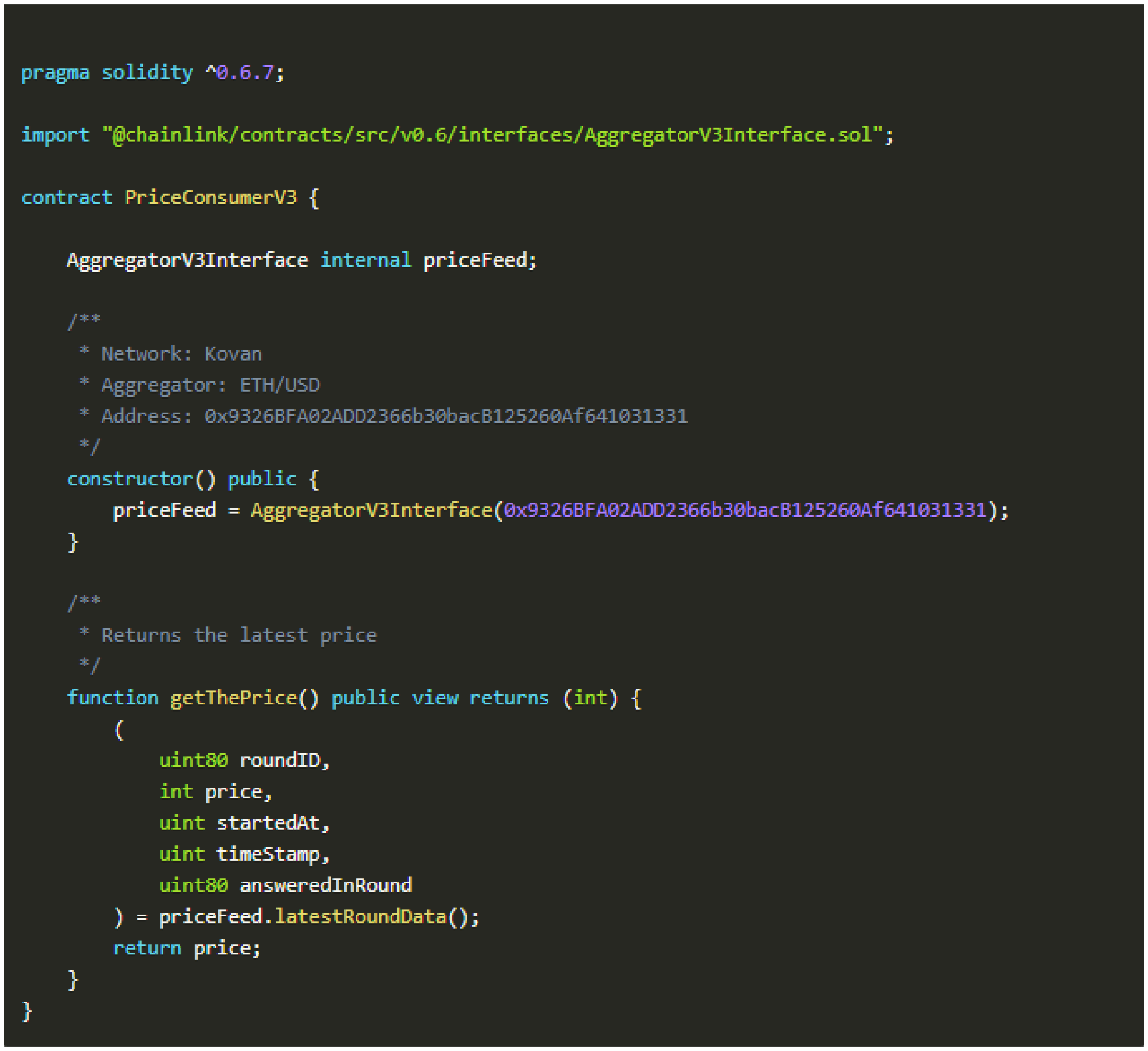
Monkhub internship assignment

- Chainlink Price Feeds are the quickest way to connect smart contracts to the real-world market prices of assets.
- The latest price of Ethereum (ETH) inside smart contracts is fetched using the ETH/USD Price Feed on the Kovan testnet.
- To consume price data, your smart contract should reference AggregatorV3Interface, which defines the external functions implemented by Price Feeds.

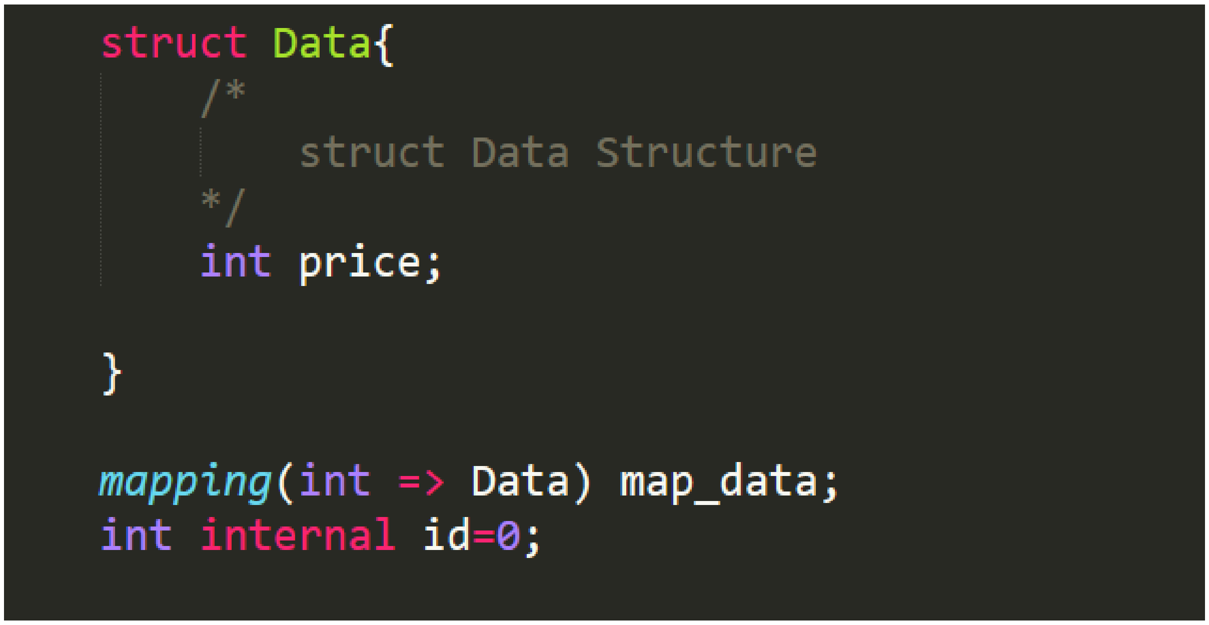
1. Fetches Current Data Price From Any Oracle

[Code Snippet]

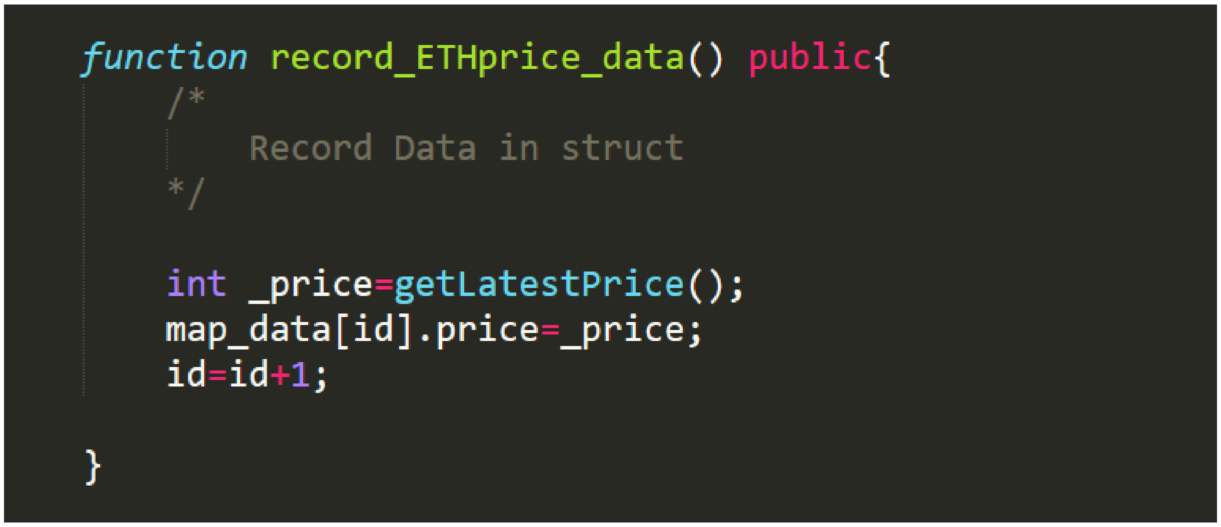


```
pragma solidity ^0.6.7;

import "@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol";

contract PriceConsumerV3 {

    AggregatorV3Interface internal priceFeed;

    /**
     * Network: Kovan
     * Aggregator: ETH/USD
     * Address: 0x9326BFA02ADD2366b30bacB125260Af641031331
     */
    constructor() public {
        priceFeed = AggregatorV3Interface(0x9326BFA02ADD2366b30bacB125260Af641031331);
    }

    /**
     * Returns the latest price
     */
    function getThePrice() public view returns (int) {
        (
            uint80 roundID,
            int price,
            uint startedAt,
            uint timeStamp,
            uint80 answeredInRound
        ) = priceFeed.latestRoundData();
        return price;
    }
}
```

2. Records that data in a struct
[Code Snippet]



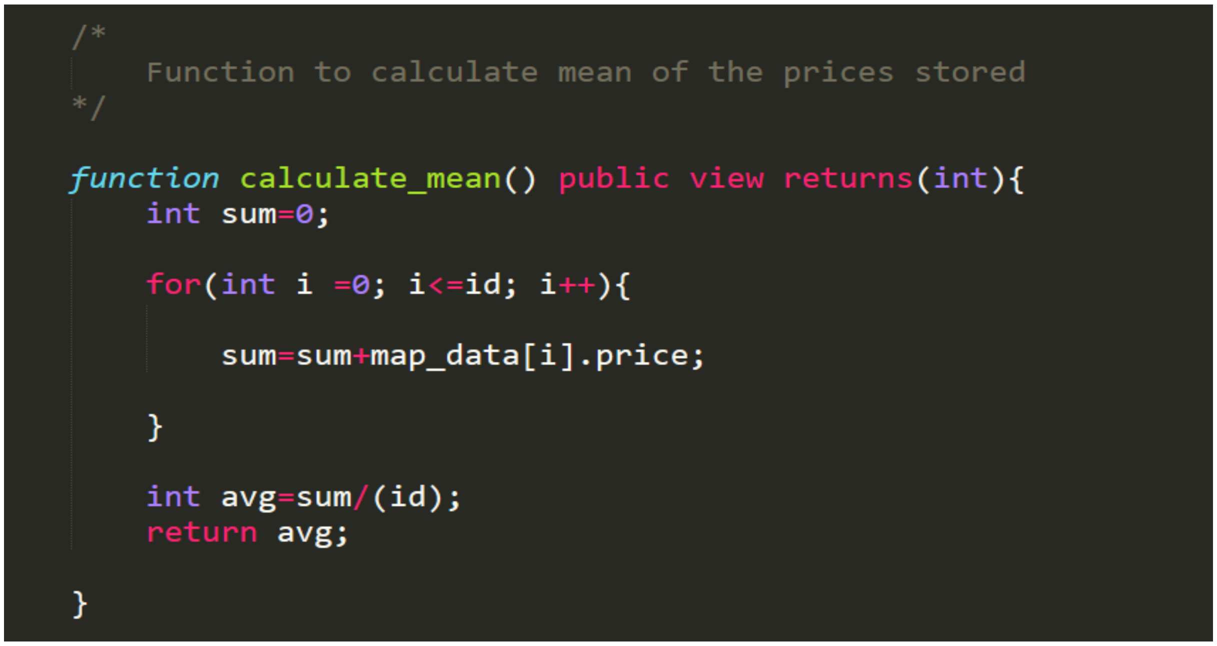
```
struct Data{
        /*
            struct Data Structure
        */
        int price;

}

mapping(int => Data) map_data;
int internal id=0;
```



```
function record_ETHprice_data() public{
        /*
            Record Data in struct
        */

        int _price=getLatestPrice();
        map_data[id].price=_price;
        id=id+1;

}
```

3. **A function to calculate mean of the prices stored**

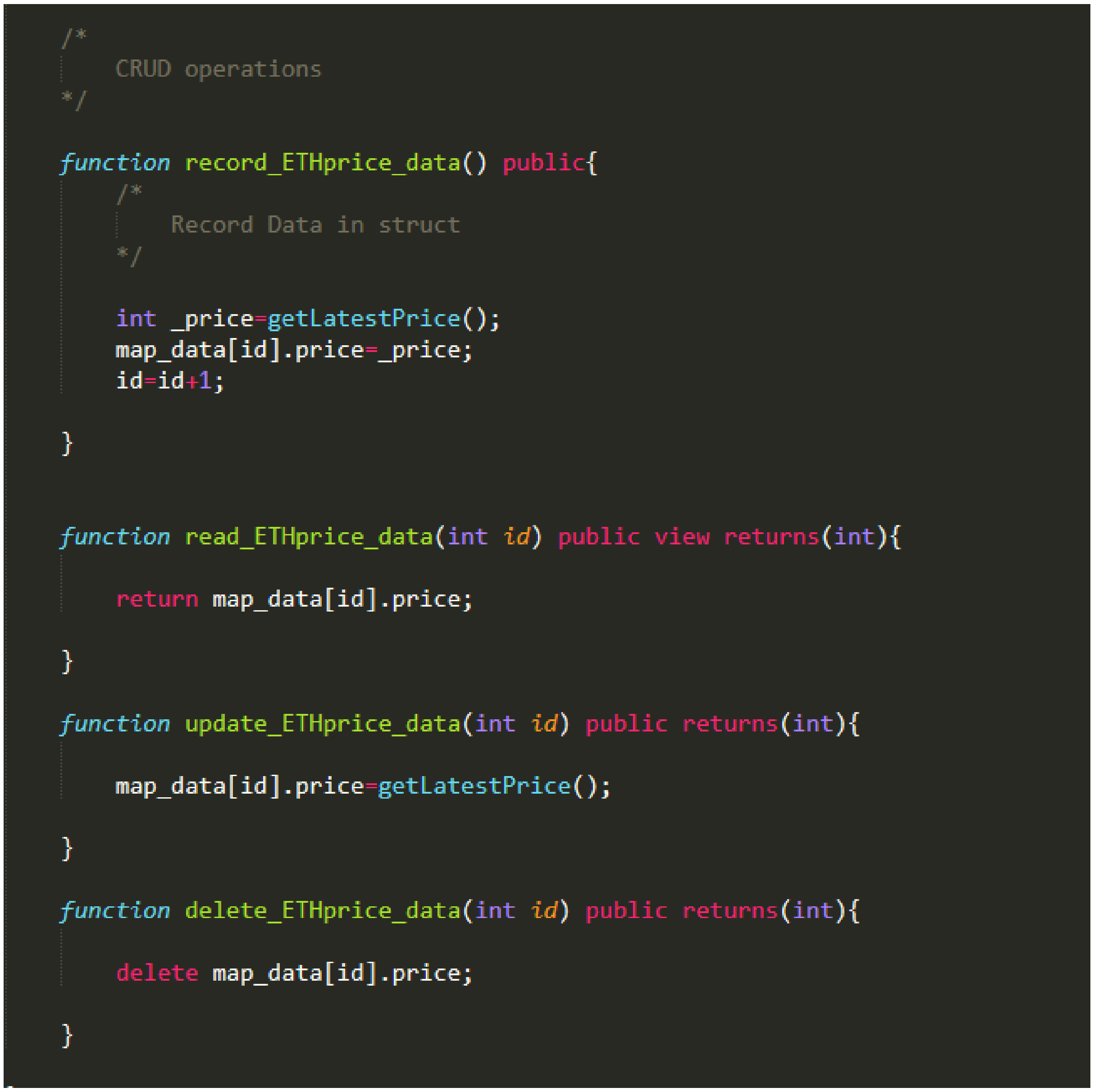
[Code Snippet]



```
/*
    Function to calculate mean of the prices stored
*/

function calculate_mean() public view returns(int){
    int sum=0;

    for(int i =0; i<=id; i++){

        sum=sum+map_data[i].price;

    }

    int avg=sum/(id);
    return avg;

}
```

