DA51 Lab Session 1: Creating a Blockchain

Description:

This lab session guides us through creating a basic blockchain using MultiChain, a platform for private blockchains. We will learn how to install MultiChain, create a blockchain, and obtain information about it using the command line. We will also test blockchain operability by creating a transaction and checking balances across nodes.

Question 6:

```
PS C:\Users\jules> multichain-util create myChain

MultiChain 2.3.3 Utilities (latest protocol 20013)

Blockchain parameter set was successfully generated.
You can edit it in C:\Users\jules\AppData\Roaming\MultiChain\myChain\params.dat before running multichaind for the first time.

To generate blockchain please run "multichaind myChain -daemon".
```

Question 9:

The ‘multichaind myChain -daemon’ command start and mine the first block called the **genesis block**. This node is accessible through the address: myChain@172.23.144.1:9259. This link is made with the name of the chain, the local Ip address of my computer and the port that the node is listening on. In this case, this Ip come from the ethernet adapter for WSL. So, the previous command tells me that I can connect the node with this link also myChain@172.23.36.20:9259

```
PS C:\Users\jules> multichaind myChain -daemon

MultiChain 2.3.3 Daemon (Community Edition, latest protocol 20013)

Looking for genesis block...
Genesis block found

Other nodes can connect to this node using:
multichaind myChain@172.23.144.1:5779

This host has multiple IP addresses, so from some networks:
multichaind myChain@172.23.36.20:5779

Listening for API requests on port 5778 (local only - see rpcallowip setting)

Node ready.
```

Question 13:

When I try to connect to the blockchain for the first time, the blockchain reject it because in the configuration file ‘anyone-can-connect’ is set to ‘false’.

Command:

```
jules@TABLET-TQGU8LI5:~$ multichaind myChain@172.23.144.1:6837

MultiChain 2.3.3 Daemon (Community Edition, latest protocol 20013)

Retrieving blockchain parameters from the seed node 172.23.144.1:6837 ...
Blockchain successfully initialized.

Please ask blockchain admin or user having activate permission to let you connect and/or transact:
multichain-cli myChain grant 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs connect
multichain-cli myChain grant 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs connect,send,receive
```

Question 15:

As feedback, I get the json inserted in configuration file and a unique ID which represent this action.

```
PS C:\Users\jules> multichain-cli myChain grant 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs connect,send,receive
{"method":"grant","params":["1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs","connect,send,receive"],"id":"72508621-1726057947","chain_name":"myChain"}

ecdbf4247ac28722b895119f7911fe673f39128a5b4a8422e8ded35815a9d220
```

Question 17:

At this point of the lab, I create the first node using –deamon parameter and I connect two other nodes using WSL and -datadir option.

Question 18 – 30:

Command feedback:

- multichain-cli myChain getinfo

  Get the general information of the blockchain like node version, the port, etc. and the software used to run the chain like the version and the edition

```
PS C:\Users\jules> multichain-cli myChain getinfo
{"method":"getinfo","params":[],"id":"71604968-1726064516","chain_name":"myChain"}

{
    "version" : "2.3.3",
    "nodeversion" : 20303901,
    "edition" : "Community",
    "protocolversion" : 20013,
    "chainname" : "myChain",
    "description" : "MultiChain myChain",
    "protocol" : "multichain",
    "port" : 5779,
    "setupblocks" : 60,
    "nodeaddress" : "myChain@172.23.144.1:5779",
    "burnaddress" : "1XXXXXXWkXXXXXXXMJXXXXXXbqXXXXXXY23Ai8",
    "incomingpaused" : false,
    "miningpaused" : false,
    "offchainpaused" : false,
    "walletversion" : 60000,
    "balance" : 0,
    "walletdbversion" : 3,
    "reindex" : false,
    "blocks" : 24,
    "chainrewards" : 0,
    "streams" : 1,
    "timeoffset" : 0,
    "connections" : 3,
    "proxy" : "",
    "difficulty" : 5.960464477753906e-8,
    "testnet" : false,
    "keypoololdest" : 1726064440,
    "keypoolsize" : 2,
    "paytxfee" : 0,
    "relayfee" : 0,
    "errors" : ""
}
```

- multichain-cli myChain help

  Get a list of all command available on a chain

```
PS C:\Users\jules> multichain-cli myChain help
{"method":"help","params":[],"id":"46214789-1726064517","chain_name":"myChain"}

== Blockchain ==
addlibraryupdate "library-identifier" "update-name" "javascript-code"
addlibraryupdatefrom "from-address" "library-identifier" "update-name" "javascript-code"
getassetinfo "asset-identifier" ( verbose )
getbestblockhash
getblock "hash"|height ( verbose )
getblockchaininfo
getblockcount
getblockhash index
getchaintips
getdifficulty
getfiltercode "filter-identifier"
getlastblockinfo ( skip )
getlibrarycode "library-identifier" ( "update-name" )
getmempoolinfo
getrawmempool ( verbose )
getstreaminfo "stream-identifier" ( verbose )
gettokeninfo "asset-identifier" "token-identifier" ( verbose )
gettxout "txid" n ( includemempool )
gettxoutsetinfo
getvariablehistory "variable-identifier" ( verbose count start )
getvariableinfo "variable-identifier" ( verbose )
getvariablevalue "variable-identifier"
listassetissues "asset-identifier" ( verbose count start )
listassets ( asset-identifier(s) verbose count start )
listblocks block-set-identifier ( verbose )
listlibraries ( library-identifier(s) verbose )
listminers ( verbose )
listpermissions ( "permission(s)" address(es) verbose )
liststreamfilters ( filter-identifier(s) verbose )
liststreams ( stream-identifier(s) verbose count start )
listtxfilters ( filter-identifier(s) verbose )
listupgrades (upgrade-identifier(s))
listvariables ( variable-identifier(s) verbose count start )
runstreamfilter "filter-identifier" ( "tx-hex"|"txid" vout )
runtxfilter "filter-identifier" ( "tx-hex"|"txid" )
setvariablevalue "variable-identifier" ( value )
setvariablevaluefrom "from-address" "variable-identifier" ( value )
testlibrary ( "library-identifier"  "update-name" "javascript-code")
teststreamfilter restrictions "javascript-code" ( "tx-hex"|"txid" vout )
testtxfilter restrictions "javascript-code" ( "tx-hex"|"txid" )
verifychain ( checklevel numblocks )
verifypermission "address" "permission"
```

- multichain-cli myChain getblockchainparams

Get the content of the configuration file param.dat

```
PS C:\Users\jules> multichain-cli myChain getblockchainparams
{"method":"getblockchainparams","params":[],"id":"20827356-1726064518","chain_name":"myChain"}

{
    "chain-protocol" : "multichain",
    "chain-description" : "MultiChain myChain",
    "root-stream-name" : "root",
    "root-stream-open" : true,
    "chain-is-testnet" : false,
    "target-block-time" : 15,
    "maximum-block-size" : 8388608,
    "maximum-chunk-size" : 1048576,
    "maximum-chunk-count" : 1024,
    "default-network-port" : 5779,
    "default-rpc-port" : 5778,
    "anyone-can-connect" : false,
    "anyone-can-send" : false,
    "anyone-can-receive" : false,
    "anyone-can-receive-empty" : true,
    "anyone-can-create" : false,
    "anyone-can-issue" : false,
    "anyone-can-mine" : false,
    "anyone-can-activate" : false,
    "anyone-can-admin" : false,
    "support-miner-precheck" : true,
    "allow-arbitrary-outputs" : false,
    "allow-p2sh-outputs" : true,
    "allow-multisig-outputs" : true,
    "setup-first-blocks" : 60,
    "mining-diversity" : 0.3,
    "admin-consensus-upgrade" : 0.5,
    "admin-consensus-txfilter" : 0.5,
    "admin-consensus-admin" : 0.5,
    "admin-consensus-activate" : 0.5,
    "admin-consensus-mine" : 0.5,
    "admin-consensus-create" : 0,
    "admin-consensus-issue" : 0,
    "lock-admin-mine-rounds" : 10,
    "mining-requires-peers" : true,
    "mine-empty-rounds" : 10,
    "mining-turnover" : 0.5,
    "first-block-reward" : -1,
    "initial-block-reward" : 0,
    "reward-halving-interval" : 52560000,
    "reward-spendable-delay" : 1,
    "minimum-per-output" : 0,
    "maximum-per-output" : 100000000000000,
    "minimum-offchain-fee" : 0,
    "minimum-relay-fee" : 0,
    "native-currency-multiple" : 100000000,
    "skip-pow-check" : false,
```

- multichain-cli myChain listpermissions

Get all the permissions granted with the address

```
PS C:\Users\jules> multichain-cli myChain listpermissions
{"method":"listpermissions","params":[],"id":"85442670-1726064519","chain_name":"myChain"}

[
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "mine",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "admin",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "activate",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "connect",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "send",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "receive",
        "startblock" : 0,
        "endblock" : 4294967295
    },
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "issue",
```

- multichain-cli myChain getaddresses

Get the addresses of the node, in this case the first folder created/the first node (genesis)

```
PS C:\Users\jules> multichain-cli myChain getaddresses
{"method":"getaddresses","params":[],"id":"85442670-1726064519","chain_name":"myChain"}

[
    "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
    "1EX2UKX7nwgmaAXAWkjaYiSEtV7FDKP8MZ2J3M"
]
```

- multichaind -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain –daemon

Connect to the first node from the second node

```
PS C:\Users\jules> multichaind -datadir="C:\dev\multi" -port=7730 -rpcport=7729 myChain -daemon

MultiChain 2.3.3 Daemon (Community Edition, latest protocol 20013)

Other nodes can connect to this node using:
multichaind myChain@172.23.144.1:7730

This host has multiple IP addresses, so from some networks:
multichaind myChain@172.23.36.20:7730

Listening for API requests on port 7729 (local only - see rpcallowip setting)

Node ready.
```

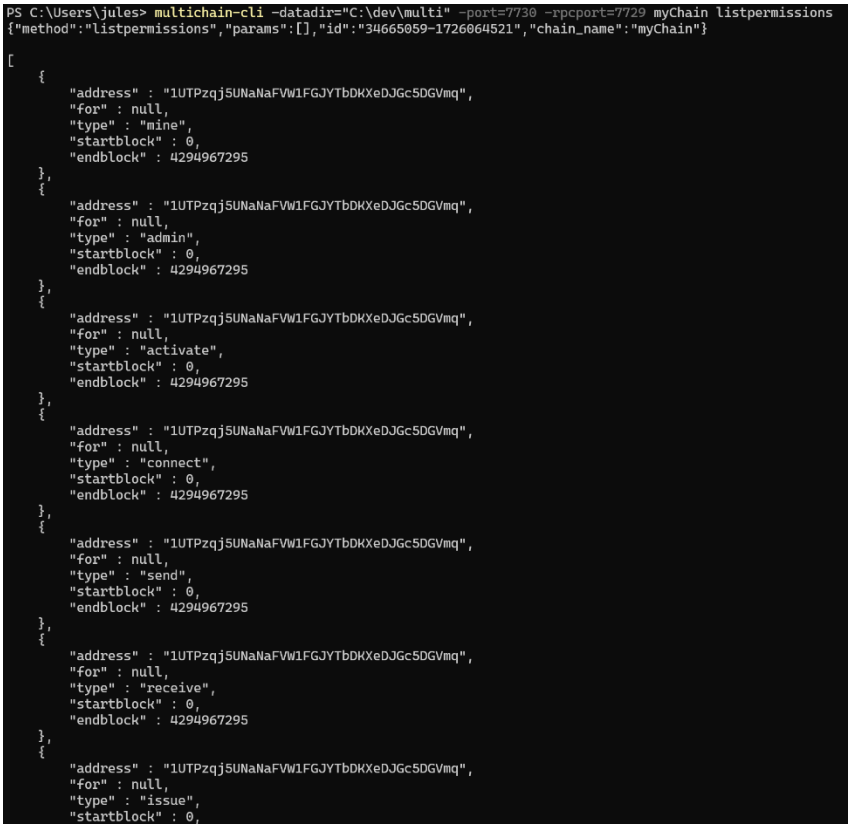
- multichain-cli -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain getinfo

Get the general information of the blockchain like node version, the port, etc. and the software used to run the chain like the version and the edition. This gives the same information regardless of the node

```
PS C:\Users\jules> multichain-cli -datadir="C:\dev\multi" -port=7730 -rpcport=7729 myChain getinfo
{"method":"getinfo","params":[],"id":"60052491-1726064520","chain_name":"myChain"}

{
    "version" : "2.3.3",
    "nodeversion" : 20303901,
    "edition" : "Community",
    "protocolversion" : 20013,
    "chainname" : "myChain",
    "description" : "MultiChain myChain",
    "protocol" : "multichain",
    "port" : 7730,
    "setupblocks" : 60,
    "nodeaddress" : "myChain@172.23.144.1:7730",
    "burnaddress" : "1XXXXXXWkXXXXXXXMJXXXXXXbqXXXXXXY23Ai8",
    "incomingpaused" : false,
    "miningpaused" : false,
    "offchainpaused" : false,
    "walletversion" : 60000,
    "balance" : 0,
    "walletdbversion" : 3,
    "reindex" : false,
    "blocks" : 24,
    "chainrewards" : 0,
    "streams" : 1,
    "timeoffset" : 0,
    "connections" : 2,
    "proxy" : "",
    "difficulty" : 5.96046447753906e-8,
    "testnet" : false,
    "keypoololdest" : 1726064404,
    "keypoolsize" : 2,
    "paytxfee" : 0,
    "relayfee" : 0,
    "errors" : ""
}
```

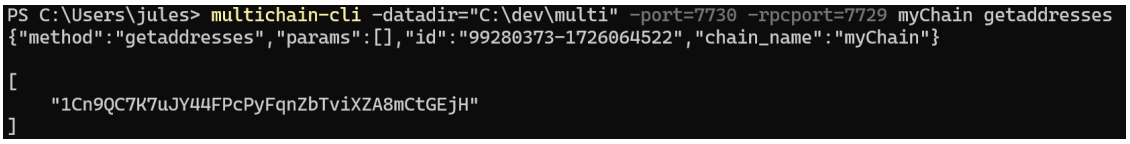
- multichain-cli -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain listpermissions

Get all the permissions granted with the address



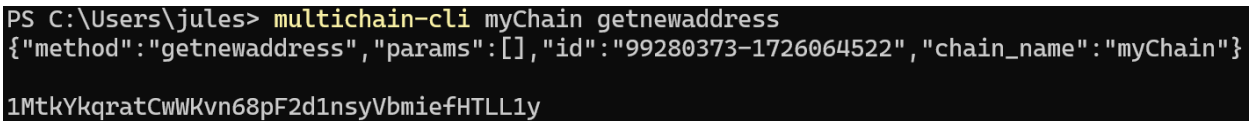
- multichain-cli -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain getaddresses

Get the addresses of the node, in this case the second node.



