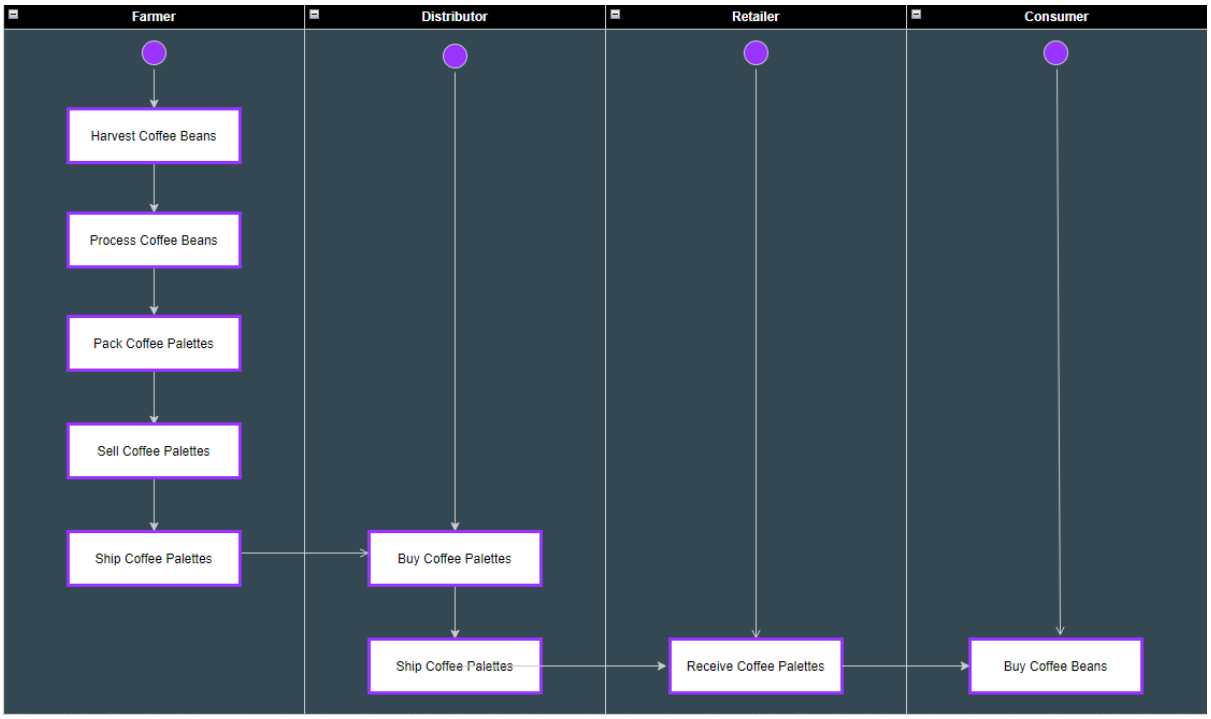
Project – Ethereum Dapp for Tracking Items through Supply Chain

This document includes all the required documentation that was required. This is the work performed by Luke Vella Critien in fulfilment for the *Blockchain Architecture* course.

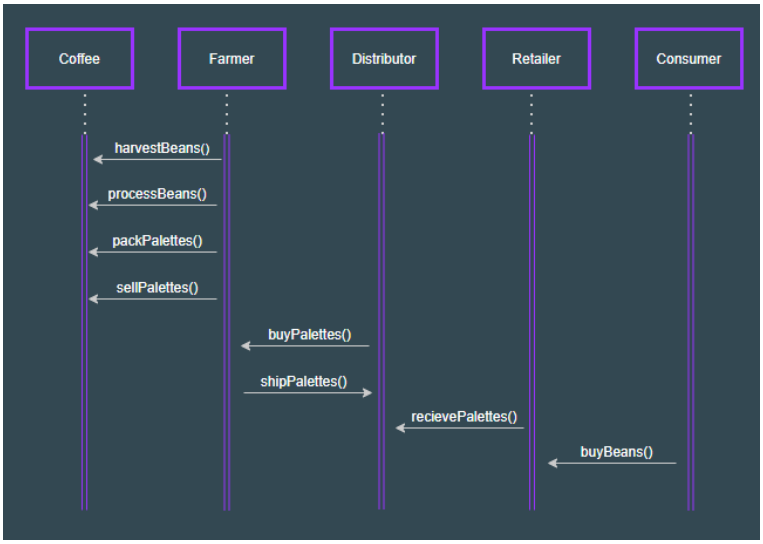
Part 1 – Plan the project with write-ups

Requirement 1 – UML

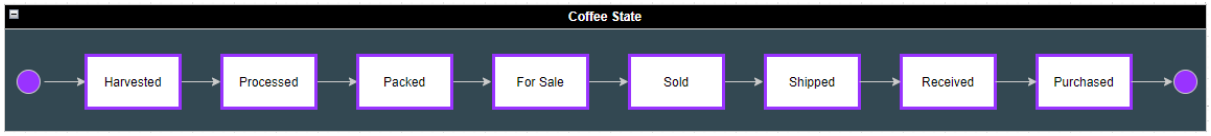
- Activity Diagram



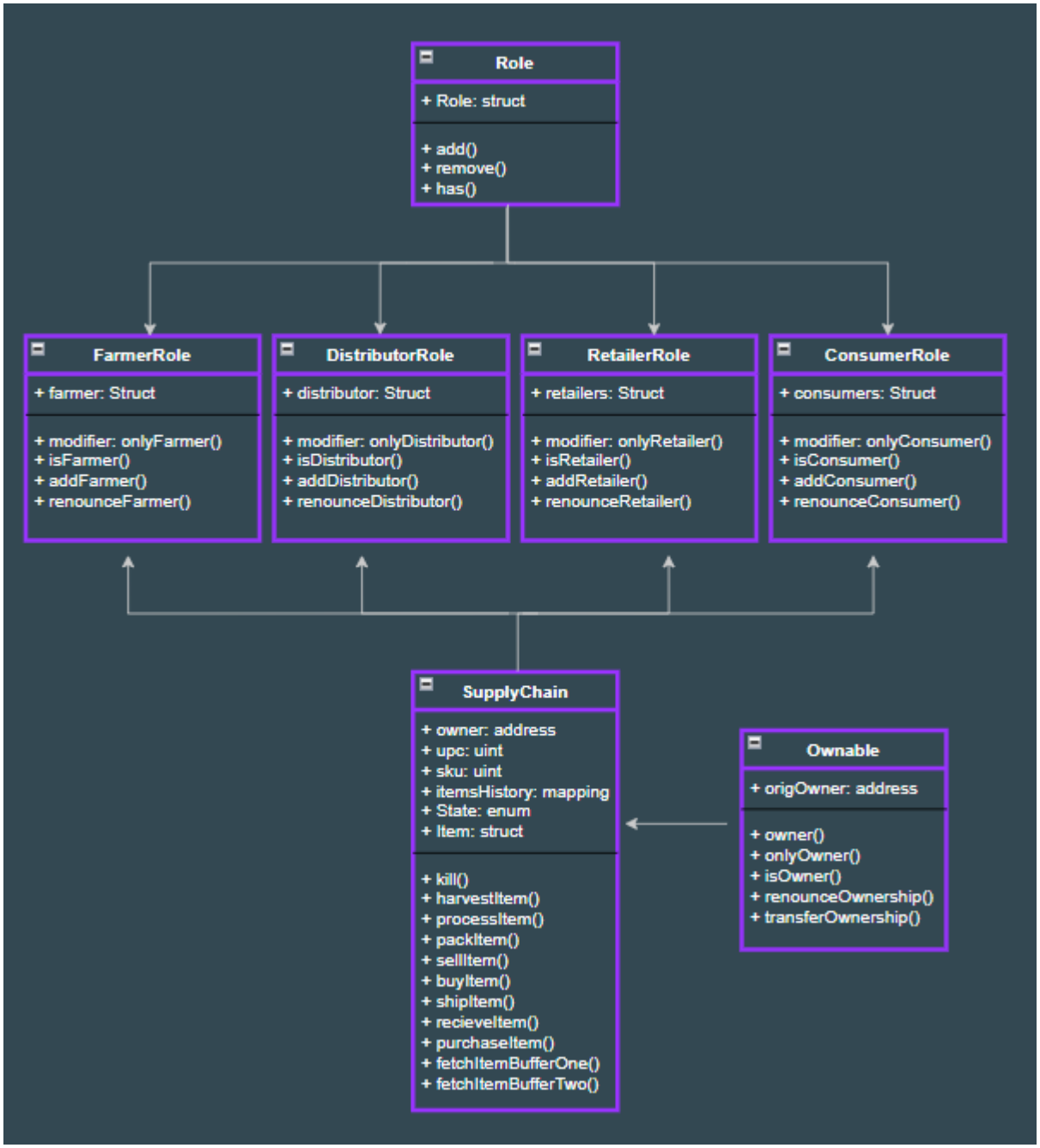
- Sequence Diagram



- State Diagram



- Classes (Data Model)



Requirement 2 – Libraries

Below is a list of libraries that were used in this project

- **Web3.js** – this is a collection of libraries that provides the possibility to interact to an Ethereum node. This is important as we used the Meta Mask which is an Ethereum supported browser. The version used was **0.20.6**.
- **Solidity** – this is the Ethereum smart contract language. Without solidity we would not have been able to write the contracts included in this project. The version used was **0.4.24**
- **Truffle** – this is a development environment which allowed us to compile and test the smart contracts on the EVM. This library made the development process much easier. The version used was **4.1.14**
- **Node –** this is used to be able to handle asynchronous activities. It is important to be able to perform the contractual tasks on the blockchain. The version used was **10.13.0**
- **Npm –** this is a package manager for node.js which helps in installing and configuring the required libraries for the project. The version used was **6.4.1**

Requirement 3 – IPFS

No IPFS was used to implement this project.

Part 2 – Write Smart Contracts

All the code related to the smart contracts can be found in the contracts folder which has three sub directories

- Coffee access control
  - This subfolder has a collection of 4 contracts *Roles, FarmerRole, DistributorRole, RetailerRole* and *ConsumerRole*
- Coffee base
  - This folder has one contract named *SupplyChain* which contains the core functionality related to data storage, constants, data types and functions
- Coffee core
  - This includes a single contract named *Ownable* which controls ownership and its transfer

Part 3 – Test smart contract code coverage

In this section 10 different tests were included to ensure that every function that was implemented in the smart contracts is correct. All the code related to the tests can be found in the *TestSupplychain.js* file which resides in the *Test* folder.

Below is a screen shot which shows the Truffle Development environment running successfully

```
PS C:\Users\lukev\OneDrive\Desktop\Luke Version\project-6> truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) c88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) aa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0f62d96d6675f32685bbdb8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 8d5366123cb560bb606379f90a0bfd4769eecc0557f1b362dcae9012b548b1e5
```

