how a blockchain works.

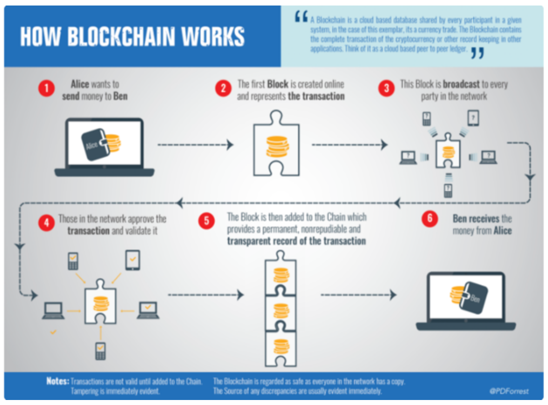

HOW BLOCKCHAIN WORKS
A Blockchain is a cloud based database shared by every participant in a given system, in the case of this exemplar, its a currency trade. The Blockchain contains the complete transaction of the cryptocurrency or other record keeping in other applications. Think of it as a cloud based peer to peer ledger.
1 Alice wants to send money to Ben
Alice
2 The first Block is created online and represents the transaction
3 This Block is broadcast to every party in the network
4 Those in the network approve the transaction and validate it
5 The Block is then added to the Chain which provides a permanent, nonrepudiable and transparent record of the transaction
6 Ben receives the money from Alice
Ben
Notes: Transactions are not valid until added to the Chain. Tampering is immediately evident.
The Blockchain is regarded as safe as everyone in the network has a copy. The Source of any discrepancies are usually evident immediately.
@PDForrest

Fig 2.a : How blockchain work [14]

2.2 Ethereum

The first blockchain platform to support smart contracts was Ethereum [12]. It is thought to have been created by Vitalik Buterin, who in 2014 released a white paper introducing it. In 2015, Buterin and Joe Lubin, the creators of the blockchain software firm ConsenSys, introduced the Ethereum platform.

Ethereum realizes blockchain technology's full potential, beyond merely providing the secure virtual payment method. Ether (ETH) has grown to become the second-largest cryptocurrency by market value since the introduction of Ethereum. Only Bitcoin outranks it [16]. Ethereum uses a consensus algorithm based on proof-of-stake. Staking at least 32 ETH into the deposit contract and running validator software are the requirements for anyone who wants to add new blocks to the chain. They can then be chosen at random to submit blocks for review and addition to the network by further validators.

Normally, there is just one chain in this architecture, but network slowness and dishonest conduct might lead to many blocks existing at the same location close to the head of the chain. A fork-choice algorithm chooses a single canonical set of blocks to solve this problem. The blocks chosen to form the heaviest chain possible, where "heavy" refers to the quantity of validators who have approved the blocks (weighted by the ETH they have staked). Participants are highly encouraged through a system of rewards and penalties to be sincere and use the internet as much as they can [15].

A decentralized Peer-to-Peer (P2P) network of Ethereum clients acting as network nodes makes up an Ethereum blockchain network. Any node that can execute smart contracts, verify new transactions, and process chain updates is referred to as an Ethereum client. It is a sort of online enclave that is spread across tens of thousands of computers and other devices linked by the Ethereum P2P network. The Ethereum Virtual Machine, often known as the EVM, and the runtime environment used in the P2P network for smart contract execution are what are enclaved. The P2P network is depicted in the figure 2.b [17]:

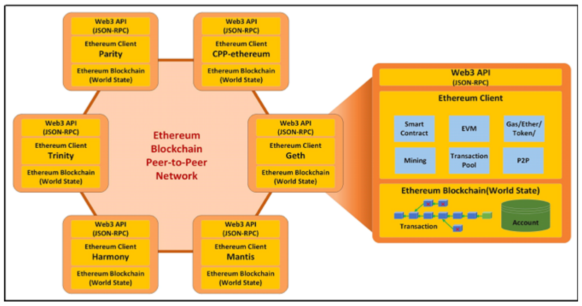


Fig 2: Ethereum Blockchain Peer-to-Peer Network [17]

2.3 Bitcoin

When centralized institutions failed the world in 2008, a person or group of persons going by the name Satoshi Nakamoto developed the Bitcoin protocol to decentralize the management of money. A collection of mathematical formulas known as the Bitcoin white paper described a new kind of distributed database known as the blockchain. The network debuted in January 2009 [18].

Bitcoin (BTC) cryptocurrency deployed on Bitcoin Network, just like (ETH) is, on Ethereum, eliminates the need for third parties to be involved in financial transactions by acting as a means of payment independent of any one person, group, or entity. It is available for purchase on numerous platforms and is given to blockchain miners as compensation for their efforts in verifying transactions [11]. Bitcoin employs peer-to-peer technology to operate without a central authority or banks; the network as a whole is responsible for handling transactions and issuing bitcoins. Since Bitcoin is an open-source project, anyone can participate, no one owns or controls it. Bitcoin enables fascinating uses that no other payment system could handle because of its special features [3]. Figure 2.c shows the steps of a bitcoin transaction on the blockchain.

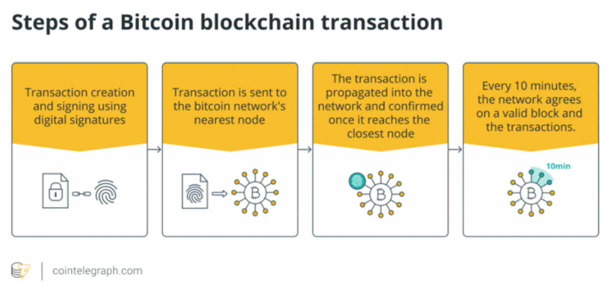

Steps of a Bitcoin blockchain transaction
Transaction creation and signing using digital signatures
Transaction is sent to the bitcoin network's nearest node
The transaction is propagated into the network and confirmed once it reaches the closest node
Every 10 minutes, the network agrees on a valid block and the transactions.
10min
cointelegraph.com

Fig 2.c: Steps of a Bitcoin blockchain transaction [18]

3. SMART CONTRACT – DEVELOPMENT AND DEPLOYMENT

3.1 Fundamental Design/Operation Mechanism

Since smart contracts are computer programs, simple "if/when...then" phrases typed into code and placed on a blockchain are how smart contracts operate. When predefined circumstances have been verified to have been met, a network of computers will carry out the actions. These can entail paying out money to the right people, registering a car, sending out notices, or writing a ticket. When the transaction is finished, the blockchain is then updated. As a result, the transaction cannot be modified, and only parties to whom permission have been granted can view the outcome.

As many conditions as are required to reassure the participants that the activity will be successfully accomplished can be included in a smart contract. Participants must agree on the "if/when...then" rules that govern those transactions, consider any potential exceptions, and design a framework for resolving disputes to set the terms. Participants must also decide how transactions and their data are represented on the blockchain. After that, a developer can construct the smart contract; however, more and more businesses using blockchain for commerce offer templates, web interfaces, and other online tools to make creating smart contracts easier [21]. Figure 3.a shows the operation mechanism of a smart contract.

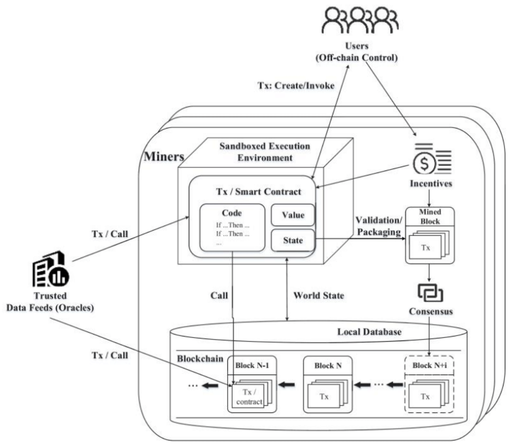


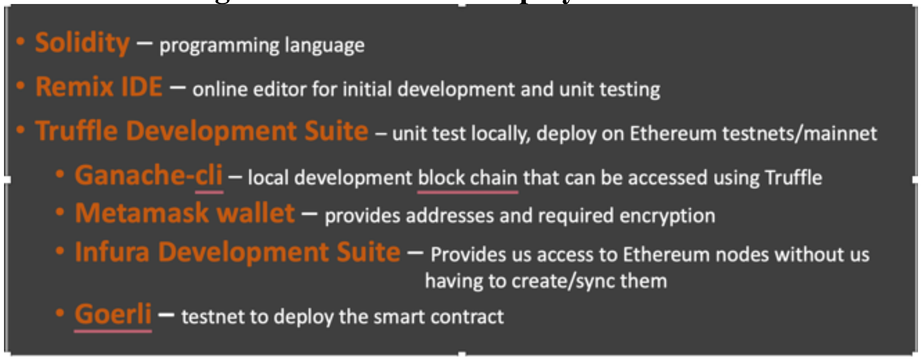
Fig 3.a: Operation mechanism of smart contract [20]

For our project, we deployed a "*Vending Machine*" smart contract [26]. The purpose of this smart contract is to be able to sell donuts that can be bought using Ether. It has functions like *getVendingMachineBalance(), purchase(), etc.,* which help us interact with the smart contract.

Here, the *purchase()* function can be used to demonstrate the power of Solidity, wherein by using the keyword "payable", this function now has the power to receive ETH payments. We then develop the function to evaluate any payments made, and vend virtual donuts accordingly only if sufficient payment is received.

3.2 Implementation Details

3.2.1 Technologies that we used to deploy our Smart Contract



- **Solidity** – programming language
- **Remix IDE** – online editor for initial development and unit testing
- **Truffle Development Suite** – unit test locally, deploy on Ethereum testnets/mainnet
  - **Ganache-cli** – local development block chain that can be accessed using Truffle
  - **Metamask wallet** – provides addresses and required encryption
  - **Infura Development Suite** – Provides us access to Ethereum nodes without us having to create/sync them
  - **Goerli** – testnet to deploy the smart contract

Ethereum smart contracts are mostly written in a programming language called *Solidity.* It is an object-oriented programming language that was specifically developed to develop Ethereum smart contracts.

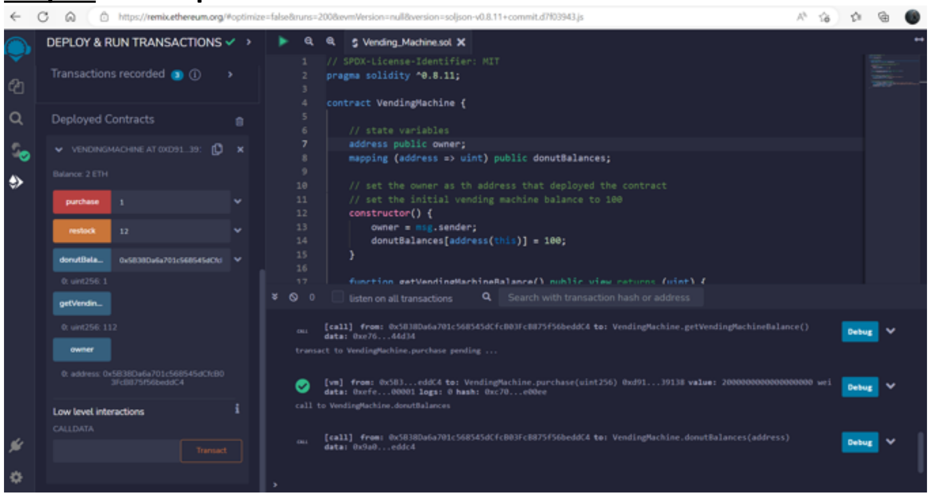
Remix IDE is an open-source program providing flexible access to numerous useful plugins and the convenience of user-friendly graphical user interfaces. Using the Solidity programming language, it can be the appropriate partner for engineers during the whole smart contract development lifecycle. Most importantly, Remix can also act as a viable training environment for mastering Ethereum. Remix IDE is accessible as a desktop or web program as well as a VS Code extension [19].

Truffle Development Suite is an environment that is designed for testing as well as deploying production-level smart contracts. It easily lets us test our smart contracts, deploy them on local test nets such as Ganache, and lets us configure our deployment network of choice to deploy the smart contract. It also lets us interact with our smart contract once it has been deployed.

Goerli is a blockchain test net that lets us deploy smart contracts and test them before deploying them to a L-3 production level blockchain. We chose Goerli since this is not only very popular, but also makes it easy to obtain some test Ether.

