

VaxDist: A Permissioned Blockchain for the Supply and Distribution of Vaccines

Team 23

Shivam Gandhi
shivam.gandhi@kcl.ac.uk

Thomas Herring
thomas.herring@kcl.ac.uk

Reece Roberts
reece.roberts@kcl.ac.uk

Loknath Tharun
tharun.loknath@kcl.ac.uk

Mohit Mukesh Ahuja
mohit_mukesh.ahuja@kcl.ac.uk

Shreyas Solanki
shreyas.solanki@kcl.ac.uk

April 2021



Contents

1	Executive Summary	1
2	Goals	2
3	Methodology	3
3.1	Design	3
3.1.1	Use Case Diagram	3
3.1.2	Activity Diagrams	3
3.2	Supporting Technologies	4
3.3	Storing the Blockchain	4
3.4	Block Generation	4
4	Results	5
4.1	Goal 1	5
4.2	Goal 2	5
4.3	Goal 3	5
4.4	Goal 4	6
4.5	Goal 5	6
4.6	Goal 6	6
5	Evaluation	7
5.1	Strengths of the project	7
5.2	Weaknesses of the project	7
6	Future Lessons	8
7	Appendices	10
7.1	GUI	10
7.2	CLI	12
7.3	Code	13
7.3.1	node.py	13
7.3.2	blockchain.py	17
7.3.3	block.py	21
7.3.4	communication.py	22
7.3.5	cli.py	26
7.3.6	api.py	30
7.3.7	style.css	33
7.3.8	add-shipment.js	34
7.3.9	api-port.js	35
7.3.10	get-block.js	35
7.3.11	get-pending-transactions.js	37
7.3.12	get-status.js	38
7.3.13	request-shipment.js	39

7.3.14	add-shipment.html	40
7.3.15	block-explorer.html	43
7.3.16	index.html	45
7.3.17	pending-transactions.html	47
7.3.18	add-shipment.html	50

1 Executive Summary

Since the release of Nakamoto's Bitcoin in 2009, the development of blockchain and related technologies has accelerated at a staggering pace over the past 10 years. As with all new technologies, thought immediately shifts to alternative implementations and use cases. With the prevalence of the COVID-19 pandemic over the past year, companies and governments have been forced to reconsider the distribution of vaccines at scale, and with disputes arising in the European Union surrounding the supply of vaccines from Astra Zeneca in the UK, and from Pfizer in India, a method must be found which ensures traceability, security, and accountability of vaccine orders - all of which being features that blockchain can facilitate.

In an attempt to cope with the unprecedented demand and tension, we present a permissioned, blockchain-based system: VaxDist, for the management of supply chains in the distribution of vaccines. We explore the use of blockchain technology in the system, evaluate its success, and discuss its potential for production deployment.

2 Goals

At current, existing systems centralise databases which are fundamentally managed by governments or their subsidiaries. This is seen in US' implementation: The Vaccine Tracking System, VTrckS (CDC, 2020), developed and controlled by the Center for Disease Control and Prevention (CDC).

By nature of the blockchain, the "need to utilize an intra-operable vaccine administration system to ensure individual records are not duplicated and monitor reports of adverse events" (Makvandi et al., 2020) is avoided through its key feature of immutability. As such, we are able to keep all parts of the vaccine supply and distribution chain within one system.

Obviously such an approach raises questions about trust and accountability, particularly to those members of the public who have lost confidence in their governments (as we saw in some countries throughout the COVID-19 pandemic of 2020/21). Is the government trying to help us or are they just providing falsified data to create a façade?

The goals we outline below aim to address these issues, with the wholistic aim of providing a guarantee in times of uncertainty.

1. Design and produce a **proof of concept (PoC)** system with:
 - (a) **Client nodes** - Anybody who wishes request a shipment of vaccines (eg: Hospitals, Doctors practices, etc...).
 - (b) **Distributor nodes** - Those belonging to vaccine manufacturers, or anybody else who is involved in distribution of vaccines (eg: Pfizer, Astra Zeneca, etc...).
2. Create a **graphical user interface** for both types of nodes.
3. Create a **command line interface** that can be used on machines without graphical output.
4. Design a blockchain which allows network nodes to **join and leave** the network at their own leisure.
5. Develop a protocol which validates shipments and requests via a **consensus model**.
6. **Understand** the potential for use of a blockchain based vaccine supply and distribution at scale, and understand the feasibility of the application in a production environment.

3 Methodology

3.1 Design

3.1.1 Use Case Diagram

The use case diagram below outlines the basic actions that clients (top) and distributors (bottom) may undertake in their use of the VaxDist system. All actions, except viewing node consensus power can be executed via either the command line interface (CLI) or graphical user interface (GUI).

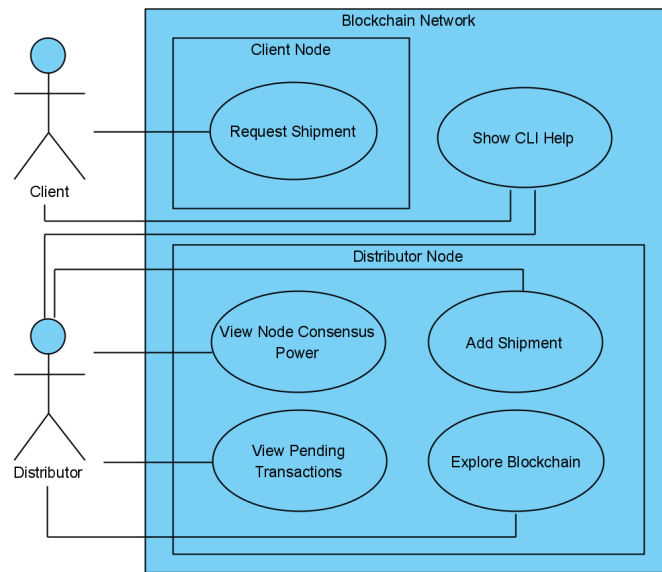


Figure 1: Use Case Diagram

3.1.2 Activity Diagrams

This first activity diagram represents the workflow that occurs on the network once a client (node) places a vaccine order on the network, either via the CLI or GUI.

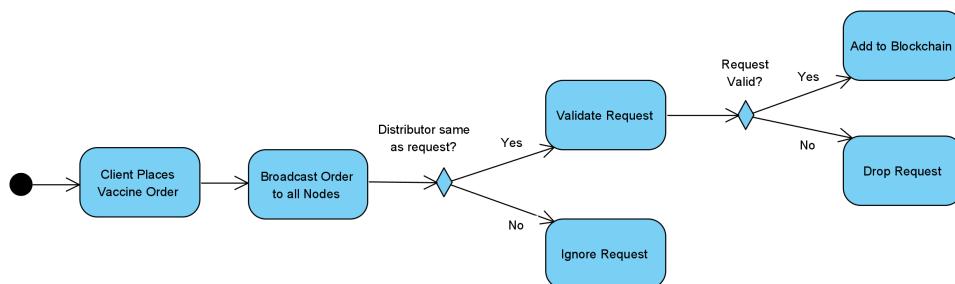


Figure 2: Request Shipment Activity Diagram

The next diagram shows how the network behaves once a distributor (node) enters shipment information for a pending shipment onto the VaxDist network.

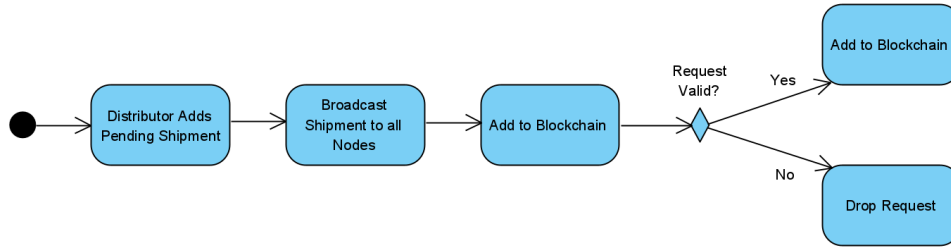


Figure 3: Add Shipment Activity Diagrams

3.2 Supporting Technologies

When deciding on a development environment, it was important to consider the support for blockchain and distributed systems in languages, as well as their available libraries. With this in mind, and based on team experience, we quickly made the decision to implement the blockchain itself in Python 3. A variety of modules were used to assist implementation, the most important of which being RabbitMQ (Rabbit Technologies Ltd., 2007), which provides a message queueing system to allow the network nodes to communicate easily and with little overhead.

The graphical user interface was developed using HTML, Bootstrap, CSS, and JavaScript. An API was developed using Python Flask (Ronacher, 2010) to facilitate interaction with the network nodes. Whilst a frontend web framework such as React may have produced a better UI, we avoided this approach as UI design is out of the scope of the project, and it would likely introduce unnecessary complexities.

3.3 Storing the Blockchain

Each node has its own version of blockchain which is stored when the node goes offline, so as to retrieve it when node comes back online. Python's object serialization library named 'pickle' was used for this purpose. Pickle produces two files:

1. blockchain.pickle - The node's instance of the blockchain (all blocks, their information, and transactions).
2. pending_transactions.pickle - The pending transactions for the current node.

3.4 Block Generation

Each node generates a new block independently to aggregate its pending transactions, and then broadcasts it over the network for validation. A new block may be created in two cases:

1. The limit is reached for the maximum number of pending transactions allowed (5 in our case).
2. The timeout for addition of pending transactions to a block is reached.

4 Results

The system which the team has implemented fulfils all goals, requirements, and hypotheses mentioned in the goals section. The features of this blockchain have been implemented from scratch, in order to cater to all user requirements that would arise in an efficient vaccine logistics system.

In this section we outline what the project has achieved and present our results.

4.1 Goal 1

Design and produce a proof of concept (PoC) system with client nodes and distributor nodes.

Our proof of concept system (VaxDist) is capable of containing both client and distributor nodes. Client's can request a batch of vaccines, and a distributor node can fulfil the request once they are ready to do so.

The use case diagram in figure 1 illustrates the actions that each of these entities can undertake using our system.

4.2 Goal 2

Create a graphical user interface for both types of nodes.

Client nodes and distributor nodes can access the blockchain through a specifically designed graphical user interface (GUI).

Figures 4-8 (Appendices, GUI sub-section) highlight the layout and operation of the GUI.

4.3 Goal 3

Create a command line interface that can be used on machines without graphical output.

Client nodes and distributor nodes can access the blockchain through a command line interface (CLI).

Figure 9 (Appendices, CLI sub-section) shows the operation of the CLI. The interface is the same for both client and distributor nodes, simply with functionality restricted as required for the permitted node functionality.

4.4 Goal 4

Design a blockchain which allows network nodes to join and leave the network at their own leisure.

VaxDist allows client and distributor nodes to join and leave at will. Client nodes are only allowed to request a batch of vaccines from distributors. Distributor nodes, however, are much more capable. Unlike client nodes, they hold the blockchain (and therefore can explore it and view pending transactions), handle block / transaction validation, and submit vaccine shipments.

For example, the blockchain may have many nodes from the company Pfizer. For a transaction, one of the Pfizer nodes will be selected for distribution of the request for a batch of vaccines (based on the transaction created by a client node). The distributor node selected is allowed to leave the network at any time. In such a case, another distributor node will pick up the request for processing.

4.5 Goal 5

Develop a protocol which validates shipments and requests via a consensus model.

The VaxDist system checks all attributes of every shipment at each point of its supply chain path.

Considering a hypothetical example: A package of 10,000 vaccine shots are to be delivered from a factory in Mumbai, India to a hospital in London, UK. This batch of vaccines are manufactured and distributed by Pfizer and is to be taken to its destination location via Dubai, UAE. Our system will ensure that the details of the package are the same when the package is received in Dubai and when it is received in London.

The important attributes of this package will be number of vaccine shots in the package (10,000), distributor name (Pfizer), origin location (Mumbai), destination location (London), etc... In the case of any mismatch of the attributes, the transaction validation will fail, and thus will not be added to the block. This ensures that the package is exactly as ordered by the client, and it remains the same throughout the supply chain path.

4.6 Goal 6

Understand the potential for use of a blockchain based vaccine supply and distribution at scale, and understand the feasibility of the application in a production environment.

The evaluation section covers this goal in detail.

5 Evaluation

The cardinal aim of this project was to design an access control system for a permissioned blockchain for distribution and supply of COVID-19 vaccines. This section will evaluate the different features of the project and characterise them as strengths and weaknesses.

5.1 Strengths of the project

1. Using a blockchain to drive the vaccine supply chain can be especially beneficial as a blockchain allows us to view and track the date, location, quality, certification, and other relevant information essential to maintaining the supply chain. The **block explorer** implemented in the project allows the users to search for a particular piece of data on the blockchain including all the transactions and transaction histories.
2. The transparent track and trace feature of blockchains ensure that only **authorised vaccines** that have been registered and approved are in circulation. This saves the cost of implementing an expensive tracing systems that might burden low and middle income countries.
3. When a client sends a request for a vaccine, the blockchain automatically identifies the correct distributor and only that particular distributor can validate the request. Features like these **automate** the supply chain considerably.
4. A blockchain increases opportunities for planning between suppliers and customers. This enables efficient inventory management and shortage identification while **decreasing risks** associated with forecasting the requirement of vaccines and stock shortages.
5. Blockchain allows for efficient and **fast market recalls** by accurately identifying unsafe vaccines and discovering the exact point of contamination.
6. Caters to **vaccine specific requirements** e.g. the blockchain can be integrated with temperature sensors to ensure refrigeration.

5.2 Weaknesses of the project

1. Establishing an international ecosystem is critical for the blockchain, but this is difficult due to the high cost of implementation and resource requirements. Implementing blockchain solutions also requires significant modification or even complete replacement of existing systems, making it difficult for companies to transition to a blockchain system.
2. A blockchain represents a total shift away from conventional approaches, international organisations might not agree with placing trust and authority in a decentralised network.

6 Future Lessons

In this section we discuss a number potential future improvements of VaxDist, and features that would've been implemented without time constraints. These features are as follows:

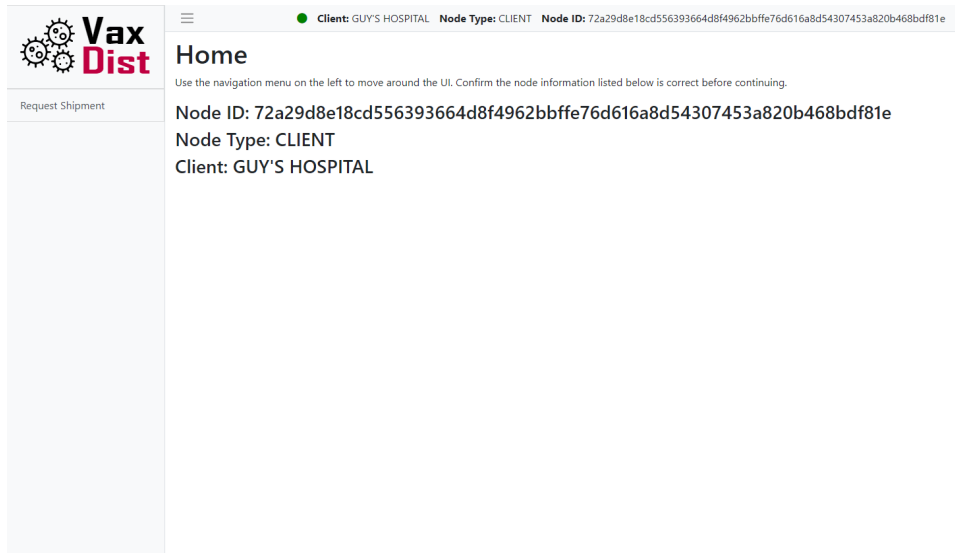
- A future version of the network could contain a complete dashboard that shows all the clients and distributors a detailed flow of the order and supply chain. This would enable stakeholders to prioritize their order requirements, and would also give the clients and distributors transparency with respect to the movement of shipments in real time.
- The dashboard could have advanced predictive statics that help the clients understand how the distributors can meet the requirements of an order on time. Machine Learning techniques could be employed to do this.
- The network could be be also be available for doctors (as well as distributors), so they are informed about vaccine availability and can plan their courses of treatment.
- Easy automation of the process of converting the current paper work and structure of the supply chain using AI, enabling a smooth transition from traditional systems to blockchain.
- The future version could contain a different type of consensus so as to make the network dynamically adaptive for various supply chain requirements.
- A priority based system could be useful in future systems, so that disease hotspots get their vaccines as quickly as possible.

7 Bibliography

- [1] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- [2] CDC. *COVID-19 Vaccine IT Overview: Vaccination Reporting | CDC*. [Online; accessed 4. Apr. 2021]. 2020. URL: <https://www.cdc.gov/vaccines/covid-19/reporting/overview/IT-systems.html>.
- [3] Monear Makvandi et al. *Modeling efficient and equitable distribution of COVID-19 vaccines*. Tech. rep. Oct. 2020. DOI: 10.2172/1718986. URL: <https://doi.org/10.2172/1718986>.
- [4] Rabbit Technologies Ltd. *Messaging that just works — RabbitMQ*. [Online; accessed 4. Apr. 2021]. 2007. URL: <https://www.rabbitmq.com>.
- [5] Armin Ronacher. *Flask*. [Online; accessed 5. Apr. 2021]. 2010. URL: <https://palletsprojects.com/p/flask>.

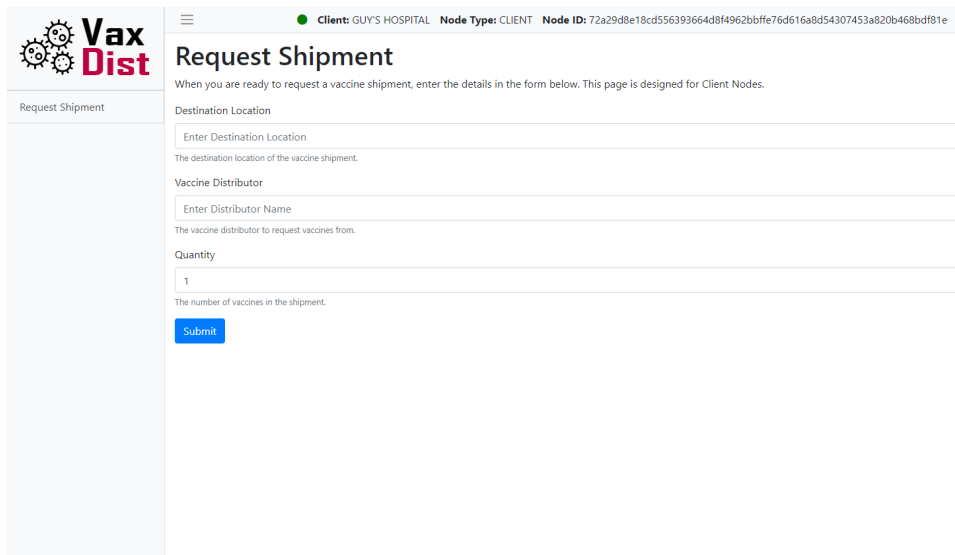
7 Appendices

7.1 GUI




The screenshot shows the 'Home' page of the Vax Dist application. On the left is a sidebar with the 'Vax Dist' logo and a 'Request Shipment' button. The main content area has a header bar with a hamburger menu icon, a status indicator (green dot), and text: 'Client: GUY'S HOSPITAL Node Type: CLIENT Node ID: 72a29d8e18cd556393664d8f4962bbffe76d616a8d54307453a820b468bdf81e'. Below the header, the title 'Home' is followed by a sub-header: 'Use the navigation menu on the left to move around the UI. Confirm the node information listed below is correct before continuing.' The node information is displayed as: 'Node ID: 72a29d8e18cd556393664d8f4962bbffe76d616a8d54307453a820b468bdf81e', 'Node Type: CLIENT', and 'Client: GUY'S HOSPITAL'.

Figure 4: Landing Page



The screenshot shows the 'Request Shipment' page of the Vax Dist application. The sidebar is identical to the previous page. The main content area has a header bar with the same status and node information. Below the header, the title 'Request Shipment' is followed by a sub-header: 'When you are ready to request a vaccine shipment, enter the details in the form below. This page is designed for Client Nodes.' The form contains three sections: 'Destination Location' with a text input field and a placeholder 'Enter Destination Location'; 'Vaccine Distributor' with a text input field and a placeholder 'Enter Distributor Name'; and 'Quantity' with a text input field and a placeholder '1'. A blue 'Submit' button is at the bottom of the form.

Figure 5: Client - Request Shipment



● **Distributor:** PFIZER **Node Type:** DISTRIBUTOR **Node ID:** 744a27025e2d79996d74a3a66af389c0979ea20d039bc2750bd54334093506fc

Add Shipment

When a shipment is ready to add to the network, enter the details in the form below. This page is designed for Distributor Nodes.

Shipment Id

The id of the shipment.

Source Location

The source location of the vaccine shipment.

Destination Location


The destination location of the vaccine shipment.

Quantity

The number of vaccines in the shipment.

[Submit](#)

Figure 6: Distributor - Add Shipment



● **Distributor:** PFIZER **Node Type:** DISTRIBUTOR **Node ID:** 744a27025e2d79996d74a3a66af389c0979ea20d039bc2750bd54334093506fc

Pending Transactions

Below are the pending transactions for this node.

ID	Origin/Client	Source	Destination	Qty	Distributor	Date/Time
N/A	GUY'S HOSPITAL	N/A	London, UK	350	Pfizer	10/4/2021 22:19:51

Figure 7: Distributor - View Pending Transactions

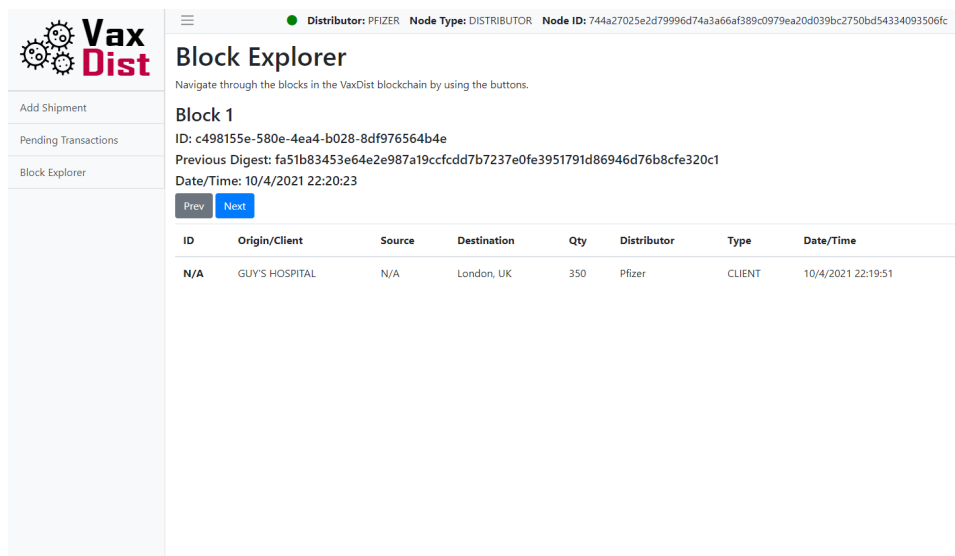


Figure 8: Distributor - Block Explorer

7.2 CLI

```
python node.py

Node initialising...
Successfully identified blockchain on this device. Loading into memory...
Initialising consensus mechanism...

7009d2d9e6aa3f9c2fdccc8320c2e7c41a250481cc55549693ec1e148e6dd3a1 GUY'S HOSPITAL

Node successfully started! Ready for input. Type 'help' to see available commands.

>
API online on port 5000 and listening for connections from the GUI!
> help
HELP          Displays this help menu.
ADDSHIPMENT   Enters a shipment into the network (distributor only).
REQUESTSHIPMENT Requests a vaccine shipment (client only).
SHOWCONSENSUS Displays the consensus power of all nodes, or a certain node if specified in the first parameter.
SHOWPENDING   Displays all transactions that are waiting to be put into a block.
SHOWBLOCKCHAIN Displays the entire blockchain.
GUI           Open the GUI for this node.

> |
```

Figure 9: Client/Distributor - Command Line Interface

7.3 Code

```
/
├── node.py
├── blockchain.py
├── block.py
├── communication.py
├── cli.py
├── api.py
├── gui
│   ├── css
│   │   └── style.css
│   └── js
│       ├── add-shipment.js
│       ├── api-port.js
│       ├── get-block.js
│       ├── get-pending-transactions.js
│       ├── get-status.js
│       └── request-shipment.js
├── add-shipment.html
├── block-explorer.html
├── index.html
├── pending-transactions.html
├── request-shipment.html
└── transactions
    ├── client_transaction.py
    └── distributor_transaction.py
```

7.3.1 node.py

```
1  from cli import CLI
2  from communication import Communication
3  from consensus import Consensus
4  from api import API
5  import pickle
6  import os
7  import atexit
8  import signal
9
10 from blockchain import Blockchain
11 import yaml
12 import hashlib
13 import random
14
15
16 class Node:
17     """
```



```

18     Singleton class - Can only be one instance of the node class
    ↪ per node.
19
20     Handle all node operations that don't relate to the actual
    ↪ blockchain data structure.
21     Networking may be handled here.
22
23     """
24
25     single_instance = None
26
27     def __init__(self):
28         self.NODE_ID = None
29         self.NODE_NAME = None
30         self.NODE_TYPE = None
31         self.setup_node()
32         # Create the blockchain instance for this node if one
    ↪ doesn't exist
33         if not os.path.exists("blockchain.pickle"):
34             self.blockchain =
    ↪ Blockchain.instance(node_id=self.NODE_ID,
    ↪ node_name=self.NODE_NAME,
    ↪ node_type=self.NODE_TYPE)
35             print("First time booting! Creating genesis block...")
36             self.blockchain.create_genesis()
37         else:
38             print("Successfully identified blockchain on this
    ↪ device. Loading into memory...")
39             loaded_blockchain = self.load_blockchain()
40             loaded_pending_transactions =
    ↪ self.load_pending_transactions()
41             self.blockchain =
    ↪ Blockchain.instance(loaded_blockchain,
    ↪ loaded_pending_transactions, self.NODE_ID,
    ↪ self.NODE_NAME, self.NODE_TYPE)
42         API.instance()
43         Consensus.instance(self.NODE_NAME)
44
45         self.node_comm = Communication.instance(self.NODE_ID,
    ↪ self.NODE_NAME, self.blockchain, self.NODE_TYPE)
46         self.blockchain.set_communication(self.node_comm)
47         print(self.NODE_ID, self.NODE_NAME)
48         # Send a connection request to other distributor nodes in
    ↪ network
49         if self.NODE_TYPE == 'DISTRIBUTOR':
50             self.node_comm.notify_connection('connect')
51
52     @staticmethod
53     def instance():
54         if Node.single_instance == None:
55             Node.single_instance = Node()
56         return Node.single_instance
57

```

```

58     # Recreate the current block so that transactions can continue
59     ↪ to be added to it even if
60     # the node has stopped midway through the block.
61     def __reinststate_current_block(self,
62     ↪ loaded_pending_transactions):
63         blockchain = self.blockchain.get_blockchain()
64         curr_block = blockchain[-1]
65         curr_block.set_transactions(loaded_pending_transactions)
66         blockchain[-1] = curr_block
67         self.blockchain.set_blockchain(blockchain)
68
69     # Save the blockchain to the blockchain.pickle file.
70     def dump_blockchain(self):
71         print("Saving blockchain state...")
72         with open("blockchain.pickle", "wb") as f:
73             pickle.dump(self.blockchain.get_blockchain(), f,
74             ↪ pickle.HIGHEST_PROTOCOL)
75
76     # Load the blockchain from the blockchain.pickle file.
77     def load_blockchain(self):
78         with open("blockchain.pickle", "rb") as f:
79             return pickle.load(f)
80
81     # Save the pending transactions to the
82     ↪ pending_transactions.pickle file.
83     def dump_pending_transactions(self):
84         print("Saving pending transactions...")
85         with open("pending_transactions.pickle", "wb") as f:
86
87             ↪ pickle.dump(self.blockchain.get_pending_transactions(),
88             ↪ f, pickle.HIGHEST_PROTOCOL)
89
90     # Load the pending transactions from the
91     ↪ pending_transactions.pickle file.
92     def load_pending_transactions(self):
93         with open("pending_transactions.pickle", "rb") as f:
94             loaded_pending_transactions = pickle.load(f)
95             return loaded_pending_transactions
96
97     def disconnect(self):
98         self.node_comm.notify_connection('disconnect')
99
100     # Setup type of node and its details
101     # Store in config.yaml if started first time
102     # Else read it from existing config.yaml
103     def setup_node(self):
104         if os.path.exists("config.yaml"):
105             with open("config.yaml", "r") as config:
106                 loaded_config = yaml.safe_load(config)
107                 self.NODE_ID = loaded_config["node_id"]
108                 self.NODE_TYPE = loaded_config["node_type"]
109                 if self.NODE_TYPE == "DISTRIBUTOR":
110                     self.NODE_NAME = loaded_config["distributor"]
111                 else:

```

```

105         self.NODE_NAME = loaded_config["client"]
106         config.close()
107     else:
108         print("No node ID found. Generating one...")
109         self.NODE_ID =
110             ↪ hashlib.sha256(str(random.getrandbits(256))
111             .encode('utf-8')).hexdigest()
112         self.NODE_TYPE = input("Are you a client or
113             ↪ distributor? (client/distributor) : ").upper()
114         while not self.NODE_TYPE in ["CLIENT", "DISTRIBUTOR"]:
115             self.NODE_TYPE = input("Are you a client or
116             ↪ distributor? (client/distributor) : ").upper()
117         self.NODE_NAME = "N/A"
118         if self.NODE_TYPE == "DISTRIBUTOR":
119             self.NODE_NAME = input(
120                 "No distributor found. What distributor do you
121                 ↪ belong to? (eg: Pfizer) : ").upper()
122             with open("config.yaml", "w") as config:
123                 yaml.dump({"node_id": self.NODE_ID,
124                     ↪ "distributor": self.NODE_NAME,
125                     ↪ "node_type": self.NODE_TYPE},
126                     config)
127             config.close()
128
129         else:
130             self.NODE_NAME = input(
131                 "No client found. What is the client's name?
132                 ↪ (eg: Guy's Hospital) : ").upper()
133             with open("config.yaml", "w") as config:
134                 yaml.dump({"node_id": self.NODE_ID, "client":
135                     ↪ self.NODE_NAME, "node_type":
136                     ↪ self.NODE_TYPE},
137                     config)
138             config.close()
139
140 if __name__ == "__main__":
141     print("\nNode initialising...")
142     node = Node.instance()
143     atexit.register(node.dump_blockchain)
144     atexit.register(node.dump_pending_transactions)
145     print("\nNode successfully started! Ready for input. Type
146     ↪ 'help' to see available commands.")
147
148     command = [""]
149     while command[0] not in ["EXIT", "QUIT", "Q"]:
150         command = input("\n> ").upper().split(" ")
151         CLI(command[0], None or command[1:])
152
153     node.disconnect()
154     node.dump_blockchain()
155     node.dump_pending_transactions()
156     print("\nExiting!")
157     os.kill(os.getpid(), signal.SIGTERM)

```

7.3.2 blockchain.py

```
1  from communication import Communication
2  import pickle
3  import threading
4  import time
5  from queue import Queue, Empty
6
7  from block import Block
8  import uuid
9
10 from transactions.client_transaction import ClientTransaction
11 from transactions.distributor_transaction import
   ↪ DistributorTransaction
12
13
14 class Blockchain:
15     """
16     Singleton class - Can only be one instance of the blockchain
   ↪ per node.
17     Get the instance via the static instance() method.
18
19     """
20
21     single_instance = None
22
23     comm = None
24     node_id = None
25
26     max_pending_transactions = 5
27     max_block_wait_time = 60
28
29     candidate_blocks = Queue()
30     temp_blocks = dict()
31     consensus_results = dict()
32
33     thread_wait = None
34
35     def __init__(self, blockchain, pending_transactions, node_id,
   ↪ node_name, node_type):
36         self.blockchain = blockchain
37         self.pending_transactions = pending_transactions
38         self.node_id = node_id
39         self.node_type = node_type
40         self.node_name = node_name
41         self.start_block_wait_timer()
42
43     def get_node_id(self):
44         return self.node_id
45
46     def get_node_name(self):
47         return self.node_name
48
49     def get_node_type(self):
```

```

50         return self.node_type
51
52     def start_block_wait_timer(self):
53         self.thread_wait =
54             ↳ threading.Thread(target=self.new_block_timer,
55                               ↳ daemon=True)
56         self.thread_wait.start()
57
58     # Display the blockchain nicely when printed.
59     def __str__(self):
60         blockchain_string = "[\n"
61         for b in self.blockchain:
62             blockchain_string += str(b)
63             blockchain_string += "\n]"
64         return blockchain_string
65
66     # STATIC - Fetch and return the single instance of the
67     ↳ blockchain, or if it doesn't already exist, instantiate
68     ↳ it.
69     @staticmethod
70     def instance(blockchain=[], pending_transactions=[],
71                 ↳ node_id=None, node_name=None, node_type=None):
72         if Blockchain.single_instance is None:
73             Blockchain.single_instance = Blockchain(blockchain,
74                                                       ↳ pending_transactions, node_id, node_name,
75                                                       ↳ node_type)
76         return Blockchain.single_instance
77
78     # Set instance to communicate with network
79     def set_communication(self, comm):
80         self.comm = comm
81
82     # Fetch the entire blockchain data structure.
83     def get_blockchain(self):
84         return self.blockchain
85
86     # Set the blockchain data structure.
87     def set_blockchain(self, blockchain):
88         self.blockchain = blockchain
89
90     # Fetch and return the pending transactions for the current
91     ↳ block.
92     def get_pending_transactions(self):
93         return self.pending_transactions
94
95     # Create the first block in the blockchain.
96     def create_genesis(self):
97         genesis = Block(
98             uuid.uuid4(), # Block ID
99             0, # block number in blockchain
100             "", # Transactions for this block.
101             time.time(), # Block timestamp.
102             "Evening Standard 02/03/2021 Hunt for mutant carrier
103             ↳ continues amid row over face masks in schools."

```

```

95         )
96         self.blockchain.append(genesis)
97         return genesis
98
99     # Initialise and add a new block to the blockchain.
100     def create_block(self):
101         print(len(self.pending_transactions))
102         block = Block(
103             id=str(uuid.uuid4()), # Block ID.
104             block_number=len(self.blockchain), # Block number
105             transactions=self.pending_transactions, #
106             ↳ Transactions for this block.
107             timestamp=time.time(), # Block timestamp.
108             previous_digest=Block.get_digest(self.blockchain[-1])
109             ↳ # Get the hash digest of the last block.
110         )
111         self.pending_transactions = [] # Clears pending
112         ↳ transactions
113         self.temp_blocks[block.id] = block # Add newly created
114         ↳ block to temporary blocks
115         self.consensus_results[block.id] = [] # Stores consensus
116         ↳ results for this block's validation
117         self.comm.broadcast_block(block) # Broadcast this block
118         ↳ over the network for validation
119         return block
120
121     # If block is validated based on consensus, add it to
122     ↳ blockchain
123     def block_validated(self, blockId):
124         if blockId not in self.temp_blocks:
125             return
126         print("\nBlock validated through consensus. Adding",
127             ↳ blockId, "to blockchain.\n> ", end="")
128         block = self.temp_blocks[blockId] # Retrieve block from
129         ↳ temporary blocks
130         self.blockchain.append(block) # Append block to
131         ↳ blockchain
132         self.temp_blocks.pop(blockId) # Remove from temporary
133         ↳ blocks
134         self.consensus_results.pop(blockId) # Clear this
135         ↳ block's consensus results
136         print("\n", str(self), "\n> ", end="")
137
138     # Create a transaction (shipment) and returns it
139     def create_distributor_transaction(self, shipment_id,
140         ↳ origin_node, src_location, dest_location, qty, type,
141         distributor):
142         transaction = DistributorTransaction(
143             shipment_id,
144             # self.__generate_shipment_id(distributor),
145             origin_node,
146             src_location,
147             dest_location,
148             qty,

```

```

136         distributor,
137         type,
138         time.time()
139     )
140     return transaction
141
142     # Create a transaction (request) and returns it
143     def create_client_transaction(self, client, dest_location,
144     ↪ qty, distributor, type):
145         transaction = ClientTransaction(
146             # self.__generate_shipment_id(client),
147             client,
148             dest_location,
149             qty,
150             distributor,
151             type,
152             time.time()
153         )
154         return transaction
155
156     # PRIVATE - Generate a shipment identifier in the format:
157     ↪ BlockNum-TransactionInBlockNum/DistributorInitials
158     def __generate_distributor_transaction_id(self, distributor):
159         block_id = str(len(self.blockchain) - 1)
160         transaction_id = str(len(self.pending_transactions))
161         distributor_id = "".join([i[0] for i in
162     ↪ distributor.split(" ")])
163
164         return block_id + "-" + transaction_id + "/" +
165     ↪ distributor_id
166
167     # PRIVATE - Generate a shipment identifier in the format:
168     ↪ BlockNum-TransactionInBlockNum/ClientInitials
169     def __generate_client_transaction_id(self, client):
170         block_id = str(len(self.blockchain) - 1)
171         transaction_id = str(len(self.pending_transactions))
172         client_id = "".join([i[0] for i in client.split(" ")])
173
174         return block_id + "-" + transaction_id + "/" + client_id
175
176     # Timer thread that creates block with pending transactions on
177     ↪ timeout
178     def new_block_timer(self):
179         while True:
180             count = 0
181             while count < self.max_block_wait_time and
182     ↪ self.thread_wait.is_alive():
183                 time.sleep(5)
184                 count += 5
185             if len(self.pending_transactions) != 0 and
186     ↪ self.thread_wait.is_alive():
187                 print("\nBlock Timer Expired. Creating block!\n>
188     ↪ ", end="")
189                 self.create_block()

```

```

181         if self.thread_wait.is_alive() is False:
182             self.start_block_wait_timer()

```

7.3.3 block.py

```

1  import hashlib
2  import json
3
4
5  class Block:
6      """
7      Manage the structure of blocks in the blockchain.
8
9      """
10
11     def __init__(self, id, block_number, transactions, timestamp,
12     ↪ previous_digest=None):
13         self.id = id
14         self.block_number = block_number
15         self.timestamp = timestamp
16         self.transactions = transactions
17         self.previous_digest = previous_digest
18
19     # Display the block nicely when printed.
20     def __str__(self):
21         transactions_string = ""
22
23         transactions_string = "["
24         for t in self.transactions:
25             transactions_string += str(t)
26             transactions_string += "\n"
27         transactions_string += "]"
28
29         block_string = f"""
30         {{
31             "id": "{self.id}",
32             "block_number": "{self.block_number}",
33             "timestamp": "{self.timestamp}",
34             "transactions": {transactions_string},
35             "previous_digest": "{self.previous_digest}"
36         }},\n"""
37         return block_string
38
39     def get_transactions(self):
40         return self.transactions
41
42     def set_transactions(self, transactions):
43         self.transactions = transactions
44
45     # STATIC - Get the hash of a given block.
46     @staticmethod
47     def get_digest(block):
48         block_as_json = json.dumps(str(block)).encode()
49         digest = hashlib.sha256(block_as_json).hexdigest()

```



```
48         return digest
```

7.3.4 communication.py

```
1  import threading
2  import time
3  from collections import Set
4
5  import jsonpickle
6  import pika
7
8
9  # Handles the communication between nodes in the blockchain
   ↪ network
10 class Communication():
11     connected_nodes = set()      # Set of connected nodes in the
   ↪ network
12     single_instance = None
13
14     my_Lock = threading.Lock()
15
16     def __init__(self, id, node_name, blockchain, node_type):
17         self.id = id
18         self.node_name = node_name
19         self.blockchain = blockchain
20         if node_type == 'DISTRIBUTOR':      # Register to
   ↪ listeners only for distributor nodes
21             thread_validation =
   ↪ threading.Thread(target=self.register_listeners,
   ↪ args=(), daemon=True)
   ↪ thread_validation.start()
22
23
24     @staticmethod
25     def instance(id=None, node_name=None, blockchain=None,
   ↪ node_type=None):
26         if Communication.single_instance == None:
27             Communication.single_instance = Communication(id,
   ↪ node_name, blockchain, node_type)
28         return Communication.single_instance
29
30     # Broadcast all nodes in network
31     # status = connect/disconnect
32     def notify_connection(self, status):
33         connection = pika.BlockingConnection(
34             pika.ConnectionParameters(host='localhost'))
35         channel = connection.channel()
36         channel.exchange_declare(status, exchange_type='fanout')
37         channel.basic_publish(exchange=status, routing_key='',
   ↪ body=self.id)
38         connection.close()
39
40     # Broadcast all nodes in network a block for validation
41     def broadcast_block(self, block):
```

```

42         connection = pika.BlockingConnection(
43             pika.ConnectionParameters(host='localhost'))
44         channel = connection.channel()
45         channel.exchange_declare('block_validation_request',
46             ↪ exchange_type='fanout')
47         channel.basic_publish(exchange='block_validation_request',
48             ↪ routing_key='',
49                                     body=jsonpickle.encode(block,
50                                     ↪ unpicklable=True))
51         connection.close()
52
53     # Broadcast all nodes in network a transaction for validation
54     def broadcast_transaction(self, transaction):
55         connection = pika.BlockingConnection(
56             pika.ConnectionParameters(host='localhost'))
57         channel = connection.channel()
58         channel.exchange_declare('transaction',
59             ↪ exchange_type='fanout')
60         channel.basic_publish(exchange='transaction',
61             ↪ routing_key='',
62                                     body=jsonpickle.encode(transaction,
63                                     ↪ unpicklable=True))
64         connection.close()
65
66     # Broadcast all nodes in network whether a block is validated
67     ↪ or not
68     def broadcast_block_validation_result(self, result):
69         connection = pika.BlockingConnection(
70             pika.ConnectionParameters(host='localhost'))
71         channel = connection.channel()
72         channel.exchange_declare('block_validation_result',
73             ↪ exchange_type='fanout')
74         channel.basic_publish(exchange='block_validation_result',
75             ↪ routing_key='', body=result)
76         connection.close()
77
78     # Register for various listeners over the network
79     def register_listeners(self):
80         connection =
81             ↪ pika.BlockingConnection(pika.ConnectionParameters(host='localhost'))
82         channel = connection.channel()
83
84         # Setup listener for node connection
85         channel.exchange_declare('connect',
86             ↪ exchange_type='fanout')
87         connectionQueue =
88             ↪ channel.queue_declare(queue='').method.queue
89         channel.queue_bind(connectionQueue, 'connect')
90         channel.basic_consume(queue=connectionQueue,
91             ↪ on_message_callback=self.node_connected,
92             ↪ auto_ack=True)
93
94     # Setup listener for node disconnection

```

```

81     channel.exchange_declare('disconnect',
82                               ↪ exchange_type='fanout')
83     disconnectionQueue =
84     ↪ channel.queue_declare(queue='').method.queue
85     channel.queue_bind(disconnectionQueue, 'disconnect')
86     channel.basic_consume(queue=disconnectionQueue,
87                           ↪ on_message_callback=self.node_disconnected,
88                           ↪ auto_ack=True)
89
90     # Setup listener for block validation approval
91     channel.exchange_declare('block_validation_result',
92                               ↪ exchange_type='fanout')
93     validationResultQueue =
94     ↪ channel.queue_declare(queue='').method.queue
95     channel.queue_bind(validationResultQueue,
96                           ↪ 'block_validation_result')
97     channel.basic_consume(queue=validationResultQueue,
98                           ↪ on_message_callback=self.block_validation_result,
99                           ↪ auto_ack=True)
100
101     # Setup listener for new transaction in the network
102     channel.exchange_declare('transaction',
103                               ↪ exchange_type='fanout')
104     validationResultQueue =
105     ↪ channel.queue_declare(queue='').method.queue
106     channel.queue_bind(validationResultQueue, 'transaction')
107     channel.basic_consume(queue=validationResultQueue,
108                           ↪ on_message_callback=self.new_transaction,
109                           ↪ auto_ack=True)
110
111     # Setup listener for block validation request
112     channel.exchange_declare('block_validation_request',
113                               ↪ exchange_type='fanout')
114     validationRequestQueue =
115     ↪ channel.queue_declare(queue='').method.queue
116     channel.queue_bind(validationRequestQueue,
117                           ↪ 'block_validation_request')
118     channel.basic_consume(queue=validationRequestQueue,
119                           ↪ on_message_callback=self.block_validation_request,
120                           ↪ auto_ack=True)
121
122     channel.start_consuming()
123
124     # Callback when a new node is connected over network
125     def node_connected(self, ch, method, properties, body):
126         remoteId = str(body).split("\n")[1]
127         if remoteId != self.id and remoteId not in
128         ↪ self.connected_nodes:
129             print("\nNode connected: ", remoteId, "\n> ", end="")
130             self.connected_nodes.add(remoteId)
131             self.notify_connection('connect')
132
133     # Callback when a node is disconnected from network
134     def node_disconnected(self, ch, method, properties, body):

```

```

119         remoteId = str(body).split("'")[1]
120         if remoteId != self.id and remoteId in
121             ↪ self.connected_nodes:
122                 print("\nNode disconnected: ", remoteId, "\n> ",
123                     ↪ end="")
124                 self.connected_nodes.remove(remoteId)
125
126     # Callback when a new pending transaction is received over
127     ↪ network
128     def new_transaction(self, ch, method, properties, body):
129         transaction = jsonpickle.decode(body)
130         if transaction not in
131             ↪ self.blockchain.pending_transactions:
132                 if transaction.get_transaction_type() == 'CLIENT':
133                     if transaction.get_requested_distributor().upper()
134                         ↪ != self.node_name.upper():
135                         return
136                 print("\nNew request received!\n> ", end="")
137
138                 ↪ self.blockchain.pending_transactions.append(transaction)
139
140     # Callback on receiving a validation from another node for a
141     ↪ block
142     def block_validation_result(self, ch, method, properties,
143         ↪ body):
144         result = str(body).split("'")[1].split(" ")
145         node_id = result[0]
146         block_id = result[1]
147         validationResult = result[2]
148         print("\nResult:", result, "\n> ", end="")
149
150         with self.my_Lock:
151             if block_id in self.blockchain.temp_blocks:
152                 block = self.blockchain.temp_blocks[block_id]
153                 if block.block_number <=
154                     ↪ self.blockchain.blockchain[-1].block_number:
155                     # Discard block since it was late in
156                     ↪ reaching/requesting consensus
157                     # and another block is already added to
158                     ↪ blockchain
159                     # Maybe send approval failure?
160                     return
161             if validationResult == 'success':
162                 # Increase consensus count for this block and
163                 ↪ check if 50% reached
164                 if block_id in
165                     ↪ self.blockchain.consensus_results.keys():
166                     consensus_results =
167                     ↪ self.blockchain.consensus_results[block_id]
168                     if node_id not in consensus_results:
169                         consensus_results.append(node_id)
170
171                 # Reach more than 50% consensus

```

```

158         # Change the measure to stake value rather
        ↪     than node count?
159         if len(consensus_results) >
        ↪     ((len(self.connected_nodes) + 1) / 2):
        ↪     # Add approved block to blockchain
160         self.blockchain.block_validated(block_id)
161     else:
162
        ↪     self.blockchain.consensus_results[block_id]
        ↪     = consensus_results
163
164     else:
165         # Received this block first time. Add to
        ↪     consensus list and wait for 50% approval
166         self.blockchain.consensus_results[block_id] =
        ↪     [node_id]
167
168     # Callback on receiving a validation request for a new block
    ↪     over the network
169     def block_validation_request(self, ch, method, properties,
    ↪     body):
170         block = jsonpickle.decode(body)
171
172         with self.my_Lock:
173             if block.block_number <=
        ↪     self.blockchain.blockchain[-1].block_number:
174             # Discard block since it was late in
        ↪     reaching/requesting consensus
175             # and another block is already added there
176             # Maybe send approval failure?
177             return
178
179             print("\nValidate block: ", block, "\n> ", end="")
180             time.sleep(10)
181             # Received a new block, add it to temp list and wait
        ↪     for 50% consensus
182             self.blockchain.temp_blocks[block.id] = block
183             if block.id not in
        ↪     self.blockchain.consensus_results.keys():
184                 self.blockchain.consensus_results[block.id] = []
185
        ↪     self.blockchain.consensus_results[block.id].append(self.id)
186
187         # If block is validated, broadcast it to all nodes
188         # Message contains current node's id, block id, approval
        ↪     result
189         return self.broadcast_block_validation_result(str(self.id)
        ↪     + " " + str(block.id) + " " + "success")

```

7.3.5 cli.py

```

1  from communication import Communication
2  from consensus import Consensus
3  from blockchain import Blockchain
4  import webbrowser

```

```

5  import os
6  from api import API
7
8
9  class CLI:
10     def __init__(self, command, params):
11         if command == "HELP":
12             self.__help()
13         elif command == "ADDSHIPMENT":
14             self.__addshipment(params)
15         elif command == "REQUESTSHIPMENT":
16             self.__requestshipment(params)
17         elif command == "SHOWCONSENSUS":
18             self.__consensus(params)
19         elif command == "SHOWPENDING":
20             self.__pending_transactions(params)
21         elif command == "SHOWBLOCKCHAIN":
22             self.__blockchain(params)
23         elif command == "GUI":
24             self.__gui(params)
25
26         elif command == "":
27             pass
28         elif not command in ["EXIT", "QUIT", "Q"]:
29             print("Invalid input. Type 'help' to see available
30                   ↪ commands.")
31
32     def __help(self):
33         print("HELP                Displays this help menu.")
34         print("ADDSHIPMENT            Enters a shipment into the
35           ↪ network (distributor only).")
36         print("REQUESTSHIPMENT        Requests a vaccine shipment
37           ↪ (client only).")
38         print("SHOWCONSENSUS          Displays the consensus power of
39           ↪ all nodes, or a certain node if specified in the first
40           ↪ parameter.")
41         print("SHOWPENDING            Displays all transactions that
42           ↪ are waiting to be put into a block.")
43         print("SHOWBLOCKCHAIN          Displays the entire blockchain.")
44         print("GUI                    Open the GUI for this node.")
45
46     def __addshipment(self, params):
47         if len(params) == 0 or params[0] in ["?", "HELP"]:
48             self.__show_command_help(
49                 usage="addshipment id srclocation destlocation
50                   ↪ qty",
51                 description="Enters a shipment into the network."
52             )
53         elif len(params) == 3:
54             shipment_id = params[0]
55             src = params[1]
56             dest = params[2]
57             qty = params[3]

```

```

52         confirmation = input(
53             "Are you sure you want to add this shipment to the
             ↳ blockchain? This action is irreversible
             ↳ without the agreement of all network nodes
             ↳ (y/n): ").upper()

54
55         if confirmation == "Y":
56             print("\nAdding to blockchain...")
57             blockchain = Blockchain.instance()
58
59             self.node_comm = Communication.instance()
60
61             transaction =
             ↳ blockchain.create_distributor_transaction(shipment_id,
             ↳ blockchain.get_node_id(), src, dest, qty,
             ↳ blockchain.get_node_type(),
             ↳ blockchain.get_node_name())
62             self.node_comm.broadcast_transaction(transaction)
63             print("Shipment is now pending.\n> ", end="")
64
65     def __requestshipment(self, params):
66         if len(params) == 0 or params[0] in ["?", "HELP"]:
67             self.__show_command_help(
68                 usage="requestshipment destlocation qty
                 ↳ distributor",
69                 description="Requests a shipment of vaccines from
                 ↳ a given distributor."
70             )
71         elif len(params) == 3:
72             dest = params[0]
73             qty = params[1]
74             distributor = params[2]
75
76             confirmation = input(
77                 "Are you sure you want to request this shipment?
                 ↳ This action is irreversible without the
                 ↳ agreement of all network nodes (y/n):
                 ↳ ").upper()
78
79             if confirmation == "Y":
80                 print("\nAdding to blockchain...")
81                 blockchain = Blockchain.instance()
82
83                 self.node_comm = Communication.instance()
84
85                 transaction =
                 ↳ blockchain.create_client_transaction(blockchain
86                 .get_node_name(), dest, qty, distributor,
                 ↳ blockchain.get_node_type())
87
88                 self.node_comm.broadcast_transaction(transaction)
89                 print("Request has been received and is now
                 ↳ pending.\n> ", end="")
90

```

```

91 def __consensus(self, params):
92     if len(params) > 0 and params[0] in ["?", "HELP"]:
93         self.__show_command_help(
94             usage="consensus [node]",
95             description="Displays the consensus power of all
96                 ↪ nodes, or a certain node if specified in the
97                 ↪ first parameter."
98         )
99     else:
100         consensus_power =
101             ↪ Consensus.instance().get_consensus_power()
102         if len(params) > 0:
103             node = params[0]
104             if node in consensus_power:
105                 print(node + " : " +
106                     ↪ str(consensus_power[node]))
107             else:
108                 print("Node " + node + " does not exist on the
109                     ↪ network. Please try again.")
110         else:
111             print(consensus_power)
112
113 def __pending_transactions(self, params):
114     if len(params) > 0 and params[0] in ["?", "HELP"]:
115         self.__show_command_help(
116             usage="showpending",
117             description="Displays all transactions that are
118                 ↪ waiting to be put into a block."
119         )
120     else:
121         for transaction in
122             ↪ Blockchain.instance().get_pending_transactions():
123             print(str(transaction))
124
125 def __blockchain(self, params):
126     if len(params) > 0 and params[0] in ["?", "HELP"]:
127         self.__show_command_help(
128             usage="showblockchain",
129             description="Displays the entire blockchain."
130         )
131     else:
132         print(str(Blockchain.instance()))
133
134 # Execute the GUI.
135 def __gui(self, params):
136     if len(params) > 0 and params[0] in ["?", "HELP"]:
137         self.__show_command_help(
138             usage="gui",
139             description="Open the GUI for this node."
140         )
141     else:
142         webbrowser.open("file:/// " + os.getcwd() +
143             ↪ "/gui/index.html", new=2)
144         with open("gui/js/api-port.js", "w") as f:

```



```

137         f.write("window.apiPort = " +
    ↪       str(API.instance().get_port()) + ";"")
138
139     def __show_command_help(self, usage, description="No
    ↪     description available for this command."):
140         print("Usage:\n " + usage)
141         print("Description:\n " + description)

```

7.3.6 api.py

```

1  from communication import Communication
2  from blockchain import Blockchain
3  from flask import Flask, request, jsonify
4  from threading import Thread
5  from flask_cors import CORS, cross_origin
6  import os
7  import socket
8  import webbrowser
9  import logging
10 from waitress import serve
11
12
13 class API:
14     """
15     Singleton class - Can only be one instance of the API per
    ↪   node.
16     Get the instance via the static instance() method.
17
18     """
19
20     app = Flask(__name__)
21
22     single_instance = None
23
24     def __init__(self):
25         log = logging.getLogger('werkzeug')
26         log.setLevel(logging.ERROR)
27         cors = CORS(API.app)
28         API.app.config['CORS_HEADERS'] = 'Content-Type'
29         Thread(target=self.start_server).start()
30
31     @staticmethod
32     def instance():
33         if API.single_instance == None:
34             API.single_instance = API()
35         return API.single_instance
36
37     def start_server(self):
38         self.port = 5000
39         while self.is_port_in_use(self.port):
40             self.port += 1
41         with open("gui/js/api-port.js", "w") as f:
42             f.write("window.apiPort = " + str(self.port) + ";"")

```

```

43         f.close()
44
45     print("\nAPI online on port", str(self.port), "and
↳ listening for connections from the GUI!\n> ", end="")
46     try:
47         webbrowser.open("file:/// " + os.getcwd() +
↳ "/gui/index.html", new=2)
48         serve(app=API.app, port=self.port)
49     except:
50         print("An error occurred when starting the REST API
↳ server. Please restart the node and try again.
↳ Otherwise, