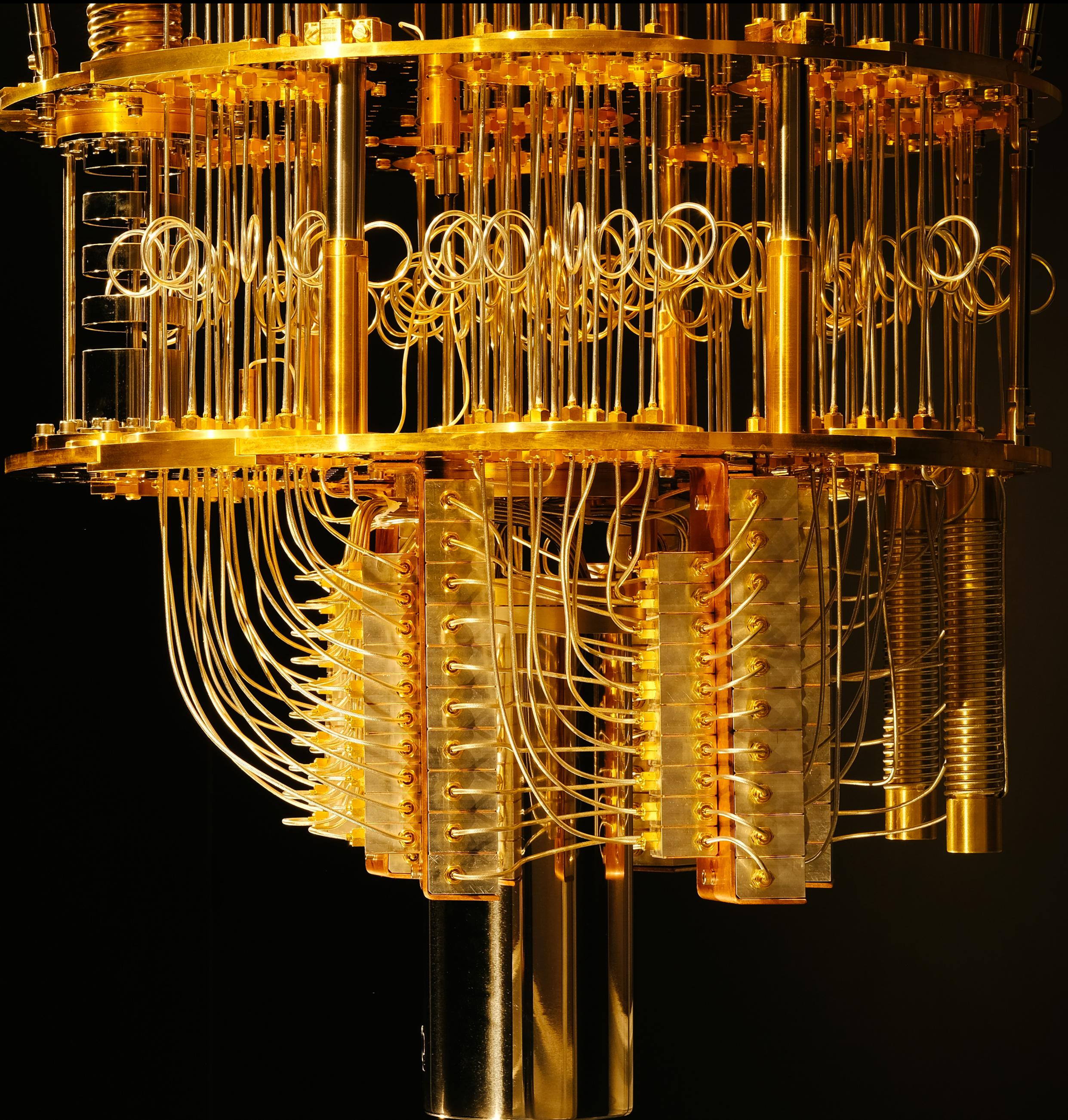


Quantum Oracles

Quantum computing as a service on-chain



Yash Goyal, Jan Ole Ernst, Jessica Pointing

Why?



