

Oracle Smart-Contract

Jinal Patel

Contents

Abstract:.....	2
Introduction:	2
Methodology:.....	3
Solidity Smart Contract:	3
Python Script:	5
Execution:	6
Conclusion:.....	8

Oracle Smart-Contract

Abstract:

This assignment involves the creation of an Oracle Smart Contract on the Ethereum blockchain.

The Oracle's purpose is to offer the current Ether price in US dollars. Solidity is used for the smart contract in this assignment, while Python is used for the off-chain script. On the Sepolia test network, the smart contract {MyOracle} is in place and has the ability to set, retrieve, and request changes to the Ether price.

The smart contract is interacting with the Python script `oracle.py`. It changes the price on the blockchain, waits for update requests from the contract, and retrieves the current Ether price from the CoinMarketCap API.

This assignment shows how to leverage Oracles in blockchain technology to provide smart contracts access to real-world data.

Introduction:

Blockchain technology has completely changed the way we manage data by offering a decentralized, transparent, and safe platform for exchanges. One of the most popular blockchain platforms, Ethereum, popularized the idea of smart contracts, which are self-executing contracts with the parameters of the agreement encoded directly into the code. These smart contracts do not require an intermediary since they automatically carry out transactions when specific criteria are satisfied.

Despite their enormous potential, smart contracts are severely limited in that they are unable to access any data that is not on the Ethereum blockchain, including data from the real world. Oracles can help with this. Third-party services called oracles supply external data to smart contracts. They act as a link between external systems and blockchains.

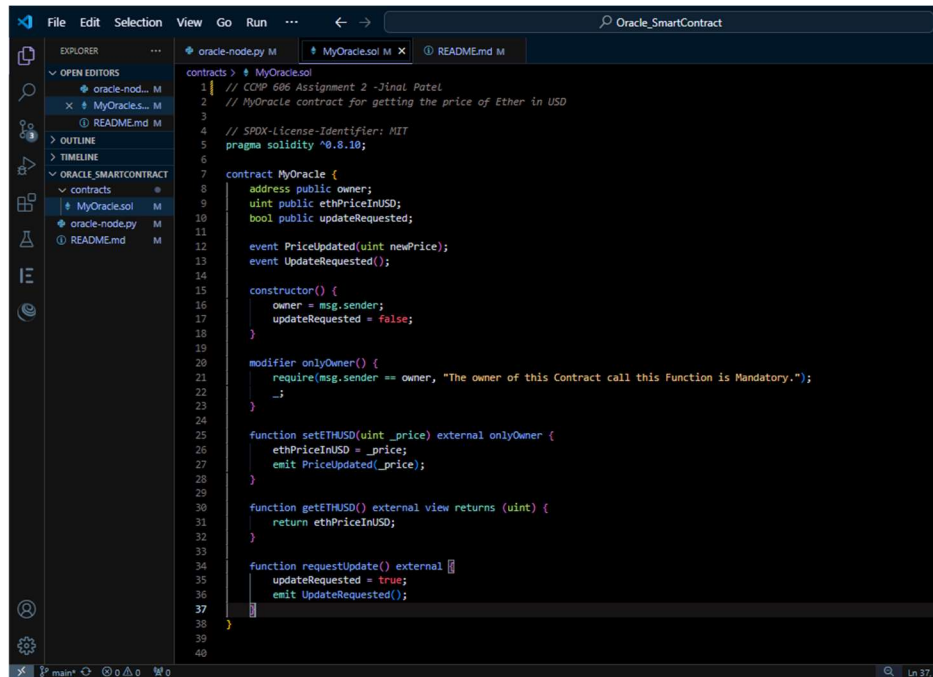
In this assignment, we have developed an Oracle Smart Contract on the Ethereum blockchain. The Oracle's purpose is to provide the current Ether (ETH) price in US dollars. An Ethereum blockchain-deployed Solidity smart contract and an off-chain Python script make up the Oracle's two primary parts. The MyOracle smart contract provides the ability to set, retrieve, and request updates to the Ether price. The smart contract is interacting with the Python script `oracle.py`. It changes the price on the blockchain, waits for update requests from the contract, and retrieves the current Ether price from the CoinMarketCap API.