- multichain-cli myChain getnewaddress

Create a new address for the first node



- multichain-cli myChain getaddresses

Get the addresses of the node

- multichain-cli myChain getpeerinfo

Get the information of the peer that are connected to the current node like the address and the Ip

```
PS C:\Users\jules> multichain-cli myChain getpeerinfo
{"method":"getpeerinfo","params":[],"id":"48502761-1726064524","chain_name":"myChain"}

[
    {
        "id" : 5,
        "addr" : "172.23.36.20:7730",
        "addrlocal" : "172.23.36.20:52826",
        "services" : "0000000000000001",
        "lastsend" : 1726064521,
        "lastrecv" : 1726064521,
        "bytessent" : 7413,
        "bytesrecv" : 6596,
        "conntime" : 1726064416,
        "pingtime" : 0.078624,
        "version" : 70002,
        "subver" : "/MultiChain:0.2.0.13/",
        "handshakelocal" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "handshake" : "1Cn9QC7K7uJY44FPcPyFqnZbTviXZA8mCtGEjH",
        "inbound" : false,
        "encrypted" : false,
        "startingheight" : 0,
        "banscore" : 0,
        "synced_headers" : 16,
        "synced_blocks" : -1,
        "inflight" : [
        ],
        "whitelisted" : false
    },
    {
        "id" : 6,
        "addr" : "172.23.144.1:52828",
        "addrlocal" : "172.23.144.1:5779",
        "services" : "0000000000000001",
        "lastsend" : 1726064521,
        "lastrecv" : 1726064518,
        "bytessent" : 2441,
        "bytesrecv" : 2131,
        "conntime" : 1726064417,
        "pingtime" : 0.076615,
        "version" : 70002,
        "subver" : "/MultiChain:0.2.0.13/",
        "handshakelocal" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "handshake" : "1Cn9QC7K7uJY44FPcPyFqnZbTviXZA8mCtGEjH",
        "inbound" : true,
        "encrypted" : false,
        "startingheight" : 18,
        "banscore" : 0,
        "synced_headers" : 18,
        "synced_blocks" : 18,
        "inflight" : [
```

- multichain-cli -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain getpeerinfo

Get the information of the peer that are connected to the current node like the address and the Ip

```
PS C:\Users\jules> multichain-cli -datadir="C:\dev\multi" -port=7730 -rpcport=7729 myChain getpeerinfo
{"method":"getpeerinfo","params":[],"id":"48502761-1726064524","chain_name":"myChain"}

[
    {
        "id" : 3,
        "addr" : "172.23.36.20:52826",
        "addrlocal" : "172.23.36.20:7730",
        "services" : "0000000000000001",
        "lastsend" : 1726064521,
        "lastrecv" : 1726064521,
        "bytessent" : 6596,
        "bytesrecv" : 7413,
        "conntime" : 1726064416,
        "pingtime" : 0.030448,
        "version" : 70002,
        "subver" : "/MultiChain:0.2.0.13/",
        "handshakelocal" : "1Cn9QC7K7uJY44FPcPyFqnZbTviXZA8mCtGEjH",
        "handshake" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "inbound" : true,
        "encrypted" : false,
        "startingheight" : 18,
        "banscore" : 0,
        "synced_headers" : 25,
        "synced_blocks" : 25,
        "inflight" : [
        ],
        "whitelisted" : false
    },
    {
        "id" : 4,
        "addr" : "172.23.144.1:5779",
        "addrlocal" : "172.23.144.1:52828",
        "services" : "0000000000000001",
        "lastsend" : 1726064518,
        "lastrecv" : 1726064521,
        "bytessent" : 2131,
        "bytesrecv" : 2441,
        "conntime" : 1726064417,
        "pingtime" : 0.030761,
        "version" : 70002,
        "subver" : "/MultiChain:0.2.0.13/",
        "handshakelocal" : "1Cn9QC7K7uJY44FPcPyFqnZbTviXZA8mCtGEjH",
        "handshake" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "inbound" : false,
        "encrypted" : false,
        "startingheight" : 18,
        "banscore" : 0,
        "synced_headers" : 25,
        "synced_blocks" : 25,
        "inflight" : [
        ],
        "whitelisted" : false
```

Question 33:

The command ‘multichain-cli myChain listpermissions issue’ get a list of all node that can issue new asset on the blockchain. In this case, only the first address can issue new asset.

```
PS C:\Users\jules> multichain-cli myChain listpermissions issue
{"method":"listpermissions","params":["issue"],"id":"90117496-1726065281","chain_name":"myChain"}

[
    {
        "address" : "1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq",
        "for" : null,
        "type" : "issue",
        "startblock" : 0,
        "endblock" : 4294967295
    }
]
```

Question 35:

multichain-cli myChain issue 1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq myCryptoMoney 500 0.01

This command will create 500 units of a new asset called “MyCryptoMoney” on the blockchain and send them to the address specified. The smallest possible unit that can be transferred is 0.01 of these assets.

```
PS C:\Users\jules> multichain-cli myChain issue 1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq myCryptoMoney 500 0.01
{"method":"issue","params":["1UTPzqj5UNaNaFVW1FGJYTbDKXeDJGc5DGVmq","myCryptoMoney",500,0.01],"id":"48376415-1726065350","chain_name":"myChain"}

5e5d0e51d4757ffdd2889062487e3be1bcc45afc1d1586def968b18705557fd3
```

Question 38:

The first command gives the same result because the command gives the list of assets that is in the blockchain. But the second command give a different result because it gives the balances of each asset in the current node.

Question 39:

multichain-cli myChain sendasset 1QRPoZWj7uNhQwEvmV6ggcAPnjV5dSDtuMbizA myCryptoMoney 50

Question 40:

multichain-cli myChain gettotalbalances

Question 41:

WSL: multichain-cli myChain gettotalbalances

Windows second node: multichain-cli -datadir=~/.multichainother -port=7730 -rpcport=7729 myChain gettotalbalances

Question 42:

First step: give enough units to system 2 and 3:

On System 1:

First command:

multichain-cli myChain sendasset 1QRPoZWj7uNhQwEvmV6ggcAPnjV5dSDtuMbizA myCryptoMoney 130

Result:

```
PS C:\Users\jules> multichain-cli myChain sendasset 1QRPoZWj7uNhQwEvmV6ggcAPnjV5dSDtuMbizA myCryptoMoney 130
{"method":"sendasset","params":["1QRPoZWj7uNhQwEvmV6ggcAPnjV5dSDtuMbizA","myCryptoMoney",130],"id":"53103427-1726062964","chain_name":"myChain"}

1c8cdcdddbb9b01027126a56e6eebddf1c676dea5e9ea032a3e7c13cdbe0c35c
```

Second command:

multichain-cli myChain sendasset 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs myCryptoMoney 30

Result:

```
PS C:\Users\jules> multichain-cli myChain sendasset 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs myCryptoMoney 30
{"method":"sendasset","params":["1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs","myCryptoMoney",30],"id":"34574419-1726062986","chain_name":"myChain"}

09e14a0c3c3ba523c8fc8d6b99fdb37c8cfaa5f6f9c503fc00bda7fd66711e5d
```

Second step: Send 30 units from system 3 to system 1:

On System 2:

Command:

multichain-cli myChain sendasset 1UmZXcJNn8zKBK9QYpvdzrBWLbsywqjEu8Zaix myCryptoMoney 30

Result:

```
jules@TABLET-TQGU8LI5:~$ multichain-cli myChain sendasset 1UmZXcJNn8zKBK9QYpvdzrBWLbsywqjEu8Zaix myCryptoMoney 30
{"method":"sendasset","params":["1UmZXcJNn8zKBK9QYpvdzrBWLbsywqjEu8Zaix","myCryptoMoney",30],"id":"37559618-1726062993","chain_name":"myChain"}

9a2182e3266de39c3a9191987ba216831ce4510425e7d7b0afadb6c82e215478
```

Third step: Send 130 units from system 2 to system 3:

Command:

multichain-cli -datadir="C:\dev\multi" -port=7730 -rpcport=7729 myChain@172.23.144.1:6837 sendasset 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs myCryptoMoney 130

Result:

```
PS C:\Users\jules> multichain-cli -datadir="C:\dev\multi" -port=7730 -rpcport=7729 myChain@172.23.144.1:6837 sendasset 1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs myCryptoMoney 130
{"method":"sendasset","params":["1R7jfy6PVqmJXVe679L5Lyf4jdAimCG6Jy1Cxs","myCryptoMoney",130],"id":"82476271-1726063959","chain_name":"myChain"}

a4156284c80a993f5b9295bd59de14dca8f65ac525db183ae7dd9f3d2b490640
```

Conclusion:

To conclude, a blockchain was created using multichain-util and connected to using the address myChain@172.23.144.1:9259. The blockchain was configured to allow only the first address to issue new assets. Assets were transferred between nodes, demonstrating the functionality of the blockchain.

DA51 Lab Session 2: The Geth client

Description:

This session teaches us how to set up a private Ethereum network with Geth, focusing on the Clique Proof of Authority (PoA) consensus algorithm. It covers creating accounts, a genesis block, and signer account keys for the network. The session also explains the structure of the extradata field in the genesis block for Clique Proof of Authority.

A private Ethereum blockchain is created, initialized, and two nodes are established. A bootnode is configured to facilitate peer-to-peer connections, and the nodes are launched with unique IDs and ports. Ether is transferred between accounts on the nodes, demonstrating transaction functionality.

Transfer ether between addresses using Wei units.

Question 5:

```
PS C:\dev\lab-session-2> geth --datadir node1 account new
INFO [09-18|14:07:15.461] Maximum peer count                       ETH=50 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x547a58C011f6121F913C378F7338B95A43526353
Path of the secret key file: node1\keystore\UTC--2024-09-18T12-07-19.616153200Z--547a58c011f6121f913c378f7338b95a4352635
3

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

The command creates a keyfile that is stored in the keystore path

Question 10:

The command initializes the database. To connect the two nodes later, this initialization must be done in both directories.

```
PS C:\dev\lab-session-2> geth init --datadir node1 genesis.json
INFO [09-18|16:15:57.383] Maximum peer count                       ETH=50 total=50
WARN [09-18|16:15:57.391] Lowering memory allowance on 32bit arch  available=8026 addressable=2048
WARN [09-18|16:15:57.391] Sanitizing cache to Go's GC limits       provided=1024 updated=682
INFO [09-18|16:15:57.393] Set global gas cap                       cap=50,000,000
INFO [09-18|16:15:57.393] Initializing the KZG library             backend=gokzg
INFO [09-18|16:15:57.750] Defaulting to pebble as the backing database
INFO [09-18|16:15:57.750] Allocated cache and file handles         database=C:\dev\lab-session-2\node1\geth\chaindata cache=16.00MiB handles=16
INFO [09-18|16:15:57.800] Opened ancient database                  database=C:\dev\lab-session-2\node1\geth\chaindata\ancient/chain readonly=false
INFO [09-18|16:15:57.801] State schema set to default              scheme=hash
INFO [09-18|16:15:57.802] Writing custom genesis block
INFO [09-18|16:15:57.806] Persisted trie from memory database      nodes=3 size=405.00B time=2.1806ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [09-18|16:15:57.826] Successfully wrote genesis state         database=chaindata hash=166a0e..3ccd7f
INFO [09-18|16:15:57.826] Defaulting to pebble as the backing database
INFO [09-18|16:15:57.826] Allocated cache and file handles         database=C:\dev\lab-session-2\node1\geth\lightchaindata cache=16.00MiB handles=16
INFO [09-18|16:15:57.889] Opened ancient database                  database=C:\dev\lab-session-2\node1\geth\lightchaindata\ancient/chain readonly=false
INFO [09-18|16:15:57.890] State schema set to default              scheme=hash
INFO [09-18|16:15:57.890] Writing custom genesis block
INFO [09-18|16:15:57.893] Persisted trie from memory database      nodes=3 size=405.00B time=2.6261ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [09-18|16:15:57.909] Successfully wrote genesis state         database=lightchaindata hash=166a0e..3ccd7f
```

Question 12:

```
PS C:\dev\lab-session-2> bootnode -nodekey boot.key -addr :30305
enode://ec572066a2059f2f330b7f2a9bc53e3cdc7b12614c90fd94c155103be7a2
da15247f96f62b42bae60029b862dd3c1c35159d878134d433b73b1483ede2b904c5
@127.0.0.1:0?discport=30305
Note: you're using cmd/bootnode, a developer tool.
We recommend using a regular node as bootstrap node for production d
eployments.
INFO [09-18|14:21:26.532] New local node record                    s
eq=1,726,662,086,529 id=10c5af03f5e031bf ip=<nil> udp=0 tcp=0
```

Question 13:

First command: geth --datadir node1 --port 30307 --bootnodes enode://7f851530fc477c9f183d02719e4066e4252b6bd2572d29d4f3d57e785701b6a19a87c12c499327d5cbcd5dac5afbd437ab57cc69c02c5619f65f327d9da40580@127.0.0.1:0?discport=30305 --networkid 1234567890 --unlock B6E9D58c3A76f9E5640e6920cA03dF9d30FcD331 --password node1/password.txt --authrpc.port 8551 --ipcpath node1 --miner.etherbase 0xB6E9D58c3A76f9E5640e6920cA03dF9d30FcD331

Second command: geth --datadir node2 --port 30308 --bootnodes enode://7f851530fc477c9f183d02719e4066e4252b6bd2572d29d4f3d57e785701b6a19a87c12c499327d5cbcd5dac5afbd437ab57cc69c02c5619f65f327d9da40580@127.0.0.1:0?discport=30305 --networkid 1234567890 --unlock c2024d10C9F18176A0Eb290F9e35DAd6F10BeeF4 --password node2/password.txt --authrpc.port 8552 --ipcpath node2

I needed to add –ipcpath because of a forbidden access of the start of the two commands at the same time. In the same way, I needed to add –miner.etherbase to sepecify the etherbase in order to mine, requirement for something later in the TP.

Question 15:

As I specified in the previous question the -ipcpath the command look like : geth attach \\.\pipe\node1 for the node 1 and geth attach \\.\pipe\node2 for the node 2.

```
PS C:\dev\lab-session-2> geth attach \\.\pipe\node1
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.15-stable-c5ba367e/windows-386/go1.21.6
coinbase: 0xb6e9d58c3a76f9e5640e6920ca03df9d30fcd331
at block: 2 (Wed Sep 18 2024 16:26:05 GMT+0200 (CEST))
 datadir: C:\dev\lab-session-2\node1
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

```
PS C:\Users\jules> geth attach \\.\pipe\node2
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.15-stable-c5ba367e/windows-386/go1.21.6
at block: 0 (Thu Jan 01 1970 01:00:00 GMT+0100 (CET))
 datadir: C:\dev\lab-session-2\node2
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

Question 16:

```
> net.peerCount
1
```

Question 17:

```
> admin.peers
[{
    caps: ["eth/68", "snap/1"],
    enode: "enode://baea567b4a411d5052fa5b83de8bc0ef1ba0020a4670bb0c21f884feab7b2449a3bb460ef552146f52d9dbdbf1b2b982b2dfa68a6655c4b31ff8c5c9bcb6b982@127.0.0.1:58004",
    id: "4b566c159451beea618364c4f8e7ac3d1408d4c505529101d7241f82ea69173f",
    name: "Geth/v1.13.15-stable-c5ba367e/windows-386/go1.21.6",
    network: {
      inbound: true,
      localAddress: "127.0.0.1:30307",
      remoteAddress: "127.0.0.1:58004",
      static: false,
      trusted: false
    },
    protocols: {
      eth: {
        version: 68
      },
      snap: {
        version: 1
      }
    }
}]
```

Question 18:

```
> eth.getBalance(eth.accounts[0])
9.999999934101971356999e+22
```

Question 19:

eth.sendTransaction({to: 'c2024d10C9F18176A0Eb290F9e35DAd6F10BeeF4', from: eth.accounts[0], value: 25000});

```
> eth.sendTransaction({to: 'c2024d10C9F18176A0Eb290F9e35DAd6F10BeeF4', from: eth.accounts[0], value: 25000});
"0x7f459fe9b10a7ebb012be79e2b629bbdb964f550d81aaac81207780817d8f524"
```

Question 20:

In order to get the value, we need to start the miner by miner.start()

```
> miner.start()
null
```

```
> eth.getBalance('c2024d10C9F18176A0Eb290F9e35DAd6F10BeeF4');
125000
```

Question 21:

```
PS C:\Users\jules> geth attach \\.\pipe\node2
Welcome to the Geth JavaScript console!

instance: Geth/v1.13.15-stable-c5ba367e/windows-386/go1.21.6
at block: 0 (Thu Jan 01 1970 01:00:00 GMT+0100 (CET))
 datadir: C:\dev\lab-session-2\node2
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

Question 22:

```
> eth.getBalance(eth.accounts[0])
125000
```

Question 23:

Yes, it matches

Question 24:

Balance in Ethereum is represented in Wei by default.

Question 25:

1 ether is equal to 10^{18} Wei

Question 26:

List of command in this order:

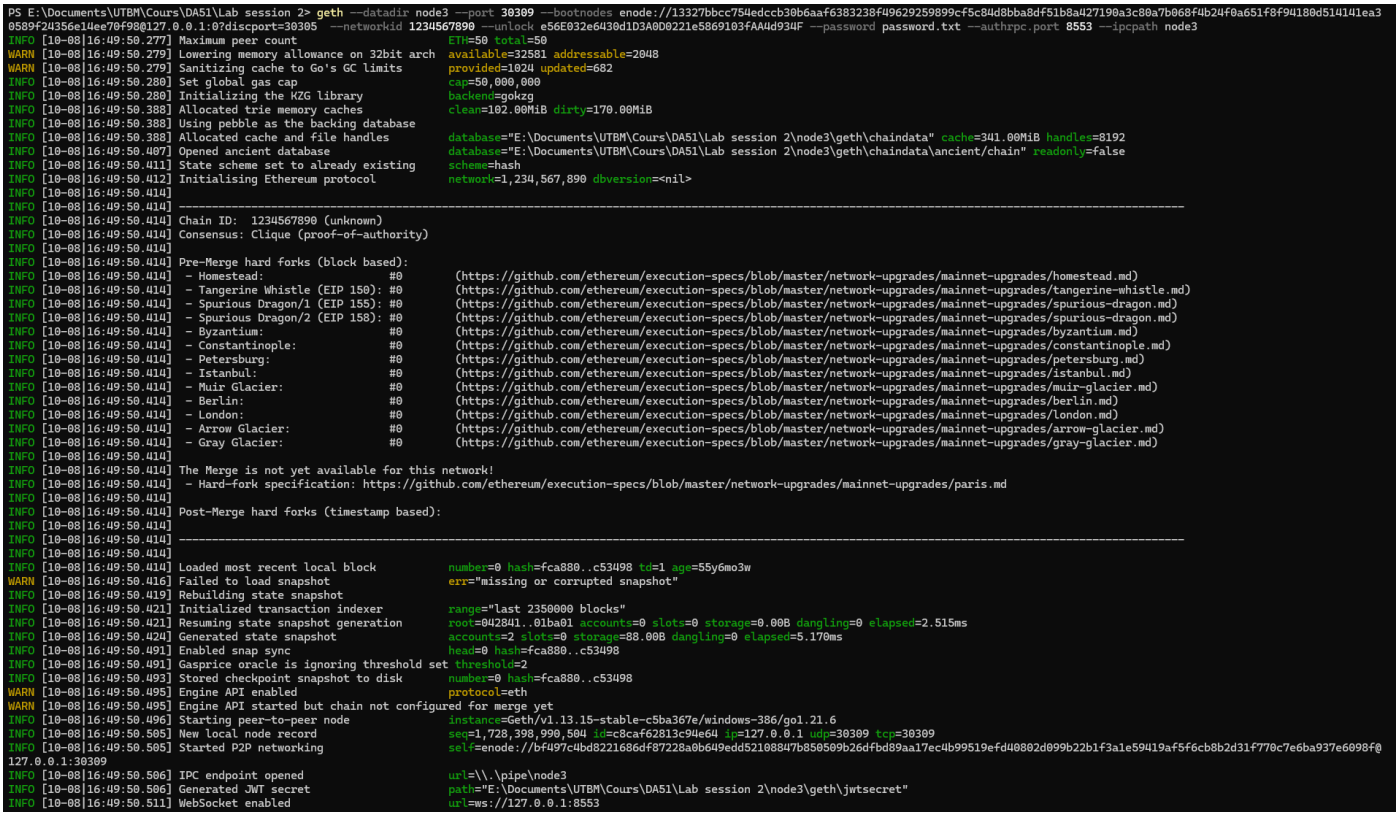
- mkdir node3
- geth –datadir node3 account new

```
PS E:\Documents\UTBM\Cours\DA51\Lab session 2> geth --datadir node3 account new
INFO [10-08|16:48:08.822] Maximum peer count                       ETH=50 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0xe56E032e6430d1D3A0D0221e5869103fAA4d934F
Path of the secret key file: node3\keystore\UTC--2024-10-08T14-48-12.925717600Z--e56e032e6430d1d3a0d0221e5869103faa4d934f

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

- geth init --datadir node3 genesis.json

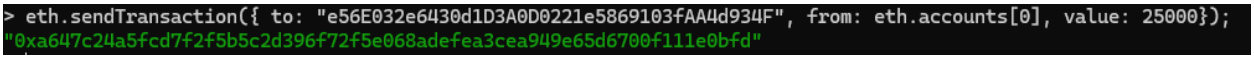
```
PS E:\Documents\UTBM\Cours\DA51\Lab session 2> geth init --datadir node3 genesis.json
INFO [10-08|16:45:15.327] Maximum peer count                       ETH=50 total=50
WARN [10-08|16:45:15.331] Lowering memory allowance on 32bit arch  available=32581 addressable=2048
WARN [10-08|16:45:15.331] Sanitizing cache to Go's GC limits       provided=1024 updated=682
INFO [10-08|16:45:15.331] Set global gas cap                       cap=50,000,000
INFO [10-08|16:45:15.331] Initializing the KZG library             backend=gokzg
INFO [10-08|16:45:15.432] Defaulting to pebble as the backing database
INFO [10-08|16:45:15.432] Allocated cache and file handles         database="E:\Documents\UTBM\Cours\DA51\Lab session 2\node3\geth\chaindata" cache=16.00MiB handles=16
INFO [10-08|16:45:15.457] Opened ancient database                  database="E:\Documents\UTBM\Cours\DA51\Lab session 2\node3\geth\chaindata\ancient/chain" readonly=false
INFO [10-08|16:45:15.457] State schema set to default              scheme=hash
INFO [10-08|16:45:15.457] Writing custom genesis block
INFO [10-08|16:45:15.459] Persisted trie from memory database      nodes=3 size=405.00B time=1.5574ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [10-08|16:45:15.470] Successfully wrote genesis state         database=chaindata hash=fca880..c53498
INFO [10-08|16:45:15.470] Defaulting to pebble as the backing database
INFO [10-08|16:45:15.470] Allocated cache and file handles         database="E:\Documents\UTBM\Cours\DA51\Lab session 2\node3\geth\lightchaindata" cache=16.00MiB handles=16
INFO [10-08|16:45:15.498] Opened ancient database                  database="E:\Documents\UTBM\Cours\DA51\Lab session 2\node3\geth\lightchaindata\ancient/chain" readonly=false
INFO [10-08|16:45:15.498] State schema set to default              scheme=hash
INFO [10-08|16:45:15.498] Writing custom genesis block
INFO [10-08|16:45:15.500] Persisted trie from memory database      nodes=3 size=405.00B time=1.5357ms gcnodes=0 gcsize=0.00B gctime=0s livenodes=0 livesize=0.00B
INFO [10-08|16:45:15.511] Successfully wrote genesis state         database=lightchaindata hash=fca880..c53498
```

- geth --datadir node3 --port 30309 --bootnodes enode://13327bbcc754edccb30b6aaf6383238f49629259899cf5c84d8bba8df51b8a427190a3c80a7b068f4b24f0a651f8f94180d514141ea30589f24356e14ee70f98@127.0.0.1:0?discport=30305 --networkid 1234567890 --unlock e56E032e6430d1D3A0D0221e5869103fAA4d934F --password password.txt --authrpc.port 8553 --ipcpath node3



To send money to node3 from node1:

eth.sendTransaction({ to: "e56E032e6430d1D3A0D0221e5869103fAA4d934F", from: eth.accounts[0], value: 25000});



Conclusion:

Geth commands are used to initialize Ethereum nodes, specifying datadir, port, bootnodes, networkid, unlock accounts, and other parameters. Transactions are sent between nodes using eth.sendTransaction, and miners are started with miner.start(). Balance is displayed in Wei, with 1 ether equal to 10^18 Wei.

DA51 Lab Session 3 : Smart Contract

Description :

Lab session on Solidity and Truffle Suite for smart contract development. We learn Solidity, compile contracts, write test cases, and automate deployment processes.

Question 4:

The purpose of NPM is to manage all dependencies of a node project.

Question 7:

One the command is executed I get a package.json that hold all information I entered during the command process

```
{
  "name": "lab3",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

```
PS C:\dev\DA53\TP3\Lab3> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (lab3)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\dev\DA53\TP3\Lab3\package.json:

{
  "name": "lab3",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}


Is this OK? (yes) yes
```

Question 8:

It contains all information of my project

```
{
  "name": "lab3",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": ""
}
```

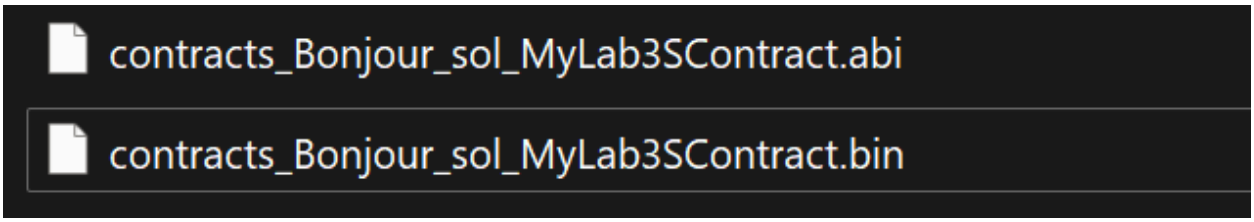
Question 14:

Npm is used to install dependencies. Option -g means that this package is installed for all projects.

After this question, all answer will be from WSL.