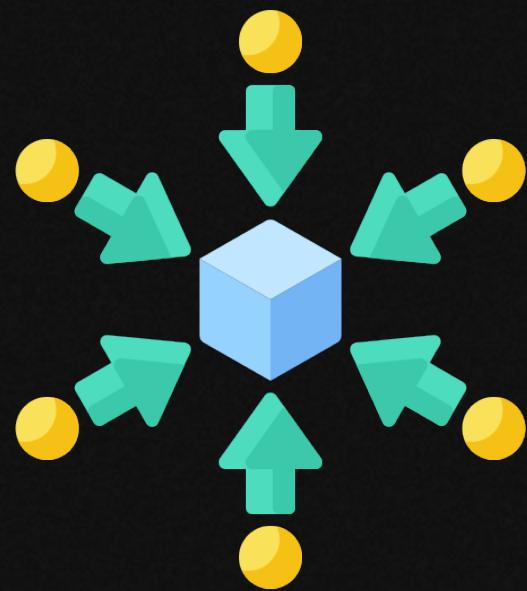
Quantum computing could solve the world's hardest problems



Quantum computing hardware & software development is accelerating



Major companies and governments are getting involved in quantum computing



**Access is centralised and not
democratised**



**The same computations are
rerun from scratch, which is
inefficient**



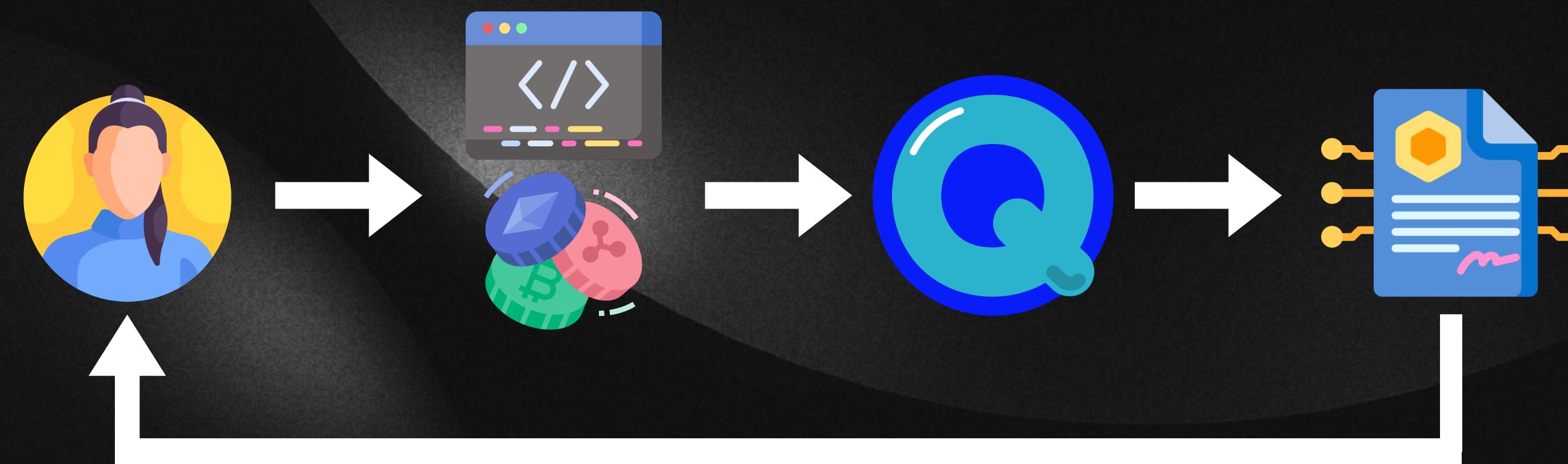
**No ability to execute logic
based on others runs**