3.2.2 Work Flow

<u>Step 1:</u> Develop and test the Smart Contract on the Remix IDE



```
purchase  1
restock   12
donutBala...  0x5B38Da6a701c568545dCfcl
0: uint256: 1
getVendin...
0: uint256: 112
owner
0: address: 0x5B38Da6a701c568545dCfcB0
3FcB875f56beddC4
```

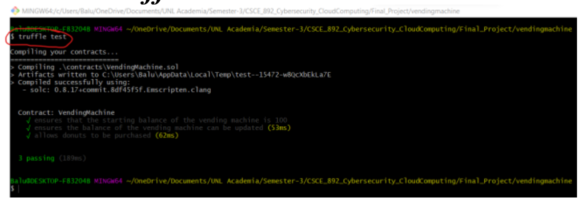
```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.11;

contract VendingMachine {

    // state variables
    address public owner;
    mapping (address => uint) public donutBalances;

    // set the owner as th address that deployed the contract
    // set the initial vending machine balance to 100
    constructor() {
        owner = msg.sender;
        donutBalances[address(this)] = 100;
    }

    function getVendingMachineBalance() public view returns (uint) {
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: VendingMachine.getVendingMachineBalance()
data: 0xe76...44d34
transact to VendingMachine.purchase pending ...

[vm] from: 0x5B3...eddC4 to: VendingMachine.purchase(uint256) 0xd91...39138 value: 2000000000000000000 wei
data: 0xefe...00001 logs: 0 hash: 0xc70...e00ee
call to VendingMachine.donutBalances

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: VendingMachine.donutBalances(address)
data: 0x9a0...eddc4
```

As shown above, the Remix IDE lets us develop and test our smart contract. For our Vending Machine Smart Contract, we could easily test all of our functions because Remix readily deploys our smart contract onto one if its simple test nets, and gives us an interface to access our functions, as shown on the left-hand side of the screenshot.

<u>Step 2:</u> Since we have now developed and tested our smart contract on Remix IDE, we will now test it using the Truffle suite, which lets us interact with our smart contract mimicking a production environment unlike Remix.

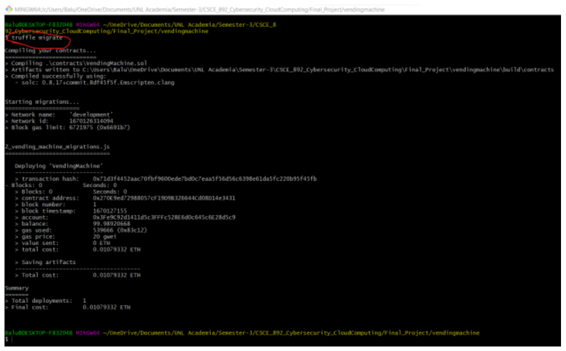
*A. : **Truffle Suite – TESTING***



```
MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
Balu@DESKTOP-F832048 MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle test

Compiling your contracts...
===========================
> Compiling .\contracts\VendingMachine.sol
> Artifacts written to C:\Users\Balu\AppData\Local\Temp\test--15472-w8QcXbEkLa7E
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang


  Contract: VendingMachine
    √ ensures that the starting balance of the vending machine is 100
    √ ensures the balance of the vending machine can be updated (53ms)
    √ allows donuts to be purchased (62ms)


  3 passing (189ms)

Balu@DESKTOP-F832048 MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ |
```

truffle test is a command that lets us test our tailor-made test cases as shown above. For our project, we checked to see if our vending machine is starting with the correct balance, is able to vend donuts, and is finally updating the donut balance of the vending machine after the transaction is completed.

B. Truffle Suite – Deploying on a Local Block Chain, Ganache



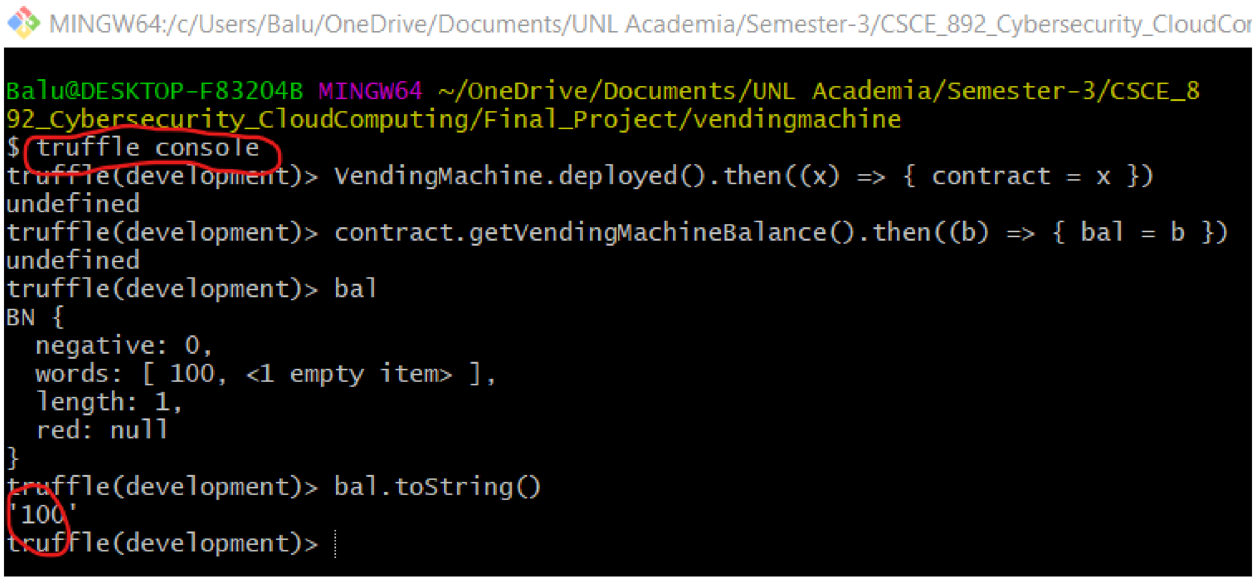
MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine

```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_8
92_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle migrate

Compiling your contracts...
===========================
> Compiling .\contracts\VendingMachine.sol
> Artifacts written to C:\Users\Balu\OneDrive\Documents\UNL Academia\Semester-3\CSCE_892_Cybersecurity_CloudComputing\Final_Project\vendingmachine\build\contracts
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang


Starting migrations...
======================
> Network name:    'development'
> Network id:      1670126314094
> Block gas limit: 6721975 (0x6691b7)


2_vending_machine_migrations.js
===============================

   Deploying 'VendingMachine'
   --------------------------
   > transaction hash:    0x71d3f4452aac70fbf9600ede7bd0c7eaa5f56d56c6398e61da5fc220b95f45fb
 - Blocks: 0            Seconds: 0
   > Blocks: 0            Seconds: 0
   > contract address:    0x270E9ed72988057cF19D9B326644Cd08D14e3431
   > block number:        1
   > block timestamp:     1670127155
   > account:             0x3Fe9C92d1411d5c3FFFc528E6d0c645c6E28d5c9
   > balance:             99.98920668
   > gas used:            539666 (0x83c12)
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01079332 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01079332 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.01079332 ETH


Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$
```