This assignment will show how blockchain technology may be used to leverage Oracles to provide smart contracts access to real-world data, expanding their functionality and potential uses. It also offers a useful illustration of how to use Solidity and Python to construct and communicate with smart contracts on the Ethereum blockchain.

Methodology:

Solidity Smart Contract:

The smart contract MyOracle.sol, was solidity code written in Visual Studio. The contract was designed with functions to set, get, and request updates to the price of Ether. The contract was compiled and deployed on the Sepolia test network using the Remix IDE.

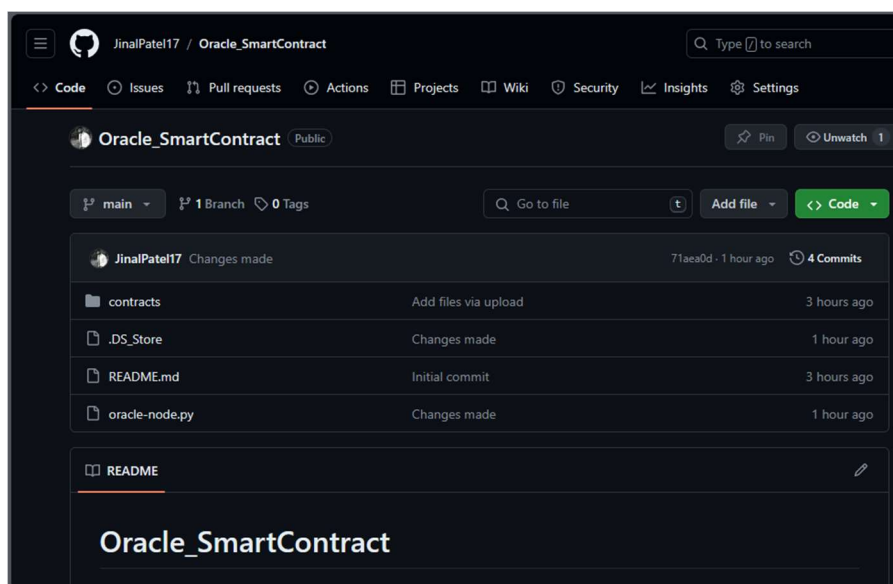


```

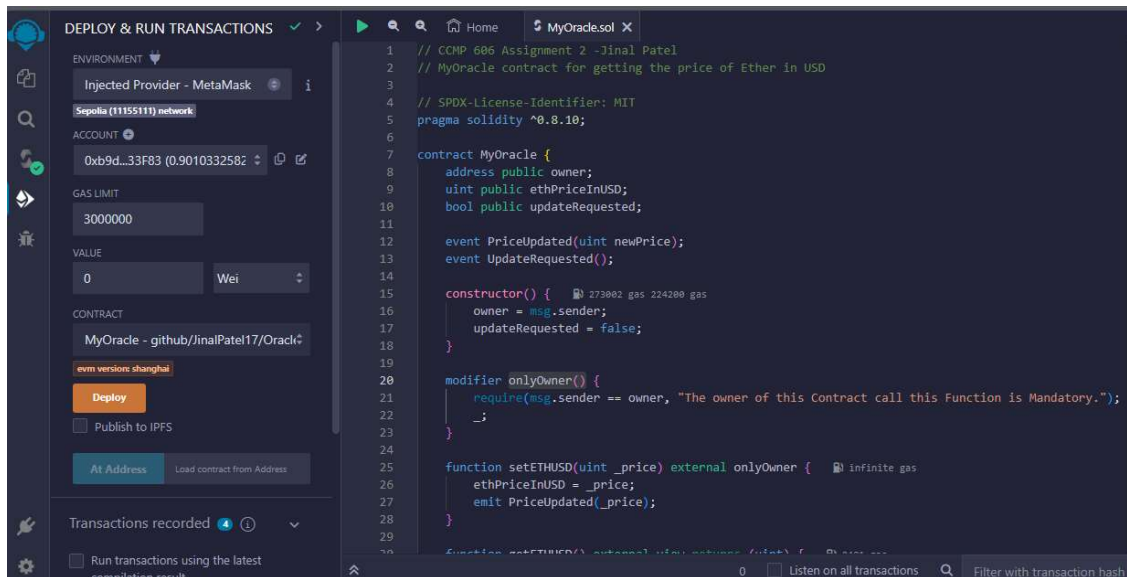
1 // COMP 686 Assignment 2 - Jinal Patel
2 // MyOracle contract for getting the price of Ether in USD
3
4 // SPDX-License-Identifier: MIT
5 pragma solidity ^0.8.10;
6
7 contract MyOracle {
8     address public owner;
9     uint public ethPriceInUSD;
10    bool public updateRequested;
11
12    event PriceUpdated(uint newPrice);
13    event UpdateRequested();
14
15    constructor() {
16        owner = msg.sender;
17        updateRequested = false;
18    }
19
20    modifier onlyOwner() {
21        require(msg.sender == owner, "The owner of this Contract call this Function is Mandatory.");
22        _;
23    }
24
25    function setETHUSD(uint _price) external onlyOwner {
26        ethPriceInUSD = _price;
27        emit PriceUpdated(_price);
28    }
29
30    function getETHUSD() external view returns (uint) {
31        return ethPriceInUSD;
32    }
33
34    function requestUpdate() external {
35        updateRequested = true;
36        emit UpdateRequested();
37    }
38
39
40

```

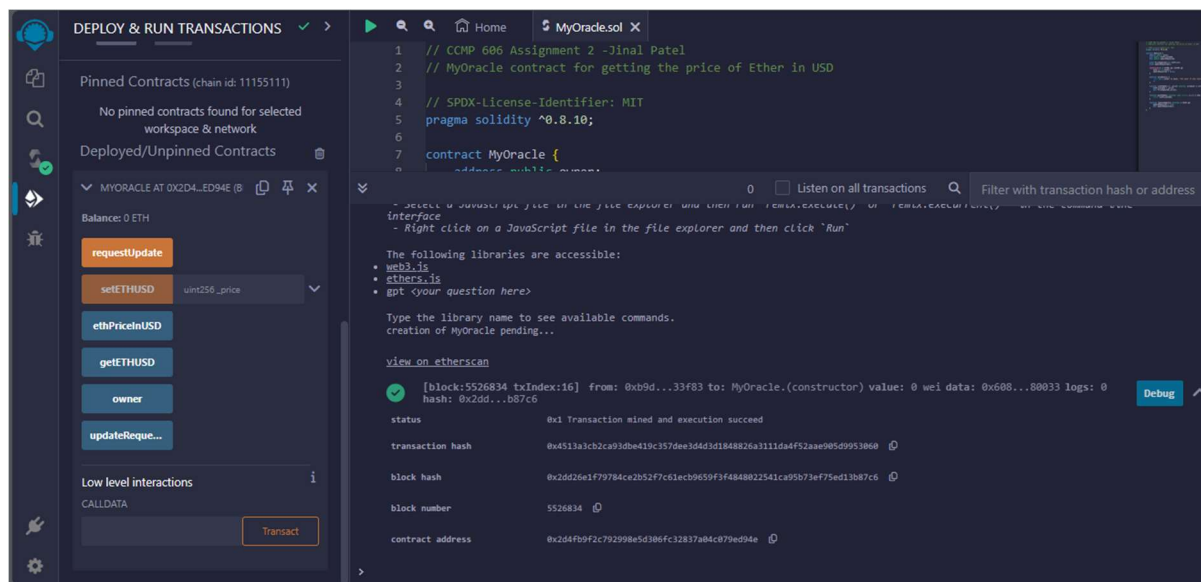
- Now this code is Complete in visual studio and uploaded to Github.



- We will create new environment and there we will put our GitHub repo contract location and compile.
- Now, in the deploy and run transactions section, we will change the environment and put injected provider as MetaMask and select our sepolia account on MetaMask. You will see as your account and in contract you will see your GitHub repo location and deploy it.