Problem

Our Solution

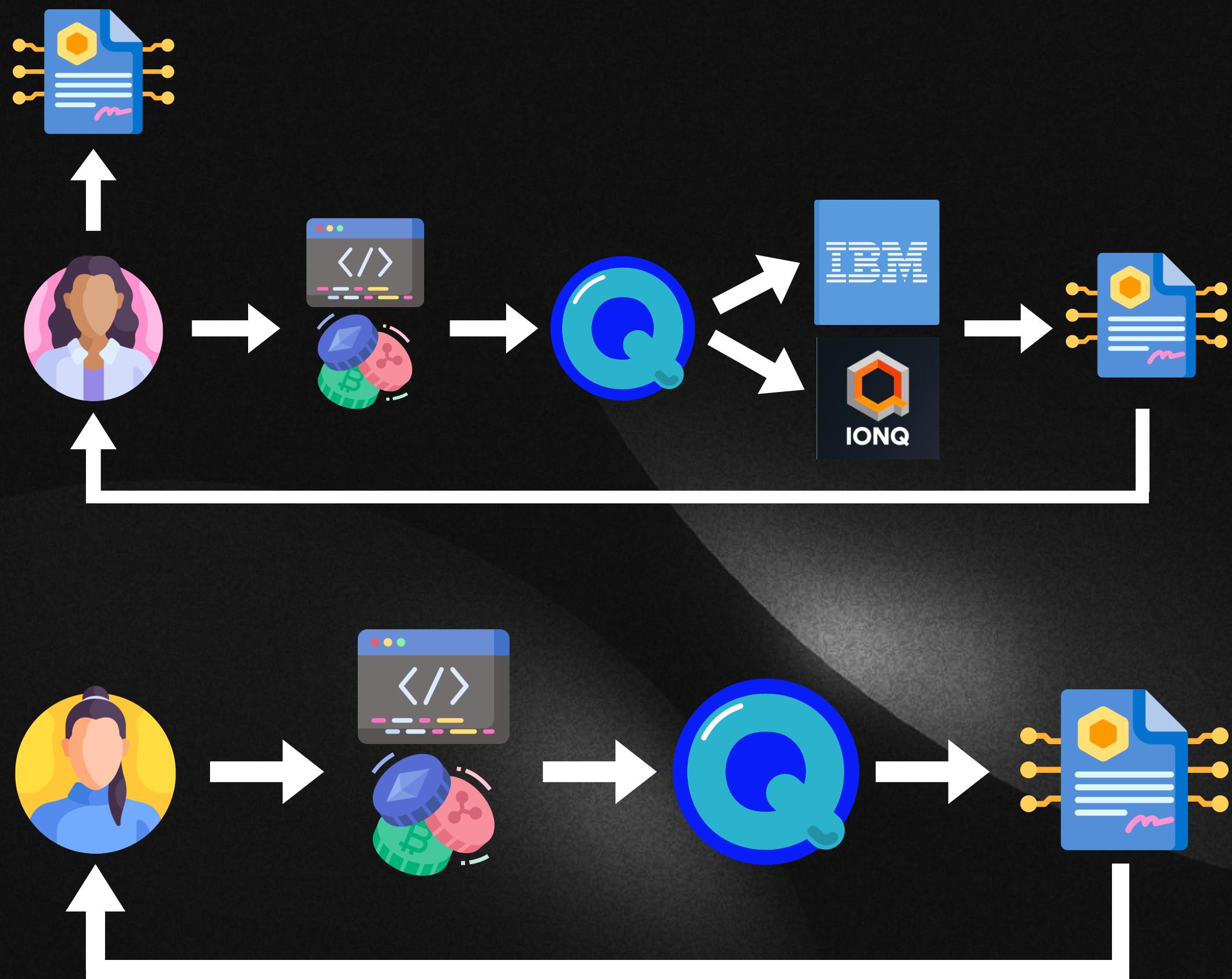


Another smart contract can execute logic based on the results stored on-chain



If another user submits the same quantum program as someone else, they can receive the results immediately and we can bypass re-running the same computation on the quantum service providers

Benefits



- 1 Hashing the circuit and results on-chain**
Efficient retrieval of results and no duplicate computation;
Privacy preserving
- 2 Our management of IBM account**
No need for the user to spend time and energy setting up IBM account and agreement to run on their quantum computer
- 3 Storing results on chain**
User can execute logic in another smart contract based on the quantum computation results
- 4 Payment in crypto**
Seamless payments
- 5 Flexible backend options**
User can choose any backend to run on (simulator, hardware, IBM, Google, etc...).
Solution is portable to other specialised devices (e.g. GPUs, classical supercomputers...)
- 6 Decentralisation**
The same quantum program is sent to multiple quantum computers/simulators and the results are aggregated and stored on-chain

Tech Stack



Fast and cheap blockchain, where we deployed our smart contract, for storing quantum computation results.



Used for building the front-end and integrating the front-end with the smart contract.



Google Cloud



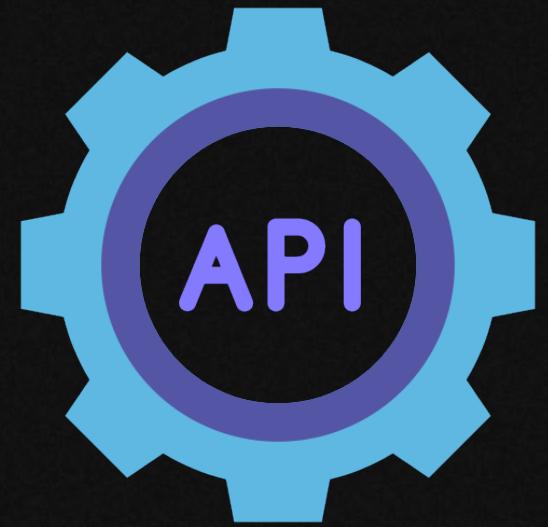
Deployed our quantum API and API3 Airnode on Google Cloud.

Deployed an Airnode to call our quantum API and used the dAPI price feed for calculating the cost in cryptocurrency for running the quantum computation.



User Interface:

- User uploads quantum program
 - Price calculation
 - Sign transaction
 - Display results



Quantum computing API

- Send a job to a quantum computer (can choose backend)
- Retrieve job result
- Draw diagram of quantum program
- Deployed at URL



Smart Contract

- Send quantum program via our oracles to quantum service
- Store hash of circuit and quantum program results