Following this is a screenshot showing that all the 10 test have been executed successfully

```
truffle(develop)> test
Using network 'develop'.

ganache-cli accounts used here...
Contract Owner: accounts[0]  0x627306090abab3a6e1400e9345bc60c78a8bef57
Farmer: accounts[1]  0xf17f52151ebef6c7334fad080c5704d77216b732
Distributor: accounts[2]  0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
Retailer: accounts[3]  0x821aea9a577a9b44299b9c15c88cf3087f3b5544
Consumer: accounts[4]  0x0d1d4e623d10f9fba5db95830f7d3839406c6af2


  Contract: SupplyChain
    √ 1 - Testing smart contract function harvestItem() that allows a farmer to harvest coffee (290ms)
    √ 2 - Testing smart contract function processItem() that allows a farmer to process coffee (131ms)
    √ 3 - Testing smart contract function packItem() that allows a farmer to pack coffee (125ms)
    √ 4 - Testing smart contract function sellItem() that allows a farmer to sell coffee (130ms)
    √ 5 - Testing smart contract function buyItem() that allows a distributor to buy coffee (120ms)
    √ 6 - Testing smart contract function shipItem() that allows a distributor to ship coffee (117ms)
    √ 7 - Testing smart contract function receiveItem() that allows a retailer to mark coffee received (194ms)
    √ 8 - Testing smart contract function purchaseItem() that allows a consumer to purchase coffee (192ms)
    √ 9 - Testing smart contract function fetchItemBufferOne() that allows anyone to fetch item details from bl
ockchain (40ms)
    √ 10 - Testing smart contract function fetchItemBufferTwo() that allows anyone to fetch item details from b
lockchain


  10 passing (2s)
```

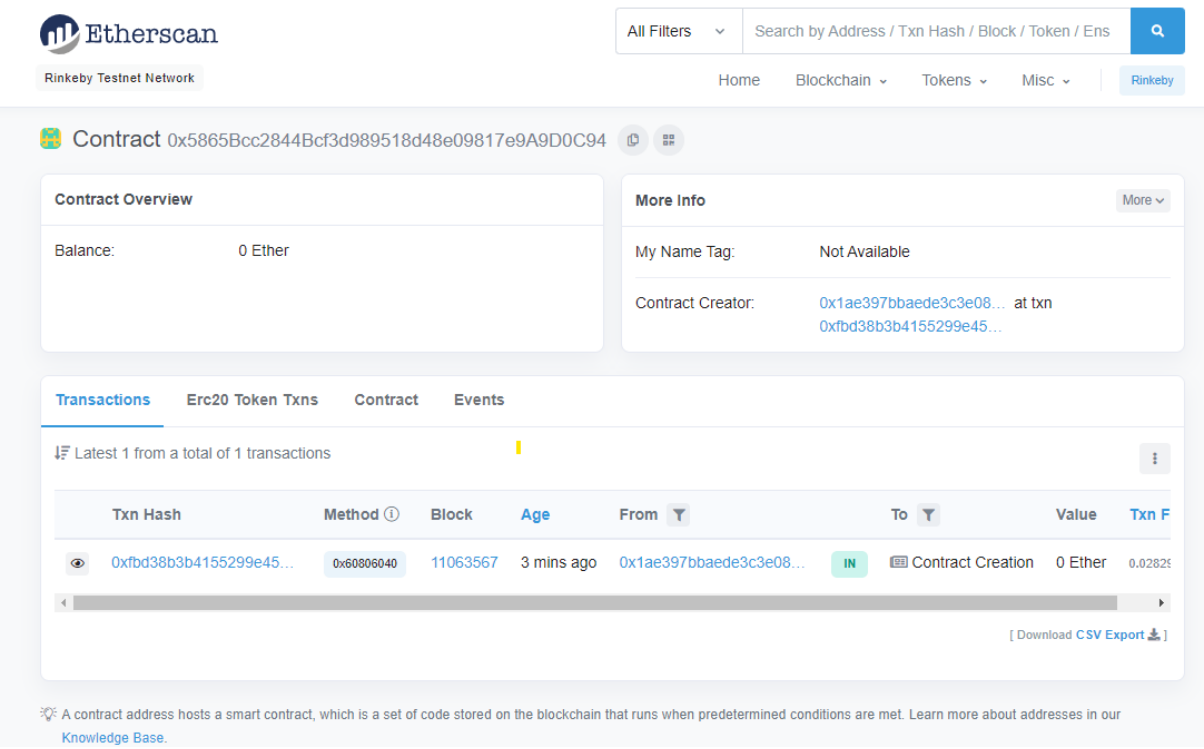
Part 4 – Deploy smart contracts on a public test network