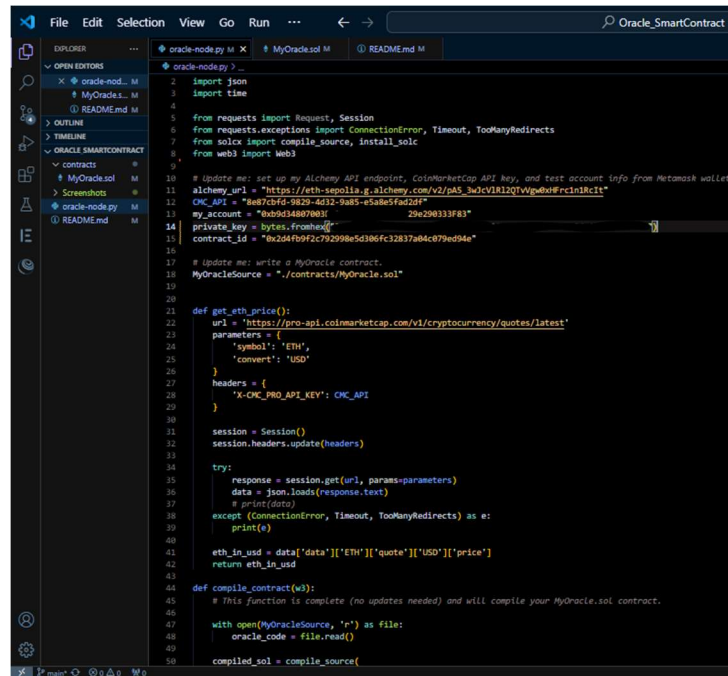


- After deployment, you can see in the terminal of remix there is our block and blockchain address with it.



Python Script:

- The Python script, named oracle.py, was written to interact with the MyOracle contract.
- The script uses the web3.py library to connect to the Ethereum blockchain via an Alchemy API endpoint.
- The script listens for UpdateRequested events emitted by the MyOracle contract. When such an event is detected, the script fetches the current price of Ether from the CoinMarketCap API and updates the price in the MyOracle contract by calling the setETHUSD function.
- The Python script uses the requests library to make HTTP requests to the CoinMarketCap API. The solcx library is used to compile the Solidity contract, and the eth_account library is used to handle Ethereum accounts and private keys.
- Now here in oracle-node.oy file we will update alchemy_url, CPC_API, my_account, private_key and contract_id.
- Contract_id retrieve from remix terminal deployed contract address.



```

1  import json
2  import time
3
4  from requests import Request, Session
5  from requests.exceptions import ConnectionError, Timeout, TooManyRedirects
6  from solcx import compile_source, install_solc
7  from web3 import Web3
8
9  # Update me: set up my Alchemy API endpoint, CoinMarketCap API key, and test account info from Metamask wallet.
10 alchemy_url = "https://eth-sepolia.g.alchemy.com/v2/pA5_3u3cV1R12QTVpdxHfrcinRcIt"
11 CM_API = "8e87cbfd-9829-4d32-9a85-e5a8e5fad2d1"
12 my_account = "0x0d14807b01129e298133f813"
13 private_key = bytes.fromhex("29e298133f813")
14 contract_id = "0x2d4f09f2c792998e5d306fc32837a84c879ed94e"
15
16 # update me: write a MyOracle contract.
17 MyOracleSource = "../contracts/MyOracle.sol"
18
19
20
21 def get_eth_price():
22     url = "https://pro-api.coinmarketcap.com/v1/cryptocurrency/quotes/latest"
23     parameters = {
24         "symbol": "ETH",
25         "convert": "USD"
26     }
27     headers = {
28         "X-CMC_PRO_API_KEY": CM_API
29     }
30
31     session = Session()
32     session.headers.update(headers)
33
34     try:
35         response = session.get(url, params=parameters)
36         data = json.loads(response.text)
37         # print(data)
38     except (ConnectionError, Timeout, TooManyRedirects) as e:
39         print(e)
40
41     eth_in_usd = data["data"][0]["ETH"]["quote"]["USD"]["price"]
42     return eth_in_usd
43
44 def compile_contract(sol):
45     # This function is complete (no updates needed) and will compile your MyOracle.sol contract.
46
47     with open(MyOracleSource, "r") as file:
48         oracle_code = file.read()
49
50     compiled_sol = compile_source(

```

- Note here that private_key and account details are hidden for privacy purpose.