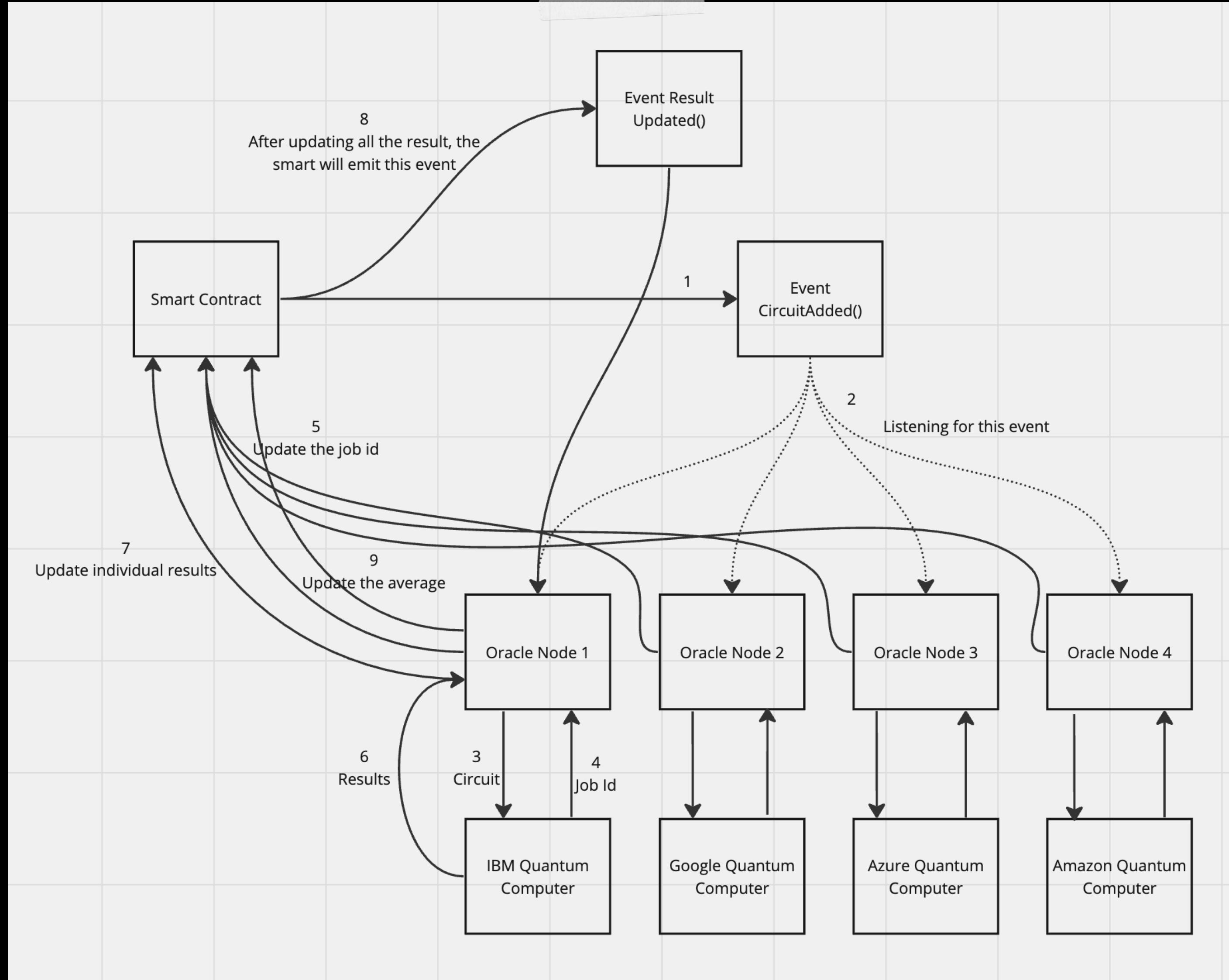


Oracles

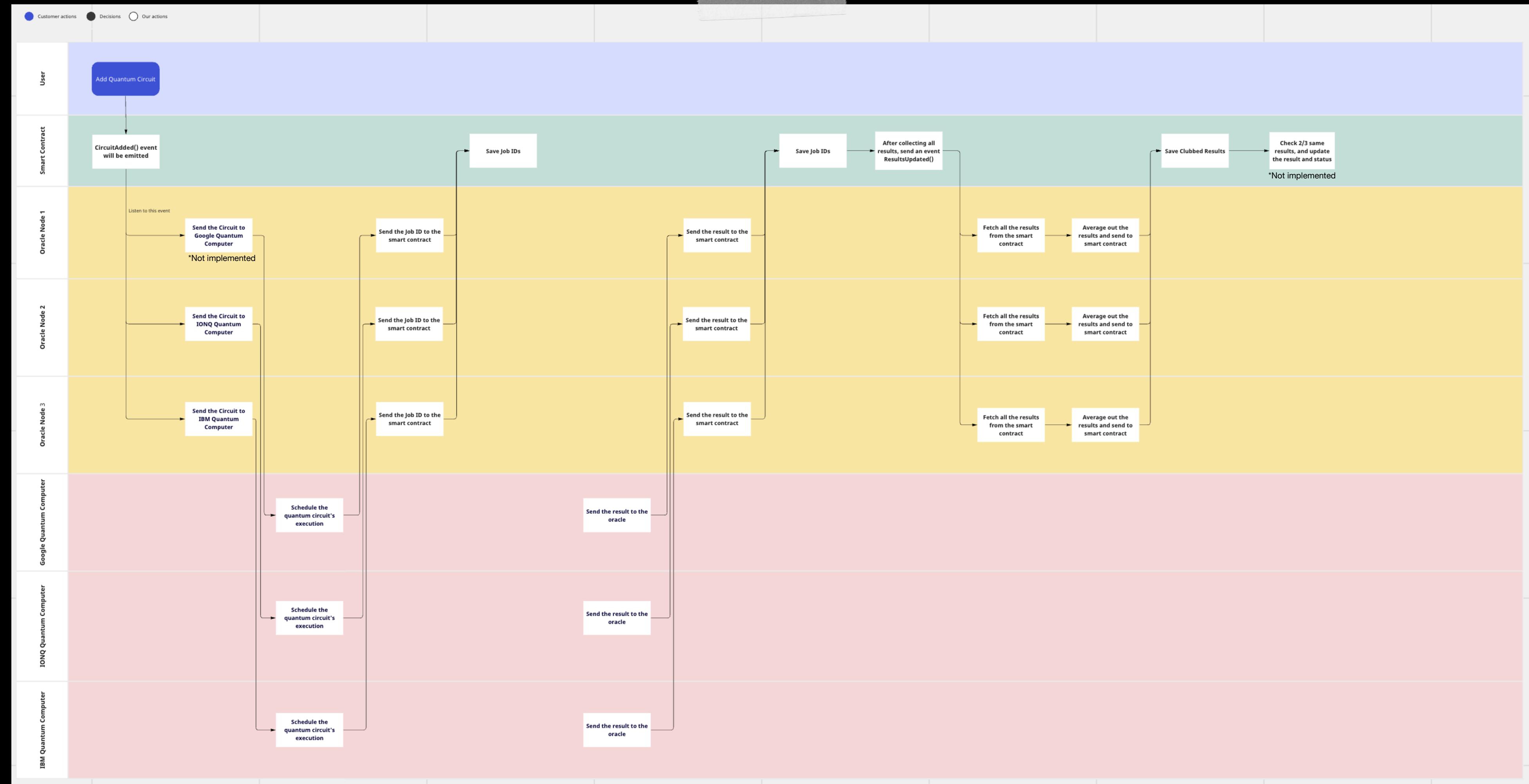
- Send quantum program to quantum computing service
- Retrieve results from quantum computing services

Components

Architecture



Flow Diagram



<https://miro.com/app/board/uXjVNVN3tMU=/>

Upload



Welcome to
QOracle

Upload your Quantum Circuit
[Upload tab.](#)

Find the results of your Quantum
Computations [here](#)

Fork me · Built with ❤ at [BuildGuild](#) · [Support](#)

9.9142 CFLR
Coston

0X11...FF4C

Results



QRoracle
Decentralised Oracle for your Quantum Circuits

Home Upload Quantum Circuit QR oracle Computations Debug Contracts

9.9142 CFLR Coston 0X11...FF4C

Upload your Quantum Circuit

File input *

Upload your QASM file

Set your Circuit

Display Circuit ↗

Your Circuit

```
OPENQASM 2.0;
include "qelib1.inc";
qreg q[3];
creg meas[3];
h q[2];
x q[0];
y q[2];
h q[2];
h q[2];
cx q[0],q[1];
cx q[0],q[2];
barrier q[0],q[1],q[2];
measure q[0] -> meas[0];
measure q[1] -> meas[1];
measure q[2] -> meas[2];
```