Ganache is a personal Ethereum blockchain used in Truffle that is used for rapid Ethereum development. Ganache can be used all through the development lifecycle of a smart contract, from initial development to unit testing, to deployment and test interactions. Ganache-UI and Ganache-CLI are its two flavors [Ganache | Overview - Truffle Suite]. We have used *ganache-cli* for our project.

Once we configure the configuration.js file using the Ethereum network of our choice (Ganache in this case), we can use the "*truffle migrate"* command to deploy our smart contract on to the blockchain network.

C. *Truffle Suite – Interacting with our Smart Contract on Ganache*



MINGW64:/c/Users/Balu/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersecurity_CloudCor

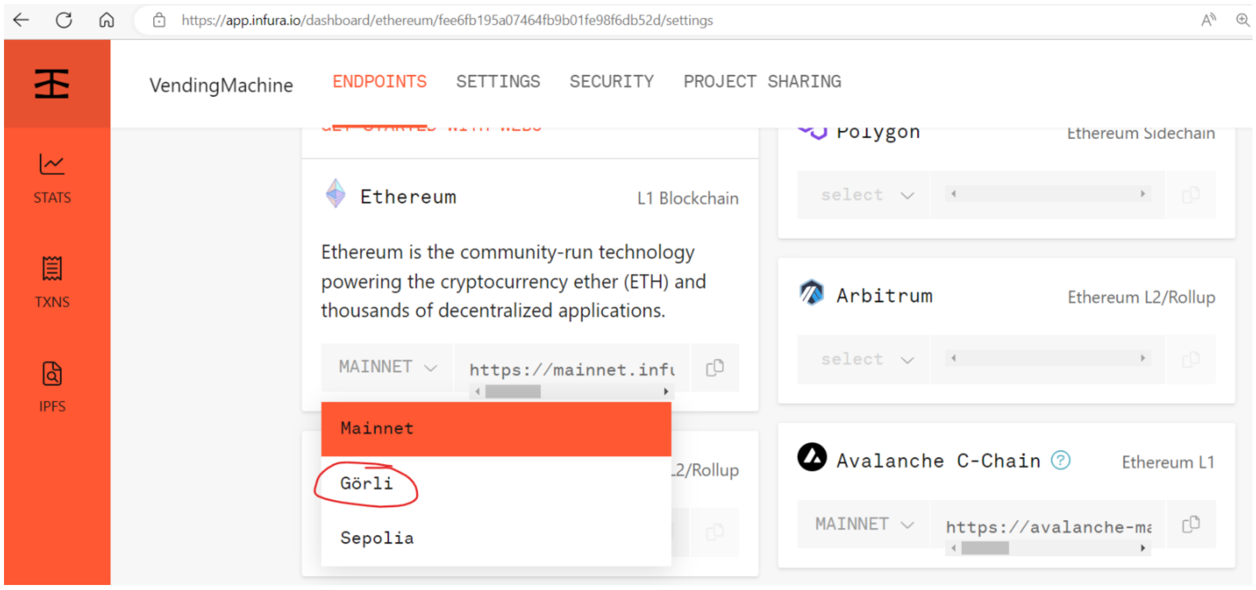
```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_8
92_Cybersecurity_CloudComputing/Final_Project/vendingmachine
$ truffle console
truffle(development)> VendingMachine.deployed().then((x) => { contract = x })
undefined
truffle(development)> contract.getVendingMachineBalance().then((b) => { bal = b })
undefined
truffle(development)> bal
BN {
  negative: 0,
  words: [ 100, <1 empty item> ],
  length: 1,
  red: null
}
truffle(development)> bal.toString()
'100'
truffle(development)>
```

Once the smart contract is deployed, Truffle lets us interact with it by providing us an interaction interface using "*truffle console*". Using this, as can be seen in the above screenshot, we can interact with our smart contract using its various functions that we developed. For this example, we try to get the donut balance of our vending machine.

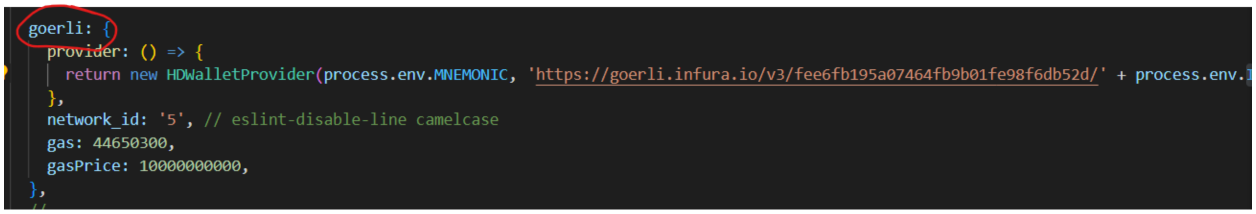
D. *Truffle Suite – Deploying on an Ethereum Test Network (Goerli):*

In order to deploy a smart contract on an Ethereum network (either test network or even L3 main net), we first need to sync our computer to one of its nodes, a.k.a. end points.

But doing so requires high-speed computer infrastructure with good memory. It also takes significant time in order to sync to one of these nodes. In order to readily gain access to one of such Ethereum nodes, we use *Infura. infura.io* provides high-speed and highly available Ethereum infrastructure. All we need to do is choose the network of our choice from a wide variety of networks available and use the end point of the network of our choice to configure the truffle_config.js file. The screenshots for these two steps are given below. In our case, we tried to use Goerli testnet.

