

# TERRARIUM IOT- HYPERLEDGER PROJECT

- Fawaz Malik

#101461582

- Namit Mani

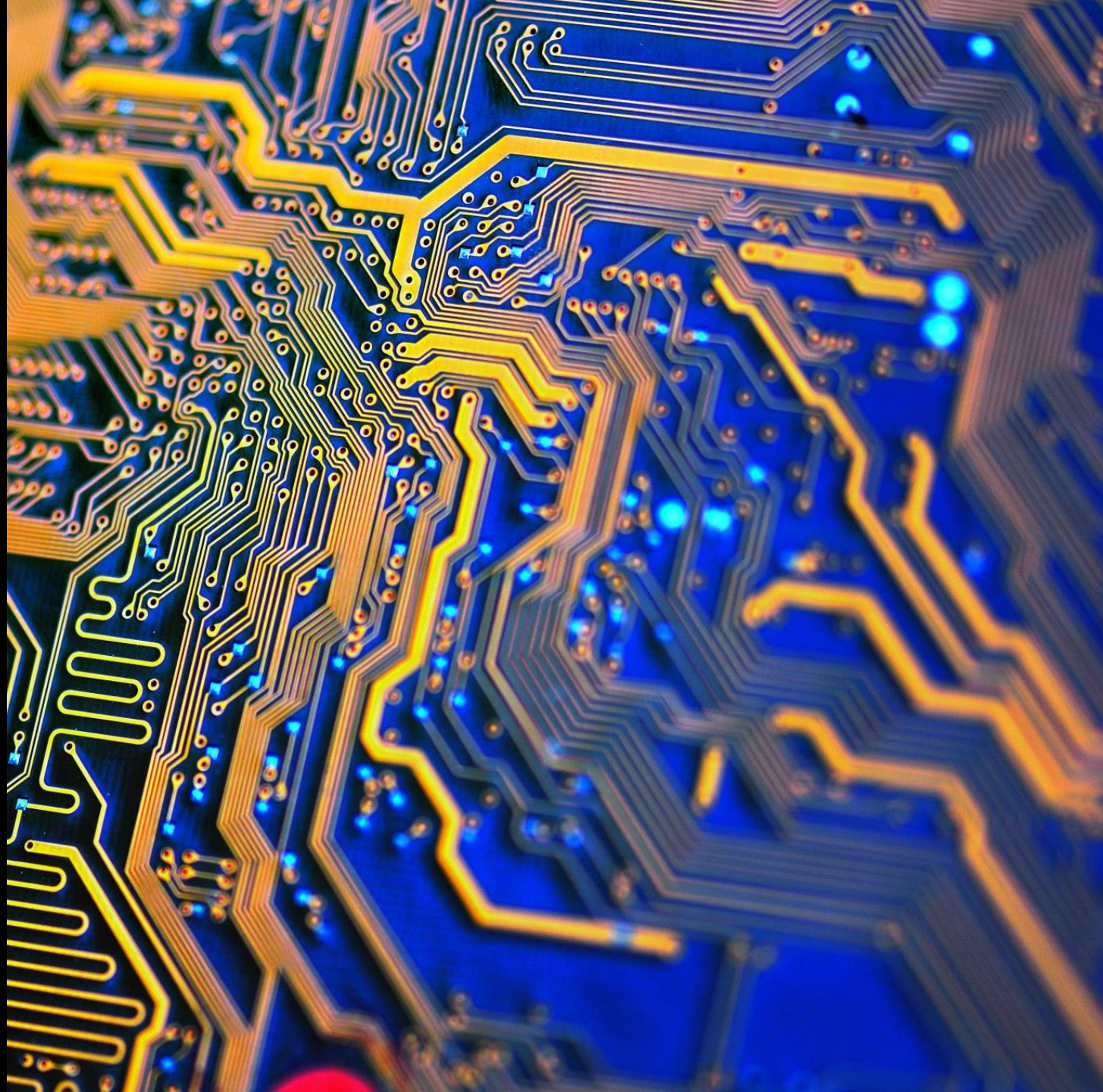
#101383808

- Aniket Srivastava #101469899

- BADARUDDIN KHUHRO

#101467663

- HEMANTH KUMAR POTHURI #101464127





# WHY BLOCKCHAIN

- **Transparency**
- **Security**
- **Efficiency**
- **Trust**
- **No Tokens needed**
- **Scalability Benefits**
- **Trackability**



# OUR CONSENSUS MECHANISM

- Proof of Elapsed Time
  - Cheap (economical)
  - Token-less (helps in live implementation)
  - Randomized (safer)







# USERS INVOLVED

- Potential Users of the Blockchain Network:
  - Shop-owners
  - Pet-Owners
  - Small IoT Device Manufacturer

# USE CASE #1

- Purchase Scenario:
  - customer buys pet and terrarium tank from shopowner. Transaction stored on blockchain Hyperledger fabric with button click on front end application
  - Shop owner sells terrarium tank and a pet with IoT device to monitor health and safety status to consumer
  - Consumer can test the IoT at the shop using Start tank button on front end

# BENEFITS

- Two-fold:
  - 1 For New Customers :- Ease of Ownership, ability to choose the level of interaction in the pet's life and allows for a better "new parent" experience.
  - 2 For Old Customers :- The ability to have an even more immersive ownership experience. An easier ownership experience as well as e-managed abilities that





# ERN STACK

- Dev Stack- Express.js, React.js, Node.js
- Front end user interface consists of Nodejs and react components
- Back end consists of Express talking back door with our Hyperledger fabric chaincode



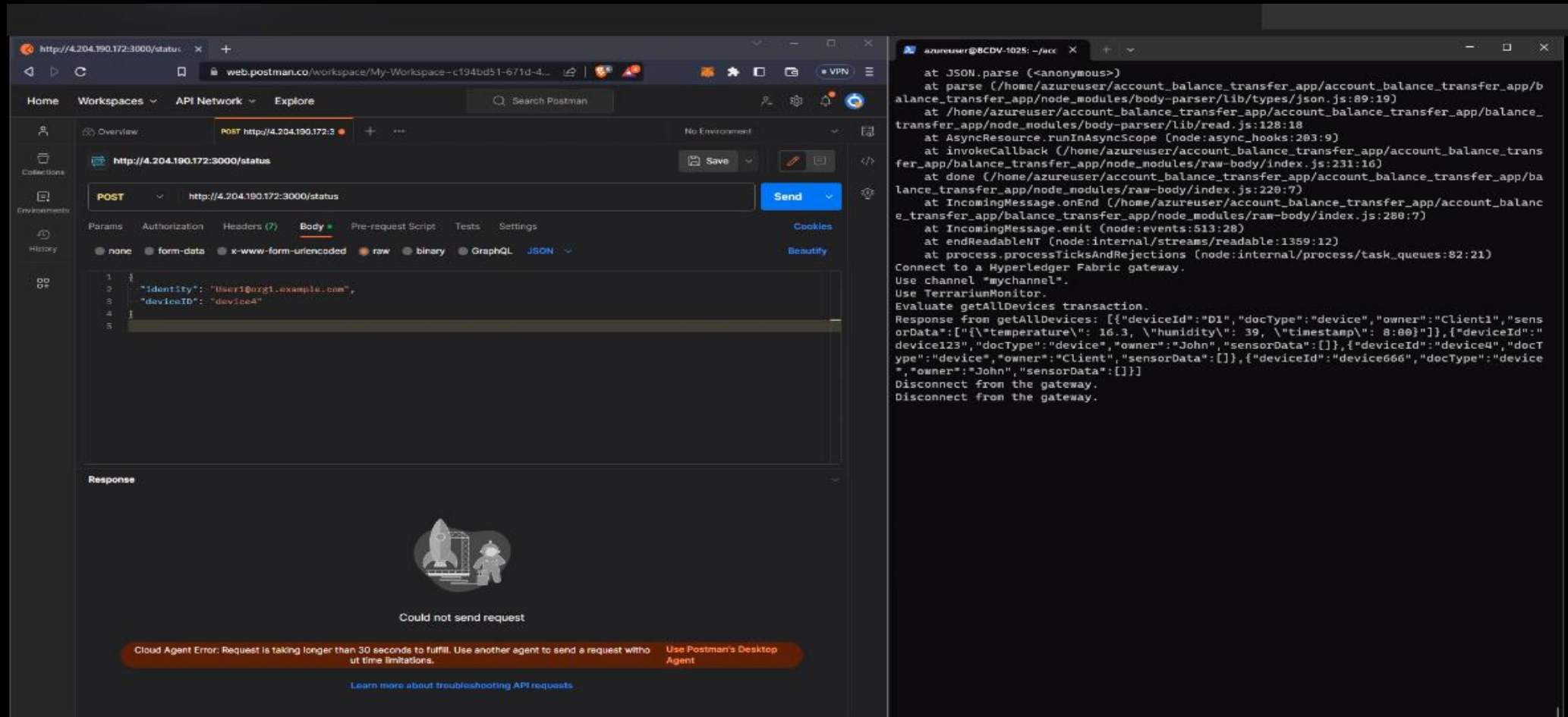
# OUR SMART CONTRACT

```
1  /*
2   * SPDX-License-Identifier: Apache-2.0
3   */
4
5  'use strict';
6
7  const { Contract } = require('fabric-contract-api');
8
9  class PetTerrariumContract extends Contract {
10     async initLedger(ctx) {
11         // Initialize any initial data or setup here
12     }
13
14     async buyPet(ctx, petId, ownerId, petName, purchaseDate) {
15         const pet = {
16             ownerId,
17             petName,
18             purchaseDate,
19             status: 'Purchased',
20             tankId: '',
21             isTankStarted: false,
22         };
23
24         await ctx.stub.putState(petId, Buffer.from(JSON.stringify(pet)));
25     }
26
27     async buyTank(ctx, petId, tankId, purchaseDate) {
28         const petBytes = await ctx.stub.getState(petId);
29         if (!petBytes || petBytes.length === 0) {
30             throw new Error(`Pet with ID ${petId} does not exist.`);
31         }
32
33         const pet = JSON.parse(petBytes.toString());
34         if (pet.status !== 'Purchased') {
35             throw new Error(`Pet with ID ${petId} is not in a valid state for buying a tank.`);
36         }
37     }
38 }
```

```
37
38     pet.tankId = tankId;
39     pet.purchaseDate = purchaseDate;
40     await ctx.stub.putState(petId, Buffer.from(JSON.stringify(pet)));
41 }
42
43 async startTank(ctx, petId) {
44     const petBytes = await ctx.stub.getState(petId);
45     if (!petBytes || petBytes.length === 0) {
46         throw new Error(`Pet with ID ${petId} does not exist.`);
47     }
48
49     const pet = JSON.parse(petBytes.toString());
50     if (pet.status !== 'Purchased' || !pet.tankId) {
51         throw new Error(`Pet with ID ${petId} is not in a valid state for starting the tank.`);
52     }
53
54     pet.isTankStarted = true;
55     await ctx.stub.putState(petId, Buffer.from(JSON.stringify(pet)));
56 }
57
58 async getPet(ctx, petId) {
59     const petBytes = await ctx.stub.getState(petId);
60     if (!petBytes || petBytes.length === 0) {
61         throw new Error(`Pet with ID ${petId} does not exist.`);
62     }
63
64     return petBytes.toString();
65 }
66 }
67
68 module.exports = PetTerrariumContract;
69
```