- We can verify our blockchain,

[This is a Sepolia Testnet transaction only]

Transaction Hash: 0xc32fddba9f9585b3cda4bef880462493f7faddbc39f5a940dbc1dba2711a3c03

Status: Success

Block: 5526958 5 Block Confirmations

Timestamp: 53 secs ago (Mar-20-2024 10:05:12 PM +UTC)

Transaction Action: Call Request Update Function by 0xb9d34807...290333F83 on 0x2D4fb9F2...C079eD94e

From: 0xb9d34807003F4B878C61d7D1474229e290333F83

To: 0x2D4fb9F2c792998e5D306Fc32837a04C079eD94e

Value: 0 ETH (\$0.00)

Transaction Fee: 0.000036474000316108 ETH (\$0.00)

Gas Price: 1.500000013 Gwei (0.000000001500000013 ETH)

- We can verify on coinmarketcap website that our api working

Developers

OVERVIEW

DEX OVERVIEW

PLAN & BILLING

NOTIFICATIONS

*****_*****_*****
*****_*****_*****

Regenerate Key

53 Credits Today

0 Credits Yesterday

53/10,000 Credits This Month

Last Updated: Today at 4:07 PM

API Request Log

Last 100 API Requests	Timestamp	Credit Count
#1 HTTP 200 - IP 65.87.241.65 /v1/cryptocurrency/quotes/latest?symbol=ETH&convert=USD	March 20th, 2024, 4:05:15 PM 2 minutes ago	1
#2 HTTP 200 - IP 65.87.241.65 /v1/cryptocurrency/quotes/latest?symbol=ETH&convert=USD	March 20th, 2024, 3:57:01 PM 10 minutes ago	1
#3 HTTP 200 - IP 65.87.241.65 /v1/cryptocurrency/quotes/latest?symbol=ETH&convert=USD	March 20th, 2024, 3:53:14 PM 14 minutes ago	1

- Also, we can verify on Alchemy Dashboard,

The screenshot displays the Alchemy Dashboard interface. On the left is a dark sidebar with navigation links: Home, Apps (selected), Analytics, Logs, Mempool, Tools, Webhooks, Gas Manager, and Accounts Manager. At the bottom of the sidebar are links for Feedback and Discord Support. The main content area is titled 'Apps' and features a search bar and a 'Create new app' button. Below this is a table listing applications:

App Name	Network	Requests (24h)	Failed requests (24h)	Created on	Actions
Oracle smart contract	Ethereum Sepolia	325	0	3/20/2024	API Key, App Details

Additional UI elements include a 'Get \$100+' button, a notification bell, a user profile 'JP', and a 'Get Support' button at the bottom right.

- We can see request that has been made from our py to smart contract.

Conclusion:

This assignment demonstrated the implementation of an Oracle on the Ethereum blockchain. Oracle was successfully deployed and functioned, providing the current price of Ether in USD. The assignment demonstrated the compatibility of various blockchain ecosystem technologies by using Python for the off-chain script and Solidity for the smart contract.

With the help of the Oracle, which updated the Ethereum blockchain with the current price of ether obtained from the CoinMarketCap API, the Ethereum blockchain and real-world data were successfully connected. Through the ability to interface with external data and APIs, this study demonstrated the critical role that Oracles play in expanding the possibilities of smart contracts.

Despite this smart contract deployed successfully, it also highlighted the difficulties and complications associated with using blockchain technology and Oracles. In future, this work could be explored more in more advanced area, such as adding more data points, improving the update mechanism, or implementing access control and security measures.

In conclusion, this assignment offered insightful information on how Oracles function and possible uses in the blockchain industry. It performs as a platform for more research and development in the areas of smart contracts and blockchain technology.