After compiling all the smart contracts and executing all the test correctly, the next phase was to deploy it on a Rinkeby test network. Below is a screenshot from the CLI, generated on deployment which shows the addresses per contract including the Supply Chain address

```
Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0xfe433089d326867f13072ad2550843080196a35a2408194f3aabcb698a244a0c
  Migrations: 0x14645715bf5328e493cf876287038e0068c64afb
Saving successful migration to network...
  ... 0x3af6de5a8ca9fcfaf55cafb9a1e008fa765b5bca91f9d6585965e6b8ae94cab5
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying FarmerRole...
  ... 0xa78108892baa7dbc380e095f67b4ddc68893ceec34f9b61d9286965add9e80e4
  FarmerRole: 0x178c64572eb29043ac2d09c644d9a9a25cc52925
  Deploying DistributorRole...
  ... 0x98086baf6680f9a458d829d1ac14a2ff4e2ce2f7c65042d805618319ccef1db9
  DistributorRole: 0xd1f9e17582bf21ee0e042b18e0466b2932ad1b4d
  Deploying RetailerRole...
  ... 0x97d3ffe93bc623d2f8c9c1eb28f7f18bb12c5684b7037dec44287796be025e6c
  RetailerRole: 0x61d96461bda0d034e95c9a47133e02b099f5c664
  Deploying ConsumerRole...
  ... 0x045a52a0b3973ad40d6455124fb493fa8c957b0a7c804c809575cf8ff84570fd
  ConsumerRole: 0xb676dd37505d3425f9a8b6ca0e4e1cc281958b37
  Deploying SupplyChain...
  ... 0xfbd38b3b4155299e45ce8569ad33f33b01009892ddc5067f030db39dde4eea44
  SupplyChain: 0x5865bcc2844bcf3d989518d48e09817e9a9d0c94
Saving successful migration to network...
  ... 0x282b4d39a30d4cc45d928be82f47f5904ced6129c69ed8444910a9649a0d3c27
Saving artifacts...
PS C:\Users\lukev\OneDrive\Desktop\Luke Version\project-6>
```

Below you can find:

- **Supply Chain Contract Address**: 0x5865bcc2844bcf3d989518d48e09817e9a9d0c94
- Proof from the blockchain explorer



Below you can find:

- **Transaction URL:** https://rinkeby.etherscan.io/tx/0x282b4d39a30d4cc45d928be82f47f5904ced6129c69ed8444910a9649a0d3c27
- **Transaction Hash:** 0x282b4d39a30d4cc45d928be82f47f5904ced6129c69ed8444910a9649a0d3c27
- **Contract Address:** 0x14645715bf5328e493cf876287038e0068c64afb
- Proof from the blockchain explorer

Etherscan

Rinkeby Testnet Network

All Filters | Search by Address / Txn Hash / Block / Token / Ens

Home | Blockchain | Tokens | Misc | Rinkeby

Transaction Details

Overview | State

[This is a Rinkeby **Testnet** transaction only]

Transaction Hash:	0x282b4d39a30d4cc45d928be82f47f5904ced6129c69ed8444910a9649a0d3c27
Status:	Success
Block:	11063568 3 Block Confirmations
Timestamp:	1 min ago (Jul-21-2022 02:16:10 PM +UTC)
From:	0x1ae397bbaede3c3e0830a08aff94c29898c01d1f
To:	Contract 0x14645715bf5328e493cf876287038e0068c64afb
Value:	0 Ether ($0.00)
Transaction Fee:	0.00028648 Ether ($0.00)
Gas Price:	0.00000001 Ether (10 Gwei)

Click to see More

A transaction is a cryptographically signed instruction from an account that changes the state of the blockchain. Block explorers track the details of all transactions in the network. Learn more about transactions in our Knowledge Base.

Part 5 – Modify client code to interact with smart contracts

All the requirements for the front end operations to go through the whole lifecycle were tested and no issues were encountered