Question 19:

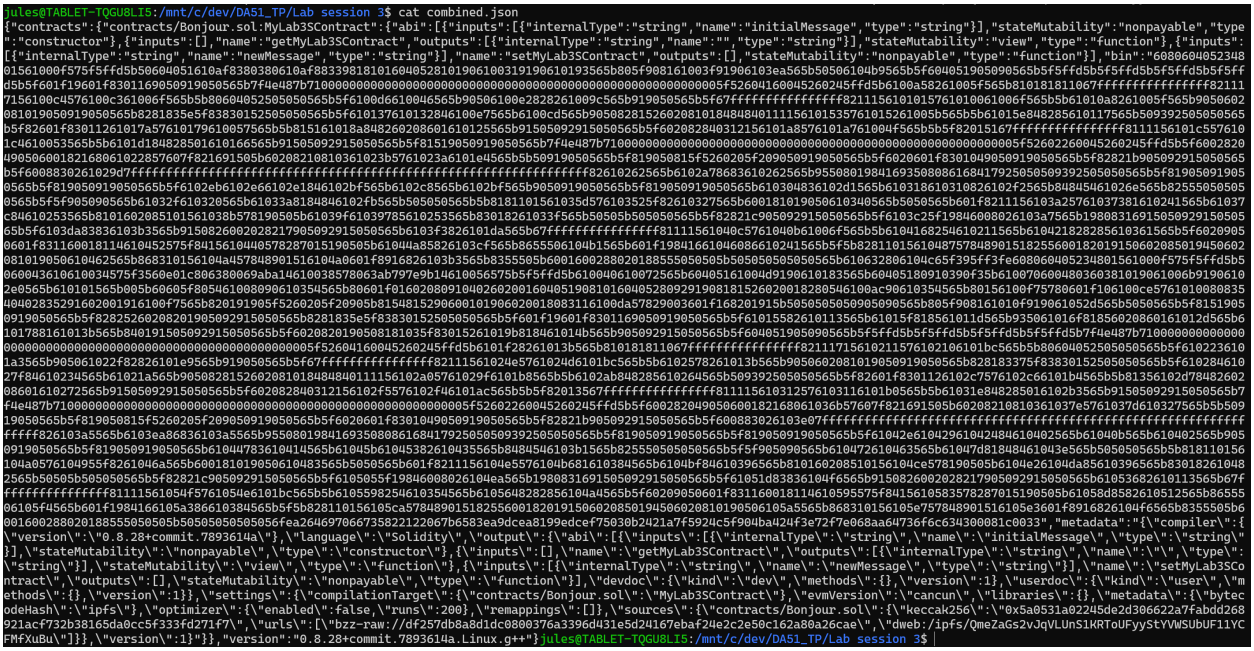
The command make two different files, one for with the .abi extension and another for the .bin extension.



Question 20:

```
jules@TABLET-TQGU8LI5:/mnt/c/dev/DA51_TP/Lab session 3$ ls
combined.json  contracts_Bonjour_sol_MyLab3SContract.abi  package.json
contracts      contracts_Bonjour_sol_MyLab3SContract.bin
```

Question 22:



```
jules@TABLET-TQGU8LI5:/mnt/c/dev/DA51_TP/Lab session 3$ cat combined.json
{"contracts":{"contracts/Bonjour.sol:MyLab3SContract":{"abi":[{"inputs":[{"internalType":"string","name":"initialMessage","type":"string"}],"stateMutability":"nonpayable","type
":"constructor"},{"inputs":[],"name":"getMyLab3SContract","outputs":[{"internalType":"string","name":"","type":"string"}],"stateMutability":"view","type":"function"},{"inputs":
[{"internalType":"string","name":"newMessage","type":"string"}],"name":"setMyLab3SContract","outputs":[],"stateMutability":"nonpayable","type":"function"}],"bin":"6080604052348
01561000f575f5ffd5b50604051610af8380380610af883398181016040528101906100319190610193565b805f908161003f91906103ea565b50506104b9565b5f604051905090565b5f5ffd5b5f5ffd5b5f5ffd5b5f5ff
d5b5f601f19601f830116905091905056 5b7f4e487b7100000000000000000000000000000000000000000000000000000000000000005f52604160045260245ffd5b6100a58261005f565b810181811067fffffffffffffffff82111
7156100c4576100c361006f565b5b8060405250505056 5b5f6100d661004656 5b90506100e2828261009c565b919050565b5f67ffffffffffffffff82111561015761010061006f565b5b61010a8261005f565b9050602
0810190509190505656 5b8281835e5f8383015250505056 5b5f6101376101328461006e7565b6100cd565b9050828152602081018484840111156101535761015261005b565b5b61015e84828561011756 5b5093925050505656
5b5f82601f830112610175a5761017961005756 5b5b815161018a84826020860161012556 5b9150509291505056 5b5f6020828403121156101a8576101a761004f565b5b5f8201516 7fffffffffffffffff8111156101c557610
1c461005356 5b5b6101d184828501610166565b9150509291505056 5b5f8151905091905056 5b7f4e487b7100000000000000000000000000000000000000000000000000000000000000005f52602260045260245ffd5b5f6002820
4905060018216806102285760 7f821691505b602082108103610 23b5761023a6101e4565b5b509190505656 5b5f8190508 15f5260205f209050919050565b5f6020601f8301049050919050565b5f82821b905092915050565
b5f6008830261029d7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff82610262565b6102a786836102626565b955080198416935080861684179250505093925050505656 5b5f81905091905
0565b5f8190509190505656 5b5f6102eb6102e66102e1846102bf565b6102c8565b6102bf565b9050919050565b5f8190509190505656 5b610304836102d1565b610318610310826102f2565b84845461026e565b82555050505
0565b5f5f905090565b61032f610320565b6133a8184846102fb565b505050565b5b818110156103 5d576103525f82610327565b600181019050610340565b5050565b601f8211156103a2576103738161024156 5b61037
c8461025356 5b8101602085101561038b57819050 5b61039f6103978561025356 5b83018261033f565b50505b505050565b5f82821c9050929150505656 5b5f6103c25f19846008026103a7565b19808316915050929150505
65b5f6103da83836103b3565b9150826002028217905092915050565b6103f3826101da565b67ffffffffffffffff81111561040c576104b61006f565b5b6104168254610211565b6104218282856103615656 5b5f6020905
0601f831160018114610452575f84156104 0578287015190505b6104 a85826103cf565b8655506104 b1565b601f198416610406861024156 5b5f5b828110156104875784890151825560018201915060208501945060 2
08101905061046256 5b868310156104a4578490151610 4a0601f8916826103b3565b8355505b6001602880201885550505050565b6106328806104cc65f395ff3fe608060405234801561000f575f5ffd5b5
060043610610034575f3560e01c806380069aba1461003857806 3ab797e9b14610056575b5f5ffd5b610040610072565b6040516104d919061018356 5b6040518091039 0390f35b6100706004803603810190610060 9190610
2e0565b610101565b005b60605f80546100809061035456 5b80601f0160208091040260200160405190810160405280929190818152602001828054610 0ac90610354565b8015610 0f75780601f106100ce57610100808035
40402835291602001916100f7565b820191905f5260205f20905b815481529060010190602001808311610 0da57829003601f168201915b505050505090509056 5b805f908161010f919061052d565b5050565b5f8151905
0919050565b5f828252602082019050929150505656 5b8281835e5f8383015250505056 5b5f601f19601f830116905091905056 5b5f6101558261011356 5b61015f818561011d565b935061016f818560208601610 12d565b6
101788161013b565b840191505092915050565 5b5f6020820190508181035f830152610 19b818461014b565b9050929150505656 5b5f604051905090565b5f5ffd5b5f5ffd5b5f5ffd5b5f5ffd5b5f5ffd5b7f4e487b710000000000000
0000000000000000000000000000000000000000000005f52604160045260245ffd5b6101f28261013b565b810181811067fffffffffffffffff82111715610211576102106101bc565b5b806040525050505656 5b5f610223610
1a3565b905061022f8282610 1e9565b919050565b5f67ffffffffffffffff82111561024e5761024d6101bc565b5b6102578261013b565b905060208101905091905056 5b828183375f8383015250505056 5b5f61028461 0
27f84610234565b61021a565b905082815260208101848484011115610 2a05761029f6101b8565b5b6102ab84828561026 4565b5093925050565b5f82601f8301126102c7576102c66101b4565b5b81356102d784826202
0860161027256 5b9150509291505056 5b5f6020828403121561026 2f5576102f4 6101ac565b5b5f82013567fffffffffffffffff81111561031257610311610 1b0565b5b61031e848285016102b3565b9150509291505056 5b7
f4e487b7100000000000000000000000000000000000000000000000000000000000000005f52602260045260245ffd5b5f60028204905060018216806103 6b57607f821691505b602082108103610 37e5761037d610327565b5b509
19050565b5f819050815f5260205f20905091905056 5b5f6020601f8301049050919050565b5f82821b9050929150505656 5b5f600883026103e07fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffff826103a5565b6103ea86836103a5565b9550801984169350808616841792505050939250505056 5b5f8190509190505656 5b5f8190509190505656 5b5f6104 2e6104296104284846104025656 b61040b565b610402565b905
0919050565b5f819050919050565b6104478361041456 5b6104 5b6104538261043565b848454 6103b1565b8255505050565b5b5f5f905090565b6104726104 63565b61047d8184846104 3e565b505050565b5b818110156
104a05761049055f82610 46a565b6001810190506104 83565b5050565b601f8211156104e5576104 be8461039656 5b8101602085101561046ce578190505b6104e2610 4da85610396565b83018261048
2565b5050565b50505056 5b5f82821c905092915050565b5f6105055f198460080261046ea565b19808316915050929150505656 5b5f61051d83836104f6565b9150826002028217905092915050565b6105368261011356 5b67f
fffffffffffffff81111561054f5761054e6101bc565b5b61055982546103545656 5b61056482828561046a4565b5f602090506 01f831160181146105955575f84156105835782870151905056 5b61058d858261051256 5b56555
06105f4565b601f198416105a386610384565b5f5b828110156105ca5784890151825560018201915060208501945060208101905061046a5565b86831015610 5e7578490151610 5e3601f8916826104f6565b8355505b6
0016002880201885550505056 5b5050505050505656fea264697066735822122067b6583ea9dcea8199edcef75030b2421a7f5924c5f904ba424f3e72f7e068aa64736f6c634300081c0033","metadata":"{\"compiler\":{
\"version\":\"0.8.28+commit.7893614a\"},\"language\":\"Solidity\",\"output\":{\"abi\":[{\"inputs\":[{\"internalType\":\"string\",\"name\":\"initialMessage\",\"type\":\"string\"
}],\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\"inputs\":[],\"name\":\"getMyLab3SContract\",\"outputs\":[{\"internalType\":\"string\",\"name\":\"\",\"type\":
\"string\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"string\",\"name\":\"newMessage\",\"type\":\"string\"}],\"name\":\"setMyLab3SCo
ntract\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"}],\"devdoc\":{\"kind\":\"dev\",\"methods\":{},\"version\":1},\"userdoc\":{\"kind\":\"user\",\"m
ethods\":{},\"version\":1}},\"settings\":{\"compilationTarget\":{\"contracts/Bonjour.sol\":\"MyLab3SContract\"},\"evmVersion\":\"cancun\",\"libraries\":{},\"metadata\":{\"bytec
odeHash\":\"ipfs\"},\"optimizer\":{\"enabled\":false,\"runs\":200},\"remappings\":[]},\"sources\":{\"contracts/Bonjour.sol\":{\"keccak256\":\"0x5a0531a02245de2d30622a7fabdd268
921acf732b38165da0cc5f333fd271f7\",\"urls\":[\"bzz-raw://df257db8a8d1dc0800376a3396d431e5d24167ebaf24e2c2e50c162a80a26cae\",\"dweb:/ipfs/QmeZaGs2vJqVLUnS1kRToUFyySYVWSUbUF11YC
FMfXuBu\"]}},\"version\":1}"}},"version":"0.8.28+commit.7893614a.Linux.g++"}jules@TABLET-TQGU8LI5:/mnt/c/dev/DA51_TP/Lab session 3$
```

Question 23:

It includes the three different output formats, which we had specified. It includes the following fields: abi, bin, and metadata

Question 24:

There is the abi output which is a descriptor of the functions and the data within a contract which will allow us to interact with it.

Question 25:

Then there is the binary byte code for the contract.

Question 26:

And finally, there is a metadata field which includes various details about the contract itself.

Question 29, 30, 31 et 32:


```
{
   "contracts":{
      "contracts/Bonjour.sol:MyLab3SContract":{
         "abi":[ ],
         "bin":"608060405234801561000f575f5ffd5b50604051610af8380380610af883398181016040528101906100319190
         "metadata":"{\"compiler\":{\"version\":\"0.8.28+commit.7893614a\"},\"language\":\"Solidity\",\"ou
      }
   },
   "version":"0.8.28+commit.7893614a.Linux.g++"
}
```

Queston 34:

```
{
  "name": "lab-session-3",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "ganache-cli": "^6.12.2",
    "mocha": "^10.8.2",
    "solc": "^0.8.28",
    "web3": "^4.14.0"
  }
}
```

Question 35:

In the package-lock.json, there are all the dependencies required for the dependencies added with the command.

Question 40:


```
{
  "contracts":{
    "MyLab3SContract":{
      "MyLab3SContract":{
        "abi":[ ],
        "evm":{
          "bytecode":{
            "functionDebugData":{ },
            "generatedSources":[ ],
            "linkReferences":{ },
            "object":"608060405234801561000f575f5ffd5b50604051610af8380380610af883398181016040528101906100319190610193565b805f908161003f91906103ea565b50506104b9565b5f604051905090565
            "opcodes":"PUSH1 0x80 PUSH1 0x40 MSTORE CALLVALUE DUP1 ISZERO PUSH2 0xF JUMPI PUSH0 PUSH0 REVERT JUMPDEST POP PUSH1 0x40 MLOAD PUSH2 0xAF8 CODESIZE SUB DUP1 PUSH2 0xAF8
            "sourceMap":"35:361:0:-:0;;;92:86;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;:::i;:::-;156:14;146:7;:24;;;;;;;:::i;:::-;;92:86;35:361;;7:75:1;40:6;73:2;67:9;57:19;;7:75;:::o;88:117::-
          }
        }
      }
    }
  },
  "errors":[
    { }
  ],
  "sources":{
    "MyLab3SContract":{
      "id":0
    }
  }
}
```


Question 44:

There is no output in the terminal because we comment the line that print in the terminal.