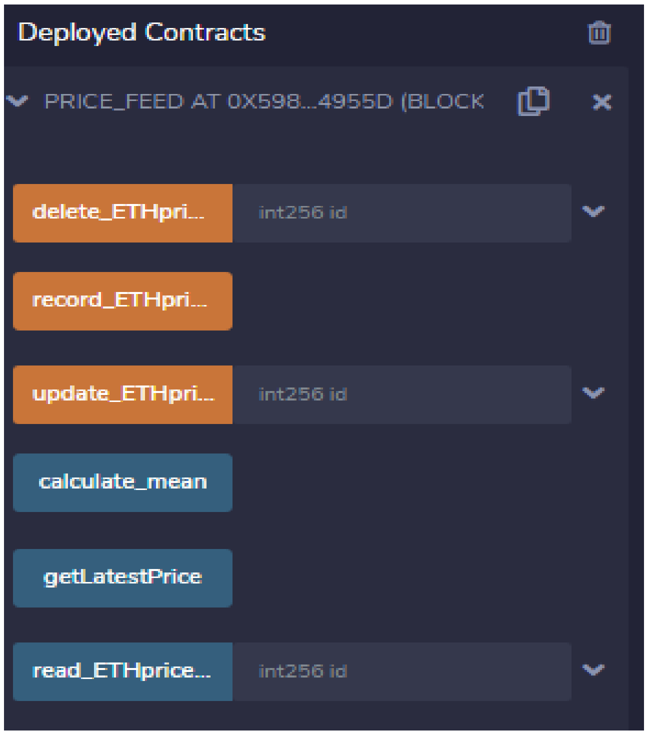
4. **Write CRUD operations for that struct**

[Code Snippet]