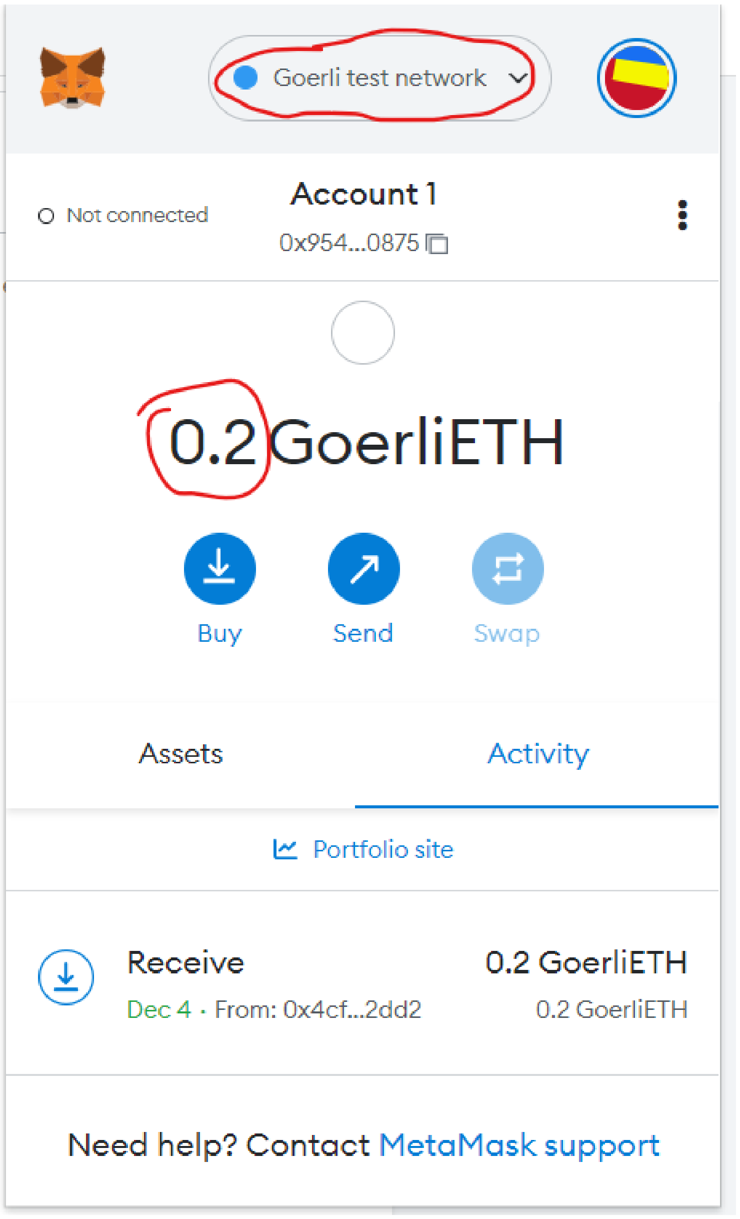


Ethereum is the community-run technology powering the cryptocurrency ether (ETH) and thousands of decentralized applications.



```
goerli: {
  provider: () => {
    return new HDWalletProvider(process.env.MNEMONIC, 'https://goerli.infura.io/v3/fee6fb195a07464fb9b01fe98f6db52d/' + process.env.
  },
  network_id: '5', // eslint-disable-line camelcase
  gas: 44650300,
  gasPrice: 10000000000,
},
```

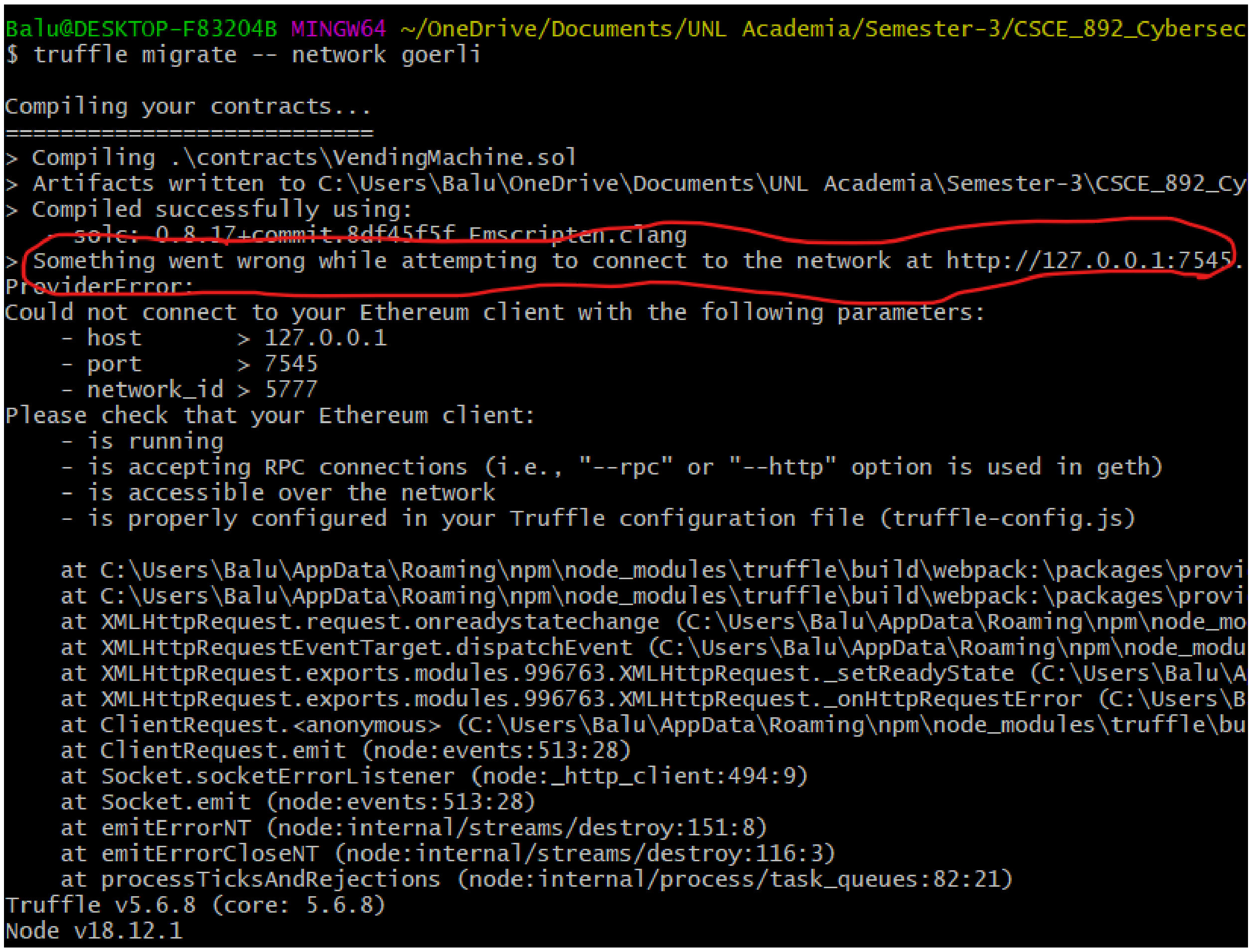
Also, since the real-world deployments require us to have our own account (unlike the disposable virtual accounts provided by the Remix IDE or Ganache test net), we need to set up accounts of our own using a blockchain wallet like Metamask, which would give us accounts encrypted using mnemonics and private keys, that can be easily configured in truffle-config.js.