Question 50:

```
PS C:\dev\DA51_TP\Lab session 3> npm test

> lab-session-3@1.0.0 test
> mocha



  MyLab3SContract
[
  '0x358bC94DeF0f9C34355ce926930F8028fbB2CC47',
  '0x4FAD93Ded19D225920610A1407083F7844930B01',
  '0x6Cea4457c61E943907908feDb57852D54b4CC5bd',
  '0x6BBF361CdeB695913Cf8133553FB204996e321B7',
  '0x7f4a0054221b7334B2C07AbDCfd37ABBe594564C',
  '0xf88Ee7967F40347Bf1Dd11e88a081ef51c204812',
  '0xeA522a0FC2e5CcA9e7914cb683113bd89b97C41e',
  '0x2cF965731a09F7D4dc4A61B2947A97cbeFcd23e6',
  '0x530E971718Df725D753d8fa7680264fD3737Bd2f',
  '0xbd11F71Ad494Ba9B9d3346f368430DD1Ca81B6D6'
]
    ✔Deploy a contract


  1 passing (232ms)
```

DA51 Lab Session 5: MetaMask

Description:

This session teaches how to use MetaMask, an Ethereum client, to create accounts, load Ether, and recover account information. It also explains how to use the Sepolia test network to generate Ether and how to use Alchemy and ChainLink.

Question 10:

Alchemy is a decentralized platform that provides developers with tools and infrastructure to build blockchain applications. It offers APIs, developer tools, and services to simplify the creation and management of blockchain-based projects.

Question 11:

Chainlink is a decentralized oracle network that enables smart contracts on blockchains to securely interact with real-world data and external APIs. It provides reliable data feeds, cross-chain interoperability, and other services to enhance the functionality of decentralized applications (DApps) and smart contracts.

Conclusion:

To conclude, Alchemy simplifies blockchain application development. Chainlink enables smart contracts to interact with real-world data.

DA51 Lab Session 6: Ethereum Decentralized Application

Description:

This lab session introduces Ethereum decentralized applications (Dapps) and the Truffle Suite for developing them. It covers setting up the development environment, including Node.js, npm, and Truffle, and explains the purpose of various tools like Ganache, web3.js, MetaMask, Lite-Server, and Gulp. The session then guides students through creating a Truffle project using the Pet-Shop box and understanding its directory structure.

Truffle framework is used to develop a smart contract for a pet adoption application, including writing the contract, compiling and migrating it to a local blockchain, testing it, and creating a user interface. Boilerplate HTML is used to speed up the development of the client-side UI, and MetaMask Wallet is configured to interact with the local blockchain. Finally, the lite server is set up to serve the front-end files.

All questions in this Lab Session are answered in the subject.

DA51 Lab Session 7: Dapp Scientific Prepublication

Description:

Lab 7 aims to build a scientific prepublication platform using blockchain for document integrity and non-repudiation. The platform will utilize smart contracts for document upload and verification, with testing on a local blockchain environment.

Question 1:

Non-repudiation and integrity are essential in scientific publications to ensure the reliability, trustworthiness, and accountability of the research. Here's how each of these concepts contributes to scientific integrity:

- Non-repudiation in scientific publishing ensures that authors cannot deny their involvement or claims made in their work.
- Integrity in scientific publishing refers to the trustworthiness and authenticity of the data, methods, and conclusions presented in research papers.

Question 2:

Blockchain technology can significantly enhance both non-repudiation and integrity in scientific publications by providing a secure, decentralized, and immutable record of data, authorship, and publication history.

Question 4:

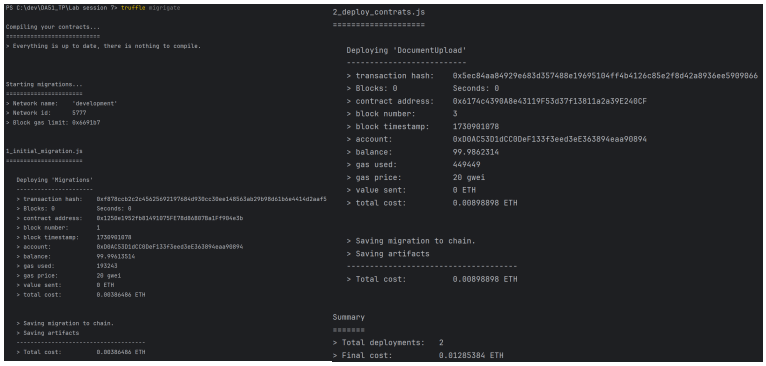
Smart Contract:

```
pragma solidity >=0.4.22 <0.9.0;

contract DocumentUpload {

    struct Document {
        string documentHash;
        uint256 timestamp;
    }

    mapping(address => mapping(string => Document)) public documents;

    function uploadDocument(string memory _documentHash) public {
        require(bytes(_documentHash).length > 0, "Document hash is required");

        Document storage document = documents[msg.sender][_documentHash];
        document.documentHash = _documentHash;
        document.timestamp = block.timestamp;

    }

    function verifyDocument(address _uploader, string memory _documentHash) public view returns (uint256) {
        Document storage document = documents[_uploader][_documentHash];
        if (bytes(document.documentHash).length == 0) {
            return 0;
        } else {
            return document.timestamp;
        }
    }
}
```

Compile:

```
PS C:\dev\DA51_TP\Lab session 7> truffle compile

Compiling your contracts...
===========================
> Compiling .\contracts\DocumentUpload.sol
> Compiling .\contracts\Migrations.sol
> Artifacts written to C:\dev\DA51_TP\Lab session 7\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

Migrate:



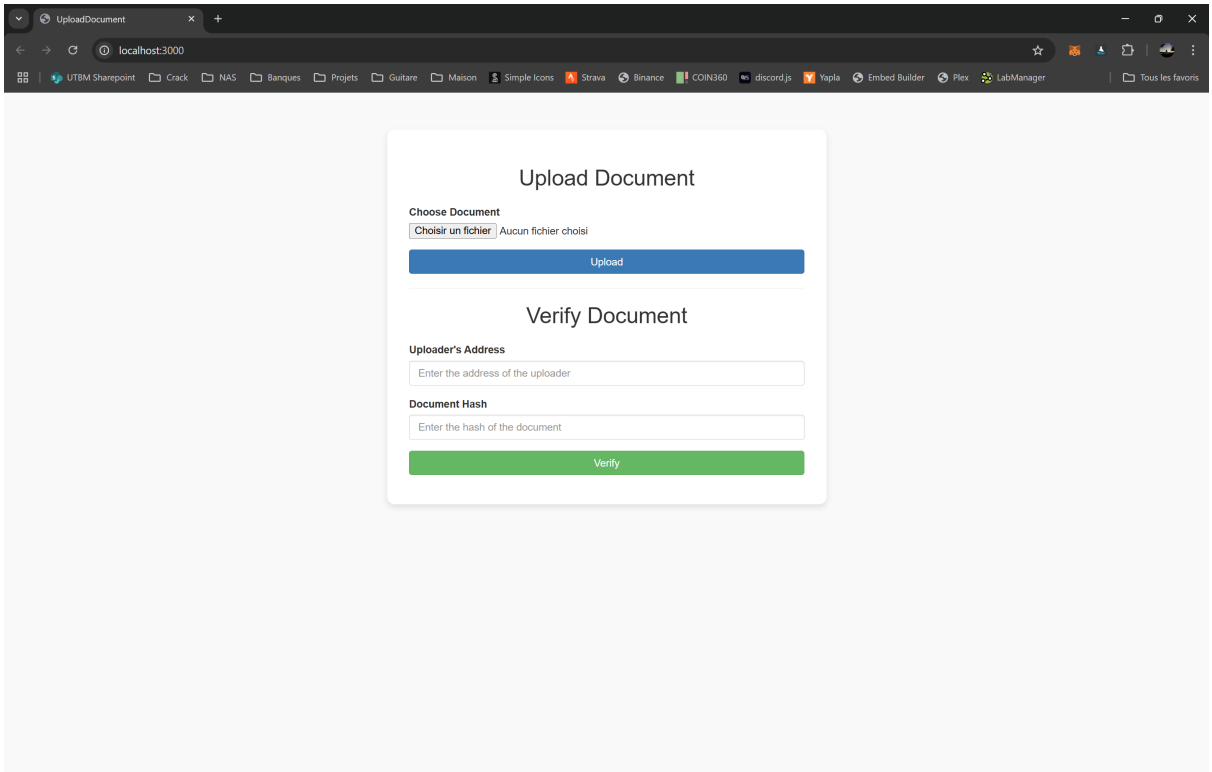
Question 5:

Tests:

```
PS C:\dev\DA51_TP\Lab session 7> truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Compiling .\contracts\DocumentUpload.sol
> Compiling .\test\TestDocumentUpload.sol
> Compilation warnings encountered:

    /C/dev/DA51_TP/Lab session 7/test/TestDocumentUpload.sol:16:5: Warning: Function state mutability can be restricted to view
    function testVerifyDocument() public {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to C:\Users\jules\AppData\Local\Temp\test-2024106-344-1mywfvx.zpuk
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



  TestDocumentUpload
    √ testUploadDocument (49ms)
    √ testVerifyDocument (40ms)


  2 passing (7s)
```

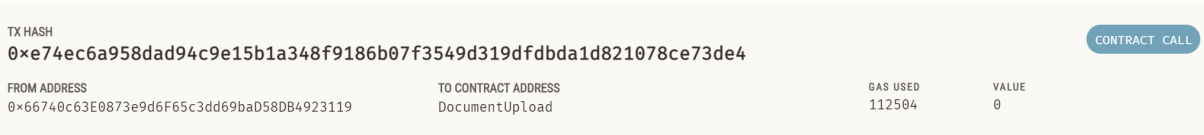
Question 6:

The platform:

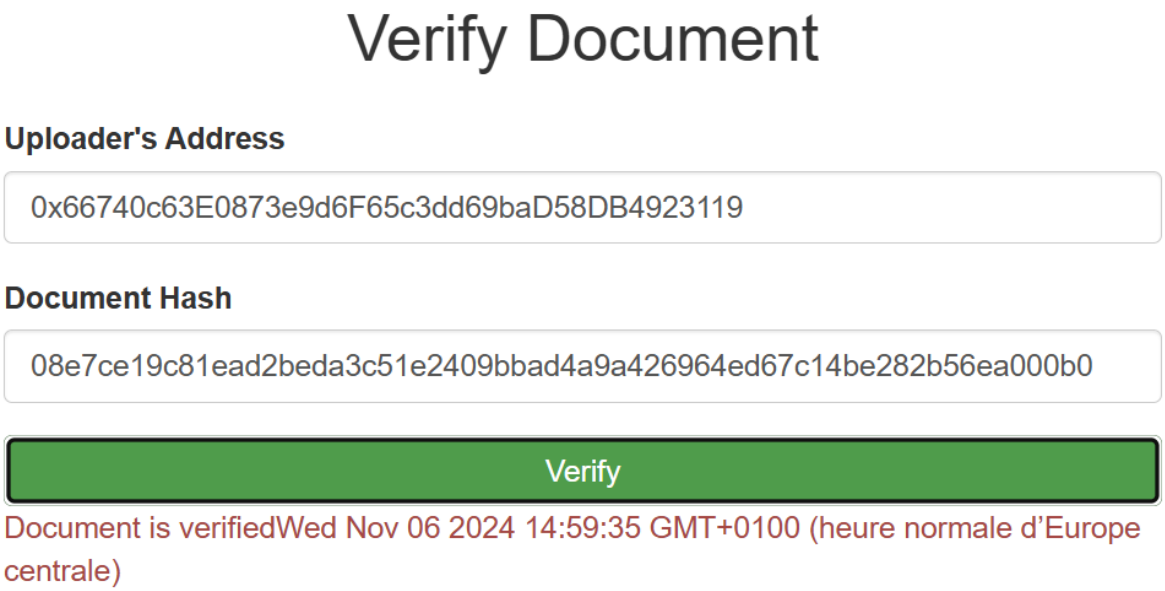


Upload Document:

In Ganache:



Verification:



Question 9:

Using blockchain in scientific prepublication has several advantages and challenges. Prepublication typically involves sharing research data, findings, and methodologies

before formal peer-reviewed publication, and blockchain can enhance this process. Here is a list:

- Permanent, Time-Stamped Records
- Improved Data Integrity
- Facilitating Open Science and Collaboration
- Enhanced Peer Review Process
- Protection of Intellectual Property

Question 10:

contracts/	contains the solidity source file (.sol) for smart contracts. The Pet-Shop box provides a smart contract called "Migrations.sol" used fordeployment.
migrations/	Truffle uses a migration system to deploy smart contracts. A Migration is a special smart contract that keeps track of changes.
test/	contains the test scripts (written in JavaScript or Solidity) for the smart contracts.
node_modules/	contains the node.js dependencies.
src/	contains client-side programs in HTML/CSS/JS and related resources such as images and fonts.
truffle-config.js	the Truffle configuration file.

DA51 Lab Session 8: Dapp Crowdfunding

Description:

This lab session focuses on developing a decentralized crowdfunding application (DApp) on Ethereum using Solidity and Truffle Suite. The DApp will allow users to create campaigns, contribute Ether, manage withdrawals, and implement a refund mechanism for unsuccessful campaigns. Deliverables include the Solidity contract source code, test cases, frontend code, and a brief report.

Question 2:

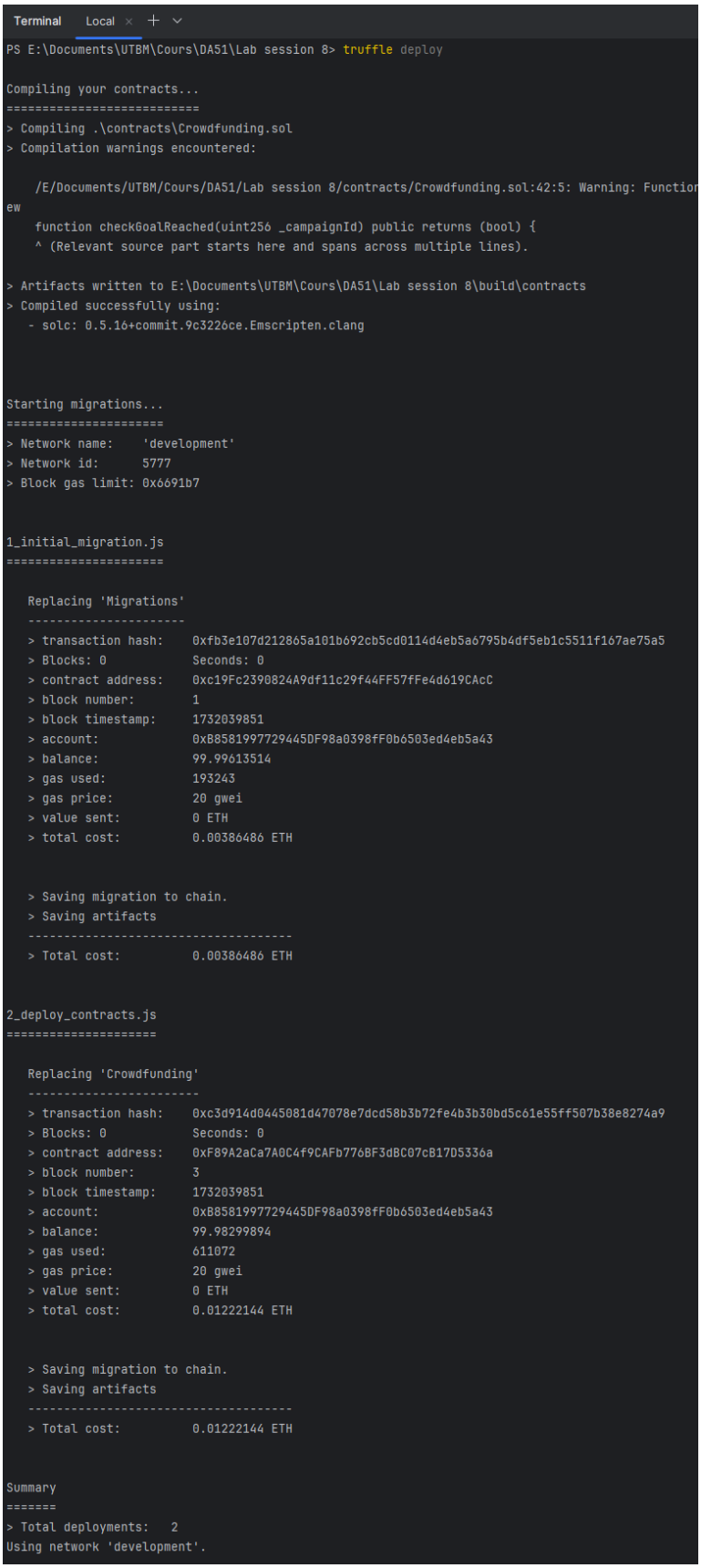
Smart Contract:

Crowdfunding.sol

```solidity
// SPDX-License-Identifier: UNLICENSED
pragma solidity >=0.4.22 <0.9.0;

contract Crowdfunding {
    struct Campaign {
        address payable creator;
        uint256 goal;
        uint256 deadline;
        uint256 balance;
        bool closed;
    }

    mapping(uint256 => Campaign) public campaigns;
    uint256 public campaignId;

    function startCampaign(uint256 _goal, uint256 _deadline) public returns (uint256) {
        require(_goal > 0, "Goal must be greater than 0");
        require(_deadline > block.timestamp, "Deadline must be in the future");

        campaigns[campaignId] = Campaign( creator: {
            creator: msg.sender,
            goal: _goal,
            deadline: _deadline,
            balance: 0,
            closed: false
        });

        campaignId++;
        return campaignId - 1;
    }

    function contribute(uint256 _campaignId) public payable returns (uint256) {
        Campaign storage campaign = campaigns[_campaignId];
        require(msg.value > 0, "Contribution must be greater than 0");
        require(!campaign.closed, "Campaign is closed");
        require(block.timestamp < campaign.deadline, "Deadline has passed");

        campaign.balance += msg.value;
        return campaign.balance;
    }

    function checkGoalReached(uint256 _campaignId) public returns (bool) {
        Campaign storage campaign = campaigns[_campaignId];
        require(!campaign.closed, "Campaign is closed");
        //require(block.timestamp >= campaign.deadline, "Deadline has not passed");

        if (campaign.balance >= campaign.goal) {
            return true;
        } else {
            return false;
        }
    }

    function closeCampaign(uint256 _campaignId) public {
        Campaign storage campaign = campaigns[_campaignId];
        require(!campaign.closed, "Campaign is already closed");

        campaign.closed = true;
    }

    function withdraw(uint256 _campaignId) public {
        Campaign storage campaign = campaigns[_campaignId];
        require(campaign.closed, "Campaign is not closed");
        require(msg.sender == campaign.creator, "Only the creator can withdraw funds");

        campaign.creator.transfer( value: campaign.balance);
    }
}
```

Question 3:

Compile and migration:



```
Terminal  Local

PS E:\Documents\UTBM\Cours\DA51\Lab session 8> truffle deploy

Compiling your contracts...
===========================
> Compiling .\contracts\Crowdfunding.sol
> Compilation warnings encountered:

    /E/Documents/UTBM/Cours/DA51/Lab session 8/contracts/Crowdfunding.sol:42:5: Warning: Functior
ew
    function checkGoalReached(uint256 _campaignId) public returns (bool) {
    ^ (Relevant source part starts here and spans across multiple lines).

> Artifacts written to E:\Documents\UTBM\Cours\DA51\Lab session 8\build\contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang



Starting migrations...
======================
> Network name:    'development'
> Network id:      5777
> Block gas limit: 0x6691b7


1_initial_migration.js
======================

   Replacing 'Migrations'
   ----------------------
   > transaction hash:    0xfb3e107d212865a101b692cb5cd0114d4eb5a6795b4df5eb1c5511f167ae75a5
   > Blocks: 0            Seconds: 0
   > contract address:    0xc19Fc2390824A9df11c29f44FF57fFe4d619CAcC
   > block number:        1
   > block timestamp:     1732039851
   > account:             0xB85819977294455DF98a0398fF0b6503ed4eb5a43
   > balance:             99.99613514
   > gas used:            193243
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.00386486 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.00386486 ETH


2_deploy_contracts.js
=====================

   Replacing 'Crowdfunding'
   ------------------------
   > transaction hash:    0xc3d914d0445081d47078e7dcd58b3b72fe4b3b30bd5c61e55ff507b38e8274a9
   > Blocks: 0            Seconds: 0
   > contract address:    0xF89A2aCa7A0C4f9CAFb776BF3dBC07cB17D5336a
   > block number:        3
   > block timestamp:     1732039851
   > account:             0xB85819977294455DF98a0398fF0b6503ed4eb5a43
   > balance:             99.98299894
   > gas used:            611072
   > gas price:           20 gwei
   > value sent:          0 ETH
   > total cost:          0.01222144 ETH


   > Saving migration to chain.
   > Saving artifacts
   -------------------------------------
   > Total cost:          0.01222144 ETH


Summary
=======
> Total deployments:   2
Using network 'development'.
```

Question 4:

Testing:

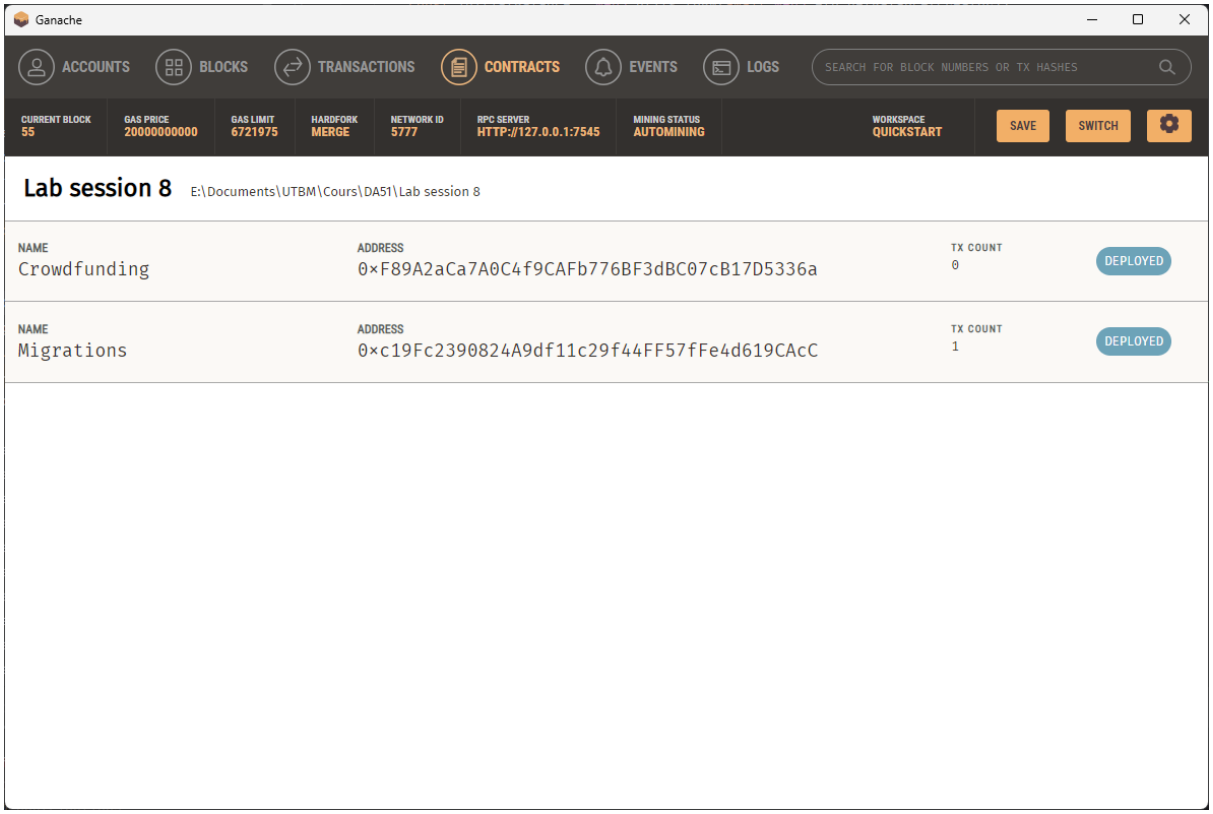
TestCrowdfunding.js

```
const Crowdfunding = artifacts.require("Crowdfunding");

contract("Crowdfunding", (accounts) :void => {
    let crowdfundingInstance;

    const [creator, contributor] = accounts;

    beforeEach(async () :Promise<void> => {
        crowdfundingInstance = await Crowdfunding.new();
    });

    it("should start a campaign", async () :Promise<void> => {
        const goal = web3.utils.toWei("10", "ether");
        const deadline :number = Math.floor( x: Date.now() / 1000) + 3600; // 1 hour from now

        await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });

        const campaign = await crowdfundingInstance.campaigns(0);
        assert.equal(campaign.creator, creator, "Campaign creator is incorrect");
        assert.equal(campaign.goal.toString(), goal, "Campaign goal is incorrect");
        assert.equal(campaign.deadline.toNumber(), deadline, "Campaign deadline is incorrect");
        assert.equal(campaign.balance.toString(), "0", "Campaign balance should start at 0");
        assert.equal(campaign.closed, false, "Campaign should not be closed initially");
    });

    it("should allow contributions", async () :Promise<void> => {
        const goal = web3.utils.toWei("10", "ether");
        const deadline :number = Math.floor( x: Date.now() / 1000) + 3600;

        await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });

        const contributionAmount = web3.utils.toWei("1", "ether");
        await crowdfundingInstance.contribute(0, { from: contributor, value: contributionAmount });

        const campaign = await crowdfundingInstance.campaigns(0);
        assert.equal(campaign.balance.toString(), contributionAmount, "Campaign balance should reflect the contribution");
    });

    it("should close the campaign and allow withdrawal when the goal is reached", async () :Promise<void> => {
        const goal = web3.utils.toWei("2", "ether");
        const deadline :number = Math.floor( x: Date.now() / 1000) + 3600;

        await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });

        await crowdfundingInstance.contribute(0, { from: contributor, value: web3.utils.toWei("2", "ether") });

        await crowdfundingInstance.checkGoalReached(0, { from: creator });

        const campaign = await crowdfundingInstance.campaigns(0);
        assert.equal(campaign.closed, true, "Campaign should be closed after goal is reached");

        const initialBalance = web3.utils.toBN(await web3.eth.getBalance(creator));
        await crowdfundingInstance.withdraw(0, { from: creator });

        const finalBalance = web3.utils.toBN(await web3.eth.getBalance(creator));
        assert(finalBalance.gt(initialBalance), "Creator should have withdrawn funds");
    });

    it("should fail to contribute after the campaign deadline", async () :Promise<void> => {
        const goal = web3.utils.toWei("10", "ether");
        const deadline :number = Math.floor( x: Date.now() / 1000) + 1; // 1 second from now

        await crowdfundingInstance.startCampaign(goal, deadline, { from: creator });

        await new Promise( executor: (resolve) :number => setTimeout(resolve, timeout: 2000));

        try {
            await crowdfundingInstance.contribute(0, { from: contributor, value: web3.utils.toWei("1", "ether") });
            assert.fail("Contribution should fail after the deadline");
        } catch (error) {
            assert(error.message.includes("Deadline has passed"), "Expected 'Deadline has passed' error");
        }
    });
});
```

Question 5:

```
PS E:\Documents\UTBM\Cours\DA51\Lab session 8> truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



  Contract: Crowdfunding
    √ should start a campaign
    √ should allow contributions
    √ should close the campaign and allow withdrawal when the goal is reached (59ms)
    √ should fail to contribute after the campaign deadline (2046ms)


  4 passing (2s)
```

Question 6-7:



Question 8-9-10:

Index.html:

index.html

```html
<!DOCTYPE html>
<html lang="en">
  <head...>
  <body>
  <h1>Crowdfunding DApp</h1>

  <!-- Start Campaign Section -->
  <h2>Start a Campaign</h2>
  <label for="goal">Goal (in Wei):</label>
  <input type="number" id="goal" placeholder="Enter goal in wei">
  <label for="deadline">Deadline (UNIX Timestamp):</label>
  <input type="number" id="deadline" placeholder="Enter deadline timestamp">
  <button class="btn btn-primary btn-block btn-start-campaign">Start Campaign</button>
  <p id="startCampaignResult"></p>

  <!-- Contribute Section -->
  <h2>Contribute to a Campaign</h2>
  <label for="campaignId">Campaign ID:</label>
  <input type="number" id="campaignId" placeholder="Enter campaign ID">
  <label for="amount">Amount (in Wei):</label>
  <input type="number" id="amount" placeholder="Enter amount to contribute">
  <button class="btn btn-primary btn-block btn-contribute">Contribute</button>
  <p id="contributeResult"></p>

  <!-- Check Goal Reached Section -->
  <h2>Check if Campaign Goal Reached</h2>
  <label for="campaignIdCheck">Campaign ID:</label>
  <input type="number" id="campaignIdCheck" placeholder="Enter campaign ID">
  <button class="btn btn-primary btn-block btn-check-goal">Check Goal</button>
  <p id="checkGoalResult"></p>

  <!-- Close the campaign -->
  <h2>Close a Campaign</h2>
  <label for="campaignIdClose">Campaign ID:</label>
  <input type="number" id="campaignIdClose" placeholder="Enter campaign ID">
  <button class="btn btn-primary btn-block btn-close">Close</button>
  <p id="closeResult"></p>

  <!-- Withdraw Section -->
  <h2>Withdraw Funds</h2>
  <label for="campaignIdWithdraw">Campaign ID:</label>
  <input type="number" id="campaignIdWithdraw" placeholder="Enter campaign ID">
  <button class="btn btn-primary btn-block btn-withdraw">Withdraw</button>
  <p id="withdrawResult"></p>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed -->
    <script src="js/bootstrap.min.js"></script>
    <script src="js/web3.min.js"></script>
    <script src="js/truffle-contract.js"></script>
    <script src="js/app.js"></script>
  </body>
</html>
```

App.js:

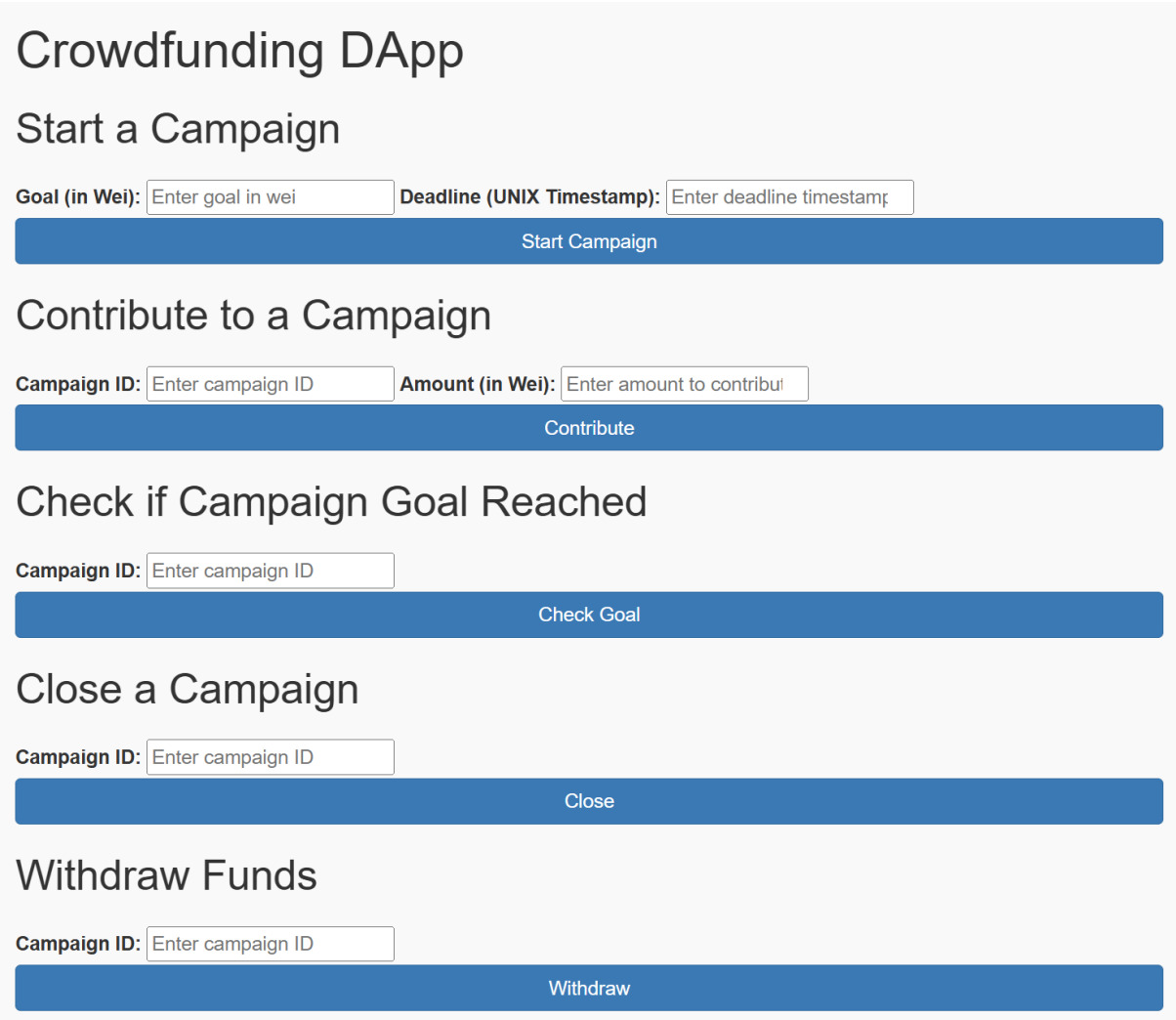
- Function to dialog with smart contract
    - The first screen is about starting a campaign, contributing to and check if the crowdfunding reaches the goal

```
startCampaignJs: function (event) :void {
    event.preventDefault();
    var doc;
    const goal = $('@ #goal').val();
    if (!goal) {
        alert('Please enter a goal');
        return;
    }
    const deadline = $('@ #deadline').val();
    if (!deadline) {
        alert('Please enter a deadline');
        return;
    }
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            doc = instance;
            return doc.startCampaign(goal, deadline, {from: account});
        }).then(function (result) :void {
            $('@ #startCampaignResult').text( value: 'Campaign started successfully' + result);
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},

contributeJs: function (event) :void {
    event.preventDefault();
    var doc;
    const campaignId = $('@ #campaignId').val();
    const amount = $('@ #amount').val();
    if (!amount) {
        alert('Please enter an amount');
        return;
    }
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            doc = instance;
            return doc.contribute(campaignId, {from: account, value: amount});
        }).then(function (result) :void {
            $('@ #contributeResult').text( value: 'Contribution successful, balance:' + result);
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},

checkGoalJs: function (event) :void {
    event.preventDefault();
    const campaignId = $('@ #campaignIdCheck').val();
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            return instance.checkGoalReached(campaignId, {from: account});
        }).then(function (result) :void {
            if (result) {
                $('@ #checkGoalResult').text( value: 'Goal reached');
            } else {
                $('@ #checkGoalResult').text( value: 'Goal not reached');
            }
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},
```

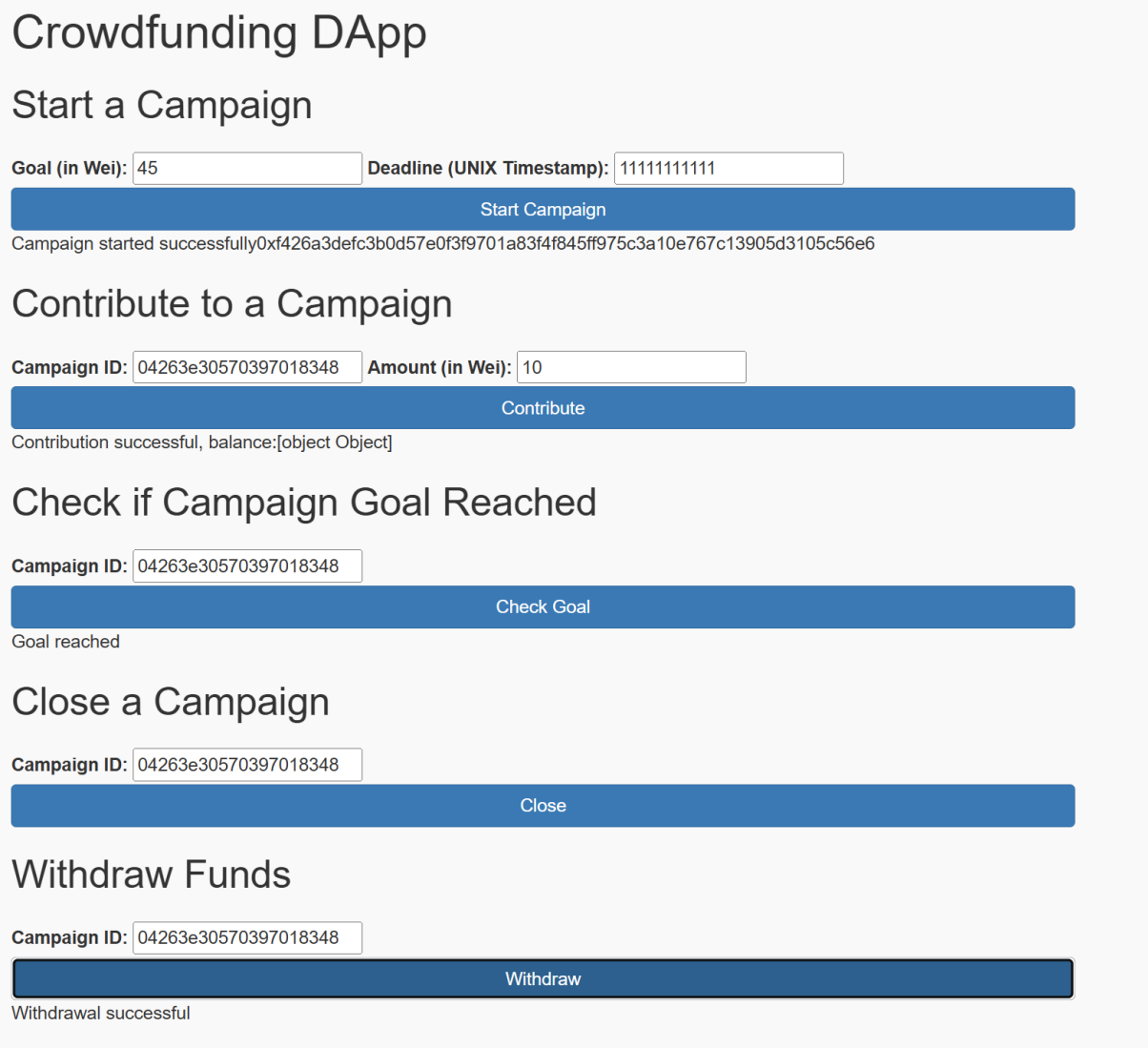
- The second screen is about closing and withdrawing a campaign

```
closeCampaignJs: function (event) :void {
    event.preventDefault();
    const campaignId = $('#campaignIdClose').val();
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            return instance.close(campaignId, {from: account});
        }).then(function (result) :void {
            $('#closeResult').text( value: 'Campaign closed');
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
},

withdrawJs: function (event) :void {
    event.preventDefault();
    const campaignId = $('#campaignIdClose').val();
    web3.eth.getAccounts(function (error, accounts) :void {
        if (error) {
            console.log(error);
        }
        var account = accounts[0];
        App.contracts.Crowdfunding.deployed().then(function (instance) {
            return instance.withdraw(campaignId, {from: account});
        }).then(function (result) :void {
            $('#withdrawResult').text( value: 'Withdrawal successful');
        }).catch(function (err) :void {
            console.log(err.message);
        });
    });
}
```

Result:


Crowdfunding DApp
Start a Campaign
Goal (in Wei):
Enter goal in wei
Deadline (UNIX Timestamp):
Enter deadline timestamp
Start Campaign
Contribute to a Campaign
Campaign ID:
Enter campaign ID
Amount (in Wei):
Enter amount to contribut
Contribute
Check if Campaign Goal Reached
Campaign ID:
Enter campaign ID
Check Goal
Close a Campaign
Campaign ID:
Enter campaign ID
Close
Withdraw Funds
Campaign ID:
Enter campaign ID
Withdraw

Function tests:



DA51 Lab Session 9: Bespoke Ethereum Blockchain Tokens

Description:

This guide outlines the process of creating a custom ERC-20 token on the Ethereum blockchain using the Remix IDE. It covers setting up the development environment, defining the ERC-20 token contract, and implementing essential functions like transfer, approve, and balanceOf.

My contract file :

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

import "Utils.sol";

contract Lab9Token {
    using Utils for *;
    string public constant name = "DA51-Lab9";
    string public constant symbol = "Lab9";
    uint8 public constant decimals = 18;

    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
    event Transfer(address indexed from, address indexed to, uint token);
    mapping(address => uint256) balances;
    mapping(address => mapping(address=> uint256)) allowed;
    uint256 totalSupply_;


    constructor(uint256 initialSupply) {    infinite gas 820600 gas
        totalSupply_ = initialSupply * 10 ** uint256(decimals);
        balances[msg.sender] = totalSupply_;
    }

    function transfer(address to, uint tokens) public returns (bool) {    infinite gas
        require(to != address(0), "Invalid address");
        require(balances[msg.sender] >= tokens, "Insufficient balance");

        balances[msg.sender] -= tokens;
        balances[to] += tokens;
        emit Transfer(msg.sender, to, tokens);
        return true;
    }

    function approve(address spender, uint tokens) public returns (bool) {    infinite gas
        require(spender != address(0), "Invalid spender address");

        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }
```

This first screen shot contains all the variables needed in this contract, the definition of the constructor and some functions like transfer and approve.

The second screen shot contains the last function needed as balanceOf, transferFrom and allowedT (to get the allowance)

```
    function balanceOf(address tokenOwner) public view returns (uint) {    2829 gas
        return balances[tokenOwner];
    }

    function transferFrom(address from, address to, uint tokens) public returns (bool) {
        require(to != address(0), "Invalid to address");
        require(from != address(0), "Invalid from address");
        require(balances[from] >= tokens, "Insufficient balance");
        require(allowed[from][msg.sender] >= tokens, "Allowance exceeded");

        balances[from] -= tokens;
        balances[to] += tokens;
        allowed[from][msg.sender] -= tokens;
        emit Transfer(from, to, tokens);
        return true;
    }

    function allowedT(address tokenOwner, address spender) public view returns (uint256)
    {
        return allowed[tokenOwner][spender];
    }
}
```

My test file:

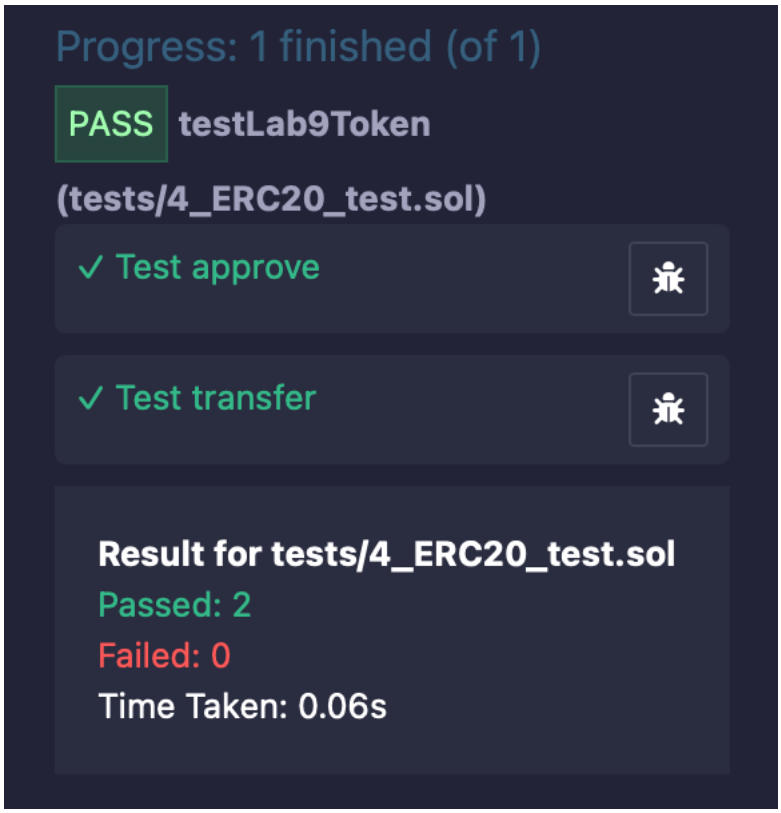
```solidity
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.4.22 <0.9.0;

import "remix_tests.sol";
import "../contracts/4_ERC20.sol";

contract testLab9Token {
    Lab9Token lab9Token;

    address owner;
    address account1 = address(0x123);
    address account2 = address(0x456);


    function beforeAll() public {    infinite gas
        lab9Token = new Lab9Token(1000);
        owner = address(this);
    }

    function testApprove() public {    infinite gas
        uint approveAmount = 1000;
        bool success = lab9Token.approve(account1, approveAmount);
        Assert.equal(success, true, "Approval should succeed");

        uint allowance = lab9Token.allowedT(owner, account1);
        Assert.equal(allowance, approveAmount, "Allowance is not set correctly");
    }
```

This first screen shot contains all the variables needed in this contract and the first test: tertApprove.

The second screen shot show the tow last tests : testTransfer to test if a transfer can be made based on several aspect and testTransferFailsOnInsufficientBalance to test if the we get an error when the balance is to low for a specific transfer.

```solidity
    function testTransfer() public {    infinite gas
        uint initialOwnerBalance = lab9Token.balanceOf(owner);
        uint transferAmount = 500;

        bool success = lab9Token.transfer(account1, transferAmount);
        Assert.equal(success, true, "Transfer should succeed");

        uint newOwnerBalance = lab9Token.balanceOf(owner);
        uint recipientBalance = lab9Token.balanceOf(account1);

        Assert.equal(
            newOwnerBalance,
            initialOwnerBalance - transferAmount,
            "Owner balance incorrect after transfer"
        );
        Assert.equal(
            recipientBalance,
            transferAmount,
            "Recipient balance incorrect after transfer"
        );
    }

    function testTransferFailsOnInsufficientBalance() public {    infinite gas
        uint transferAmount = 10000;

        try lab9Token.transfer(account1, transferAmount) {

        } catch Error(string memory reason) {
            Assert.equal(reason, "Insufficient balance", "Incorrect revert reason");
        }
    }
}
```

The results of the tests are correct as we can see in this screen shot:



Conclusion:

At the end of this Lab session, we are able to create a custom ERC-20 token for our Ethereum-based applications, understanding the requirements and functionalities of the ERC-20 standard.

DA51 Lab Session 1 Part II : Python

Description:

Lab session 1 (Part II) focuses on distinguishing service, interface, and protocol, implementing a connection-oriented service, and using Python's socket library. Students will implement a service for reliable data transmission between a client and server.

The tow files:

Server.py

server.py

```python
from socket import *

s = socket(AF_INET, SOCK_STREAM)
s.bind(("127.0.0.1", 12345))
s.listen(5)
while True:  # forever
    (conn, addr) = s.accept()  # accept connection from client
    data = conn.recv(1024)  # receive data from client
    if not data: break  # stop if client stopped
    print(data)
    conn.send((str(data) + "*").encode('utf-8'))  # return sent data plus an "*"
conn.close()  # close the connection
```

Client.py

client.py

```python
from socket import *

s = socket(AF_INET, SOCK_STREAM)
s.connect(("localhost", 12345))  # connect to server (block until accepted)
s.send('Hello, world'.encode('utf-8'))  # send some data
data = s.recv(1024) # receive the response
print(data) # print the result
s.close() # close the connectio
```

DA51 Lab Session 2 Part II : Le protocole MQTT

Question 11 :

On peut consulter la spécification complète sur le site officiel de l'OASIS ou directement via leur documentation officielle.

Question 12 :

Broker (serveur MQTT) : Le composant central qui distribue les messages entre les producteurs (publishers) et les consommateurs (subscribers).

Producers : Dispositifs ou programmes qui envoient des messages au broker sur des "topics".

Consumers : Dispositifs ou programmes qui s'abonnent à des "topics" pour recevoir des messages.

Topic : Un canal ou sujet spécifique sur lequel les messages sont publiés ou reçus.

Question 13 :

MQTT utilise un modèle "publish/subscribe". Les producteurs envoient des messages sur des "topics" au broker, qui les distribue aux consommateurs abonnés aux mêmes "topics".

Question 14 :

Les jokers permettent de s'abonner à plusieurs topics à la fois :

+ : Remplace un seul niveau de hiérarchie.

Exemple : home/+/temperature couvre home/livingroom/temperature et home/kitchen/temperature.

: Remplace plusieurs niveaux.

Exemple : home/# couvre home/livingroom/temperature et home/garage/door.

Question 15 :

Mosquitto, HiveMQ, RabbitMQ, Eclipse MQTT broker

Question 16 :

QoS 0 : Au moins une fois, sans garantie.

QoS 1 : Au moins une fois, avec confirmation de réception.

QoS 2 : Une seule fois, garantie d'évitement des doublons.

Question 17 :

sudo apt update && sudo apt upgrade

Question 18 :

Ajoute un dépôt de packages à votre système. Dans ce cas, il ajoute le dépôt Mosquitto.

Question 19-20 :

sudo systemctl status mosquito

systemctl est utilisé pour contrôler et vérifier les services sur les systèmes basés sur SystemD.

Question 22 :

Terminal 1 : mosquitto_sub -t unSujet

Terminal 2 : mosquitto_pub -t unSujet -m "unMessage"

Question 24-25 :

-t : Spécifie le "topic".

-m : Définit le message à publier.

Question 27-28 :

Localisation typique : /etc/mosquitto/mosquitto.conf

Question 29-30 :

Installer NodeJS : Suivez ce guide.

Installer MQTT.js : npm install -g mqtt

Question 31-32:

Mosquitto_sub -v -h mqtt.eclipse.org -t '$SYS/#'

Question 33 :

mqtt subscribe -v -h iot.eclipse.org '#' 2>/dev/null | head -10 | tee mqtt.log

- Cela s'abonne au topic racine # pour lire tous les messages, les dirige vers un fichier log, et affiche les 10 premiers messages.

Question 34 :

Risques :

- Détournement de données.
- Envoi de données falsifiées.
- Mise en danger des systèmes connectés.

DA51 Lab Session 3 Part II: Securing a Distributed Messaging Service in a Distributed System

Introduction

This lab session focused on building a secure, scalable, and reliable messaging service by integrating sockets, MQTT, and blockchain technologies. It highlighted the use of advanced security protocols, including ECDH, ECDSA, and HMAC, to ensure confidentiality, authenticity, and integrity. The project was implemented in three phases: secure messaging with sockets and MQTT, enhanced MQTT messaging, and blockchain integration for immutable logging.

Technologies and Security Mechanisms

1. Sockets

Sockets were utilized to establish a direct and secure connection between client and server. Using Python's socket library, an ECDH key exchange was implemented to create a shared secret, enabling encrypted communication. Additionally, ECDSA provided message authenticity through digital signatures, and HMAC ensured message integrity.

2. MQTT

MQTT, a lightweight publish-subscribe protocol, was employed for efficient and scalable message routing. The paho-mqtt library facilitated secure key exchange over MQTT topics, with subsequent message transmissions incorporating ECDSA signatures and HMACs for security.

3. Blockchain

Blockchain served as an immutable audit log for all exchanged messages. By logging verified MQTT messages with metadata, the system ensured an indelible record of communications. Blockchain integrity validation guaranteed that no messages were altered post-transmission.

Implementation Details

Part A: Secure Messaging with Sockets and MQTT

1. **Socket Setup for Key Exchange:**
    - Client and server established connections using Python's socket library.
    - ECDH key exchange generated a shared secret for secure communication.
2. **Key Exchange Using ECDH:**
    - ECDH key pairs were generated and exchanged between client and server.
    - A shared secret was computed to derive session keys for encryption and HMAC.
3. **Message Signing with ECDSA:**
    - Messages were signed using ECDSA private keys.
    - Signatures were verified on the receiving end to authenticate the sender.
4. **Message Integrity with HMAC:**
    - HMACs were generated using the shared secret to ensure message integrity.
    - Both the signed message and HMAC were transmitted for verification.
5. **Transition to MQTT:**
    - MQTT connection was established using Mosquitto broker.
    - Messages were routed through MQTT topics with ECDSA signatures and HMACs.

Part B: Secure Messaging with MQTT

1. **Setup with paho-mqtt:**
    - The paho-mqtt library was used to implement publish-subscribe messaging.
    - ECDH key exchange was re-established over secure MQTT topics.
2. **Secure Messaging Workflow:**
    - Messages were published with ECDSA signatures and HMACs.
    - Recipients verified signatures and HMACs for authenticity and integrity.
3. **QoS and Security:**
    - MQTT QoS levels were configured to explore delivery guarantees.
    - SSL/TLS was considered for transport-layer security, complementing ECDH, ECDSA, and HMAC for end-to-end security.

Part C: Blockchain Integration

1. **Blockchain Setup:**
    - A simple blockchain was created to log verified MQTT messages.
    - Each block contained metadata, including timestamps, sender details, and HMACs.
2. **Logging Messages to Blockchain:**
    - Verified messages were added to the blockchain, ensuring an immutable record.
3. **Verification of Integrity:**
    - Blockchain validation detected tampering by verifying the chain's integrity.
    - Any modifications to a block resulted in a broken chain.
4. **End-to-End Testing:**
    - The system was tested by running all components together.
    - Tests verified message flow, blockchain integrity, and response to tampered messages.

Security Analysis

1. **ECDH:**

Enabled secure key exchange to establish a shared secret for encryption and HMAC generation.

2. **ECDSA:**

Provided message authenticity through digital signatures, preventing forgery.

3. **HMAC:**

Ensured integrity by validating that messages were not altered during transmission.

Challenges and Resolutions

1. **Challenge:** Implementing seamless integration between sockets, MQTT, and blockchain.

Resolution: Modularized each component and ensured standardized communication formats.

2. **Challenge:** Verifying blockchain integrity with large datasets.

Resolution: Optimized hashing mechanisms and implemented incremental validation.

3. **Challenge:** Handling MQTT QoS levels under high network latency.

Resolution: Adjusted QoS settings and implemented retries for reliable message delivery.

Potential Real-World Applications

1. **IoT Systems:**

Secure communication between distributed sensors and actuators.

2. **Finance:**

Immutable logging of financial transactions for audit and compliance.

3. **Secure Communications:**

End-to-end encrypted messaging applications for personal and corporate use.

Conclusion

This lab session provided a comprehensive understanding of building secure distributed messaging systems. By integrating sockets, MQTT, and blockchain, it demonstrated the synergy between networking protocols and security mechanisms. The project highlights the practicality of using these technologies in real-world applications requiring confidentiality, authenticity, integrity, and accountability.