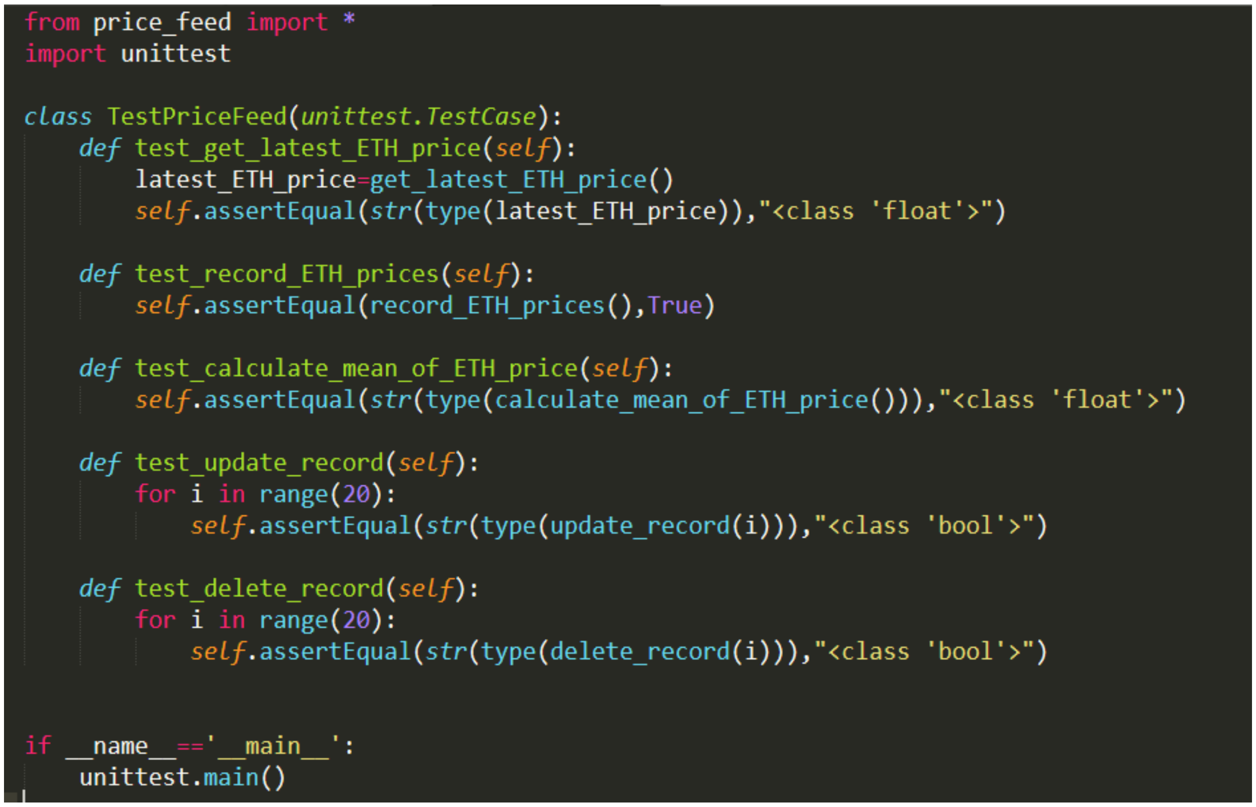


```
/*
    CRUD operations
*/

function record_ETHprice_data() public{
    /*
        Record Data in struct
    */

    int _price=getLatestPrice();
    map_data[id].price=_price;
    id=id+1;

}


function read_ETHprice_data(int id) public view returns(int){

    return map_data[id].price;

}

function update_ETHprice_data(int id) public returns(int){

    map_data[id].price=getLatestPrice();

}

function delete_ETHprice_data(int id) public returns(int){

    delete map_data[id].price;

}
```

5. Also write tests for these
    a. Deployed contracts



    b. Unit testing



```python
from price_feed import *
import unittest

class TestPriceFeed(unittest.TestCase):
    def test_get_latest_ETH_price(self):
        latest_ETH_price=get_latest_ETH_price()
        self.assertEqual(str(type(latest_ETH_price)),"<class 'float'>")

    def test_record_ETH_prices(self):
        self.assertEqual(record_ETH_prices(),True)

    def test_calculate_mean_of_ETH_price(self):
        self.assertEqual(str(type(calculate_mean_of_ETH_price())),"<class 'float'>")

    def test_update_record(self):
        for i in range(20):
            self.assertEqual(str(type(update_record(i))),"<class 'bool'>")

    def test_delete_record(self):
        for i in range(20):
            self.assertEqual(str(type(delete_record(i))),"<class 'bool'>")


if __name__=='__main__':
    unittest.main()
```