Once all these have been setup, deploying the smart contract and interacting with it is same as how it was done with ganache test net. In order to do so, all we need to do is set up the truffle-config.js file to connect to the test network of our choice, and use "—network our_choice" while using "truffle migrate" command (by default, it deploys in development networks like ganache).

Once the smart contract is deployed, we can keep track of our transactions using "*etherscan.io"* website.

Note: We faced some errors during the deployment of our smart contract to the Goerli testnet and were unable to resolve the error. Rest assured; the procedure is very similar to how we do it for the ganache test network. Based on our research, we found out that it had something to do with the *Infura*'s nodes, and hence were unable to resolve it. Although we hoped to try and deploy it to the *Rinkeby* test network, *Infura* has recently deprecated support for Rinkeby, and left us with no options that are free.



```
Balu@DESKTOP-F83204B MINGW64 ~/OneDrive/Documents/UNL Academia/Semester-3/CSCE_892_Cybersec
$ truffle migrate -- network goerli

Compiling your contracts...
===========================
> Compiling .\contracts\VendingMachine.sol
> Artifacts written to C:\Users\Balu\OneDrive\Documents\UNL Academia\Semester-3\CSCE_892_Cy
> Compiled successfully using:
   - solc: 0.8.17+commit.8df45f5f.Emscripten.clang
> Something went wrong while attempting to connect to the network at http://127.0.0.1:7545.
ProviderError:
Could not connect to your Ethereum client with the following parameters:
    - host       > 127.0.0.1
    - port       > 7545
    - network_id > 5777
Please check that your Ethereum client:
    - is running
    - is accepting RPC connections (i.e., "--rpc" or "--http" option is used in geth)
    - is accessible over the network
    - is properly configured in your Truffle configuration file (truffle-config.js)

    at C:\Users\Balu\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\provi
    at C:\Users\Balu\AppData\Roaming\npm\node_modules\truffle\build\webpack:\packages\provi
    at XMLHttpRequest.request.onreadystatechange (C:\Users\Balu\AppData\Roaming\npm\node_mo
    at XMLHttpRequestEventTarget.dispatchEvent (C:\Users\Balu\AppData\Roaming\npm\node_modu
    at XMLHttpRequest.exports.modules.996763.XMLHttpRequest._setReadyState (C:\Users\Balu\A
    at XMLHttpRequest.exports.modules.996763.XMLHttpRequest._onHttpRequestError (C:\Users\B
    at ClientRequest.<anonymous> (C:\Users\Balu\AppData\Roaming\npm\node_modules\truffle\bu
    at ClientRequest.emit (node:events:513:28)
    at Socket.socketErrorListener (node:_http_client:494:9)
    at Socket.emit (node:events:513:28)
    at emitErrorNT (node:internal/streams/destroy:151:8)
    at emitErrorCloseNT (node:internal/streams/destroy:116:3)
    at processTicksAndRejections (node:internal/process/task_queues:82:21)
Truffle v5.6.8 (core: 5.6.8)
Node v18.12.1
```

4. APPLICATIONS OF SMART CONTRACTS

4.1 Trade Finance

The adoption of smart contracts will be extremely beneficial for trade finance. According to *Santander Innoventures*, blockchain technology will result in annual savings of more than $20 billion [22]. Due to the irreversible, transparent, and trustless nature of blockchain and smart contract technology, decentralized finance (*DeFi*) *dApps* are becoming a more and more attractive alternative to conventional financial services. Lending, borrowing, trading, and a variety of other financial services are only a few of the services that *DeFi dApps* offer in addition to completely new kinds of products and decentralized business models that may be of great use and profit to consumers. *DApps* could lower the barriers to entry into the financial services industry for people all over the world thanks to the enhanced transparency provided by smart contracts (combined with 24/7 functionality and lower prices) [23].

4.2 Gaming:

The $100 billion global gaming business is a rapidly expanding ecosystem, yet the way value is created and shared across the sector may sometimes be unfair. Games are created and released by developers, and players pay to access and participate in those games. As a result, gamers continue to pay money to gain access to in-game resources and gaming options, perpetuating a one-way flow of value. Contrarily, blockchain technology in gaming can help gamers more effectively realize the value and utility of in-game transactions and asset acquisitions [23].

4.3 Real Estate

By fusing blockchain and real estate transactions, smart contracts are increasing fractional ownership of assets and lowering the entry barrier for investing for many. There have been a few successful attempts to tokenize real estate assets, including through real estate and blockchain-based platforms like RealT and SolidBlock. By adding blockchain into real estate deals, smart contract technology can also remodel the documentation and transaction processes. For instance, a blockchain-based land title register is being developed in the Republic of Georgia (in the Caucasus region) since 2016, and similar initiatives are being carried out in other countries as the United Arab Emirates (UAE) [23].

4.4 Emerging Technology

The capacity of blockchain technology and related smart contract technology to simplify difficult computing operations like those involved in machine learning and artificial intelligence is one of the most interesting uses of these technologies (AI). There is potential to develop AI-powered smart contracts by fusing the data-intensive processing of AI with the decentralized security and immutability of blockchain technology. Applications for smart contracts will need to develop into increasingly complicated systems as they are adopted by more and more sectors of the economy. Simple smart contract use cases can be created manually, but AI-enabled smart contracts may make it possible to create dApps and highly complicated, responsive smart contracts that have the potential to significantly increase the possibilities of the technology [23].

4.5 Insurance

Tens of millions of dollars are spent annually by the insurance industry on claim processing. Additionally, it loses millions of dollars as a result of false claims. Aside from supporting the initial insurance policy, smart contracts also significantly enhance the claim processing procedure. They may be able to permit mistake checks and choose payout amounts in accordance with a set of standards that take the type of policy that the person or business held into consideration.

Once more, the key advantages of smart contract technology are shorter processing times, lower prices, and far fewer errors. In the long run, pay-as-you-go insurance policies and the quick activation of claims following an accident could be made possible by smart contracts working in conjunction with Internet of Things-enabled automobiles. Driver's licenses, driving records, accident reports, and policy information may all be processed instantly to enable quick reimbursements that would be advantageous to both parties [22].

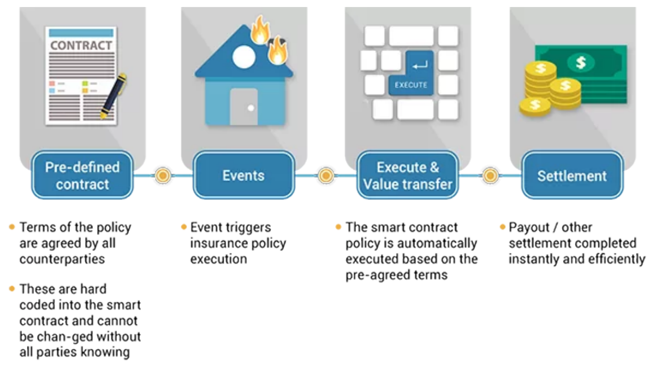

CONTRACT
EXECUTE
Pre-defined contract
Events
Execute & Value transfer
Settlement
Terms of the policy are agreed by all counterparties
These are hard coded into the smart contract and cannot be chan-ged without all parties knowing
Event triggers insurance policy execution
The smart contract policy is automatically executed based on the pre-agreed terms
Payout / other settlement completed instantly and efficiently

Fig 4.5: Smart contract use in Insurance [22].

4.6 Medical Research

Similar advantages will apply to the medical research sector as they do to the healthcare sector. To begin with, after being securely encrypted using blockchain technology, extremely sensitive data like patient records might be transmitted between departments or research institutions. It is crucial to keep these records secure because a lot of the people taking part in medical research have delicate medical issues that they frequently want to keep confidential.

The same is true for the vast amounts of data that pharmaceutical research organizations must safeguard, including test findings and new drug formulations. If they must disclose any of this information to a third party for any reason, these might be protected through the use of smart contracts. Only one smart contract blockchain example exists that has the potential to significantly help the medical research sector [22].

5. ADVANTAGES AND DISADVANTAGES OF SMART CONTRACT

5.1 Advantages

Accuracy

Explicit recording of all terms and conditions is one of the key prerequisites for a smart contract. This is necessary because omitting it could lead to transaction issues. Automated contracts avoid the hazards of manually filling out a ton of paperwork as a result [24].

Transparency

All parties who may be affected by these contracts have full visibility of and access to their terms and conditions. Once the contract is in place, there is no way to challenge it. This makes the transaction completely transparent to all parties involved [24].

Effective Communication

Everything is explicit since the contract needs to be detailed accurately. Miscommunication or misinterpretation cannot be an option. Therefore, efficiency lost due to communication breakdowns can be significantly reduced via smart contracts [24].

Speed

These contracts are internet-based contracts that are run by computer code. They are therefore able to complete transactions quite rapidly. Many conventional business processes can be shortened by hours with this speed. Processing documents manually is not necessary [24].

Security

Automated contracts employ the highest data encryption standard currently available, the same one that contemporary crypto currencies do. They rank among the most secure things on the internet thanks to this level of security [24].

Efficiency

The effectiveness of these contracts comes because of how quickly and precisely they are written. More value-generating transactions are executed per unit of time because of higher efficiency [24].

Paper Free

Businesses across the world are becoming more aware of their environmental impact. The "go-green" movement is made possible by smart contracts because they are fully functional in the virtual environment. Large reams of paper are no longer required as a result [24].

Backup & Storage

These contracts document crucial information about each transaction. Therefore, whenever your information is utilized in a contract, it is permanently preserved for use in the future. These properties can be quickly recovered in the event of data loss [24].

Savings

The absence of a lengthy chain of middlemen is arguably one of the most important benefits of automated contracts. Lawyers, witnesses, banks, and other middlemen are not required [24].

Trust

The execution of smart contracts inspires complete confidence. Any chance of manipulation, bias, or error is eliminated by the agreement's openness, independence, and security. The contract is then automatically carried out by the network after being solemnized [24].

Guaranteed Results (Bonus)

The possibility to greatly minimize or perhaps completely remove the need for litigation and judicial proceedings may be another appealing aspect of these contracts.

Parties agree to abide by the laws and judgments of the underlying code by employing a self-executing contract [24].

5.2 Disadvantages

Seeking legal help is difficult

Most conventional text contracts are binding agreements. Contractual parties have the option to seek legal assistance to resolve problems if they arise. But legal professionals are of little assistance when it comes to smart contracts. They cannot check the code-only agreements to see if there was a contract breach. Additionally, because blockchain and smart contracts are still relatively new concepts, a court may find it challenging to resolve disagreements involving them. Difficult to Amend or Terminate Smart Contracts [25].

Difficult to amend or terminate smart contracts

Multiple signatures are needed to complete a transaction in *multisig* (short for multi-signature) smart contracts. For instance, only when several people have given their consent to the transaction's execution may money be sent from Party X to Party Y. Even if all parties agree to alter their smart contract, they could not really succeed in doing so. It is as a result of blockchains' immutability. Additionally, attempting to change the code can raise transaction costs and risk of human error. Similar to the above, a party cannot cancel a smart contract if they discover an inaccuracy that is detrimental to their interests [25].

Smart contract vulnerabilities not easy to fix

Smart contracts have caught the attention of malevolent users as they now enable transactions worth millions of dollars. Additionally, due to design problems in smart contracts, the majority of them are susceptible to hacking. Once the smart contract is kept on a blockchain, it cannot be fixed for these errors. It's because the data in the blockchain cannot be changed or altered [25].

6. CONCLUSION AND FUTURE WORK

As shown below, smart contracts have many applications and are now on-track to scale even further. For our project, our goal was to get to know what a smart contract is, how to develop it, test and deploy it. In the process, we also understood various deployment details and have become familiar with multiple blockchain environments, and real-world infrastructure like different wallets, tracking websites, node-providers etcs., As part of future work, we intend to explore "Solidity" even more to discover its various powerful features, and develop, test and deploy smart contracts that are much more complex than the *Vending Machine*.

7. REFERENCESss

[1] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han and F. -Y. Wang, "Blockchain-Enabled Smart Contracts: Architecture, Applications, and Future Trends," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 11, pp. 2266-2277, Nov. 2019, doi: 10.1109/TSMC.2019.2895123.

[2] Smart Contracts: Building Blocks for Digital Markets. [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/ Information-InSpeech/CDROM/Literature/LOTwinterschool2006/szabo. best.vwh.net/smart_contracts_2.html

[3] *Open source P2P money*. Bitcoin. (n.d.). Retrieved December 9, 2022, from https://bitcoin.org/.

[4] *Home.* ethereum.org. (n.d.). Retrieved December 9, 2022, from https://www.ethereum.org/.

[5] *Solidity*ℑ. Solidity. (n.d.). Retrieved December 9, 2022, from http://solidity.readthedocs.io/en/latest/

[6] YouTube. (2017, November 20). *Smart contracts - simply explained*. YouTube. Retrieved December 9, 2022, from https://www.youtube.com/watch?v=ZE2HxTmxfrI

[7] YouTube. (2021, December 31). *Smart contracts 101 - create a simple Ethereum Smart contract with solidity*. YouTube. Retrieved December 9, 2022, from https://www.youtube.com/watch?v=bNXJNeaYl8Q

[8] *Real World examples of smart contracts*. Gemini. (n.d.). Retrieved December 9, 2022, from https://www.gemini.com/cryptopedia/smart-contract-examples-smart-contract-use-cases

[9] Frankenfield, J. (2022, September 16). *What are smart contracts on the blockchain and how they work*. Investopedia. Retrieved December 9, 2022, from https://www.investopedia.com/terms/s/smart-contracts.asp

[10] Coinbase. (n.d.). *Smart contracts allow developers to build apps that take advantage of blockchain security, reliability, and accessibility.* Coinbase. Retrieved December 9, 2022, from https://www.coinbase.com/learn/crypto-basics/what-is-a-smart-contract

[11] Frankenfield, J. (2022, November 23). *What is bitcoin? how to mine, buy, and use it*. Investopedia. Retrieved December 9, 2022, from https://www.investopedia.com/terms/b/bitcoin.asp

[12] Gupta, R. (2020, November 20). *Are blockchain smart contracts legally binding?* Market Realist. Retrieved December 9, 2022, from https://marketrealist.com/p/which-blockchains-support-smart-contracts/

[13] *Intro to ethereum*. ethereum.org. (n.d.). Retrieved December 9, 2022, from https://ethereum.org/en/developers/docs/intro-to-ethereum/

[14] Belgium, O. (n.d.). *Blockchain introduction - Ken Coenen, Jeroen de Prest and Kevin Leyssens*. Ordina. Retrieved December 9, 2022, from https://ordina-jworks.github.io/blockchain/2017/05/10/Blockchain-Introduction.html

[15] Belgium, O. (n.d.). *Blockchain introduction - Ken Coenen, Jeroen de Prest and Kevin Leyssens*. Ordina. Retrieved December 9, 2022, from https://ordina-jworks.github.io/blockchain/2017/05/10/Blockchain-Introduction.html

[16] Frankenfield, J. (2022, October 20). *What is ethereum and how does it work?* Investopedia. Retrieved December 9, 2022, from https://www.investopedia.com/terms/e/ethereum.asp#:~:text=Ethereum%20is%20a%20blockchain%2Dbased,long%2Dterm%20visions%20and%20limitations.

[17] Makers, D. C. W. (n.d.). *Coding bootcamps in Virginia*. Coding Bootcamps. Retrieved December 9, 2022, from https://coding-bootcamps.com/blog/ethereum-architecture-and-components.html

[18] Cointelegraph. (2022, March 14). *What is the bitcoin blockchain? A guide to the technology behind BTC*. Cointelegraph. Retrieved December 9, 2022, from https://cointelegraph.com/bitcoin-for-beginners/how-does-blockchain-work-a-beginners-guide-to-blockchain-technology#:~:text=Bitcoin%20technology%20uses%20peer%2Dto,going%20through%20any%20financial%20institution.

[19] *Real World examples of smart contracts*. Gemini. (n.d.). Retrieved December 9, 2022, from https://www.gemini.com/cryptopedia/smart-contract-examples-smart-contract-use-cases

[20] *Figure 2. operational mechanism of smart contract - researchgate*. (n.d.). Retrieved December 9, 2022, from https://www.researchgate.net/figure/Operational-mechanism-of-smart-contract_fig2_342850916

[21] *What are smart contracts on Blockchain?* IBM. (n.d.). Retrieved December 9, 2022, from https://www.ibm.com/topics/smart-contracts

[22] Davies, A. (2022, December 8). *What are 10 smart contracts use cases? I DevTeam.space*. DevTeam.Space. Retrieved December 9, 2022, from https://www.devteam.space/blog/what-are-10-smart-contracts-use-cases/

[23] *Real World examples of smart contracts*. Gemini. (n.d.). Retrieved December 9, 2022, from https://www.gemini.com/cryptopedia/smart-contract-examples-smart-contract-use-cases

[24] ChainTrade. (2017, December 27). *10 advantages of using smart contracts*. Medium. Retrieved December 11, 2022, from https://medium.com/@ChainTrade/10-advantages-of-using-smart-contracts-bc29c508691a

[25] Lucas. (2021, December 13). *What are the disadvantages of smart contracts?* WhalesHeaven. Retrieved December 11, 2022, from https://articles.whalesheaven.com/what-are-the-disadvantages-of-smart-contracts/

[26] **Vending Machine code taken from** https://github.com/jspruance/block-explorer-tutorials/blob/main/smart-contracts/solidity/VendingMachine.sol

[27] **Reference for Truffle Code Base:** Deploy a smart contract to Ethereum using Truffle - A step-by-step guide. – YouTube