# CHAINCODE INSTALLATION & DEPLOYMENT



The image displays a development environment with two windows. The left window is the Postman API client, showing a POST request to `http://4.204.190.172:3000/status` with a JSON body. The right window is a terminal showing the Node.js error logs for the failed request.

**Postman Request Details:**

- Method: POST
- URL: `http://4.204.190.172:3000/status`
- Body (JSON):

```
{  "Identity": "User1@org1.example.com",  "deviceId": "device4"}
```

**Terminal Output (Node.js Error Logs):**

```
at JSON.parse (<anonymous>)
at parse (/home/azureuser/account_balance_transfer_app/account_balance_transfer_app/b
alance_transfer_app/node_modules/body-parser/lib/types/json.js:89:19)
at /home/azureuser/account_balance_transfer_app/account_balance_transfer_app/balance_
transfer_app/node_modules/body-parser/lib/read.js:128:18
at AsyncResource.runInAsyncScope (node:async_hooks:203:9)
at invokeCallback (/home/azureuser/account_balance_transfer_app/account_balance_trans
fer_app/balance_transfer_app/node_modules/raw-body/index.js:231:16)
at done (/home/azureuser/account_balance_transfer_app/account_balance_transfer_app/ba
lance_transfer_app/node_modules/raw-body/index.js:220:7)
at IncomingMessage.onEnd (/home/azureuser/account_balance_transfer_app/account_balanc
e_transfer_app/balance_transfer_app/node_modules/raw-body/index.js:280:7)
at IncomingMessage.emit (node:events:513:28)
at endReadableNT (node:internal/streams/readable:1359:12)
at process.processTicksAndRejections (node:internal/process/task_queues:82:21)
Connect to a Hyperledger Fabric gateway.
Use channel "mychannel".
Use TerrariumMonitor.
Evaluate getAllDevices transaction.
Response from getAllDevices: [{"deviceId":"D1","docType":"device","owner":"Client1","sens
orData":[{"temperature": 16.3, "humidity": 39, "timestamp": "8:00"}]},{"deviceId":"
device123","docType":"device","owner":"John","sensorData":[]},{"deviceId":"device4","docT
ype":"device","owner":"Client","sensorData":[]},{"deviceId":"device666","docType":"device
","owner":"John","sensorData":[]}]
Disconnect from the gateway.
Disconnect from the gateway.
```

**Postman Response:**

Could not send request

Cloud Agent Error: Request is taking longer than 30 seconds to fulfill. Use another agent to send a request without time limitations. [Learn more about troubleshooting API requests.](#)

Use Postman's Desktop Agent

# LOGICAL ILLUSTRATION OF OUR CONTRACT