Result: {"110":2502,"111":2522}

Fork me · Built with ❤ at [Build Guild](#) · Support

🌓

Smart Contract Functions



QOracle
Decentralised Oracle for your Quantum Circuits

Home Upload Quantum Circuit QOracle Computations Debug Contracts

9.9142 CFLR Coston 0X11...FF4C

QuantumOracleV1
0x7A1...4d69 Balance: 0.0001 CFLR Network: Coston

owner 0xC3A...0cA0 totalOracleAddresses 2

Read

calculateCost string circuitQASM READ

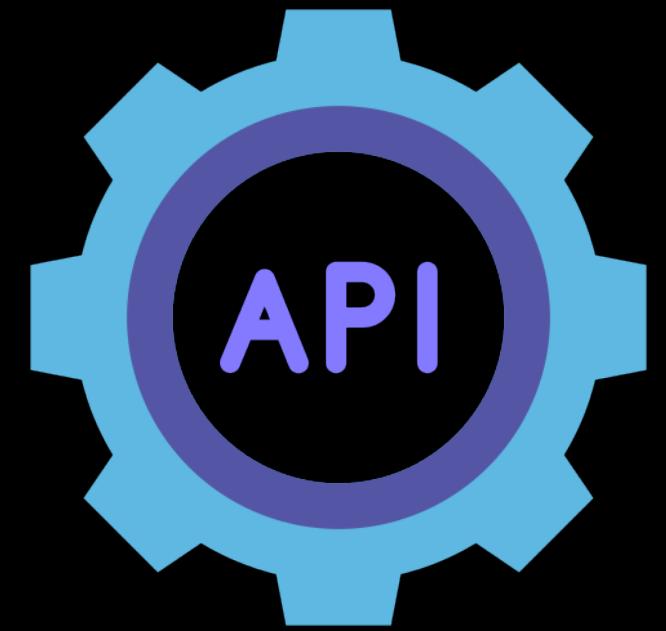
circuits bytes32 # READ

hasSubmittedResponse bytes32 # uint8 * address READ

isOracleAddress address

A toggle switch is located at the bottom right.

The screenshot shows the QOracle web interface. At the top, there's a navigation bar with links for Home, Upload Quantum Circuit, QOracle Computations, and Debug Contracts. On the right, it shows the balance (9.9142 CFLR) and network (Coston), along with a wallet address (0X11...FF4C). The main area displays information about a smart contract named "QuantumOracleV1", including its address (0x7A1...4d69), balance (0.0001 CFLR), and network (Coston). Below this, there are two sections: "owner" (address 0xC3A...0cA0) and "totalOracleAddresses" (value 2). The central part of the interface is a list of functions with their parameters and "READ" buttons. The functions listed are calculateCost, circuits, hasSubmittedResponse, and isOracleAddress. The "calculateCost" function takes a string parameter "circuitQASM". The "circuits" function returns a bytes32 value. The "hasSubmittedResponse" function takes bytes32, uint8, and address parameters. The "isOracleAddress" function takes an address parameter. A toggle switch is located at the bottom right of the interface.



Add circuit
Draw circuit
Retrieve results

FastAPI 0.1.0 OAS 3.1 /openapi.json

default

POST /circuit Create Circuit

POST /draw Draw Circuit

POST /result Get Result

Schemas

Body_create_circuit_circuit_post > Expand all object

Body_get_result_result_post > Expand all object

HTTPValidationError > Expand all object

ValidationError > Expand all object

How it began...

The idea was born at the Oxford Blockchain Hack a week ago. During that hackathon, we implemented:

- A simple frontend (not using Scaffold ETH)
- A quantum API with one quantum service provider (IBM)
- A smart contract with Chainlink functions deployed on Polygon
- We built these components individually - they were not integrated together

For this hackathon, we've created a new project based on the same concept. This new project includes new code implementation, new sponsor integration, and improvements to the concept.

For this hackathon, we implemented:

- A new frontend using Scaffold ETH
- Other service providers to the quantum API in order to add decentralisation to our app
- Our own oracles to call the quantum API
- Deployment of the smart contract on Flare
- Deployment of the quantum API and API3 Airnode on Google Cloud
- Deployment of an API3 Airnode and integrating its dAPI price feed in our computational cost calculations

Team



Jan Ole Ernst
PhD in Physics

Specialising in experimental
quantum control
Wealth of web3 experience



Yash Goyal
Digital Nomad

Specialised in blockchain
development
and backend development



Jessica Pointing
PhD in Physics

Specialising in quantum neural
networks
Web3 security researcher

+ Brandon Severin (emotional and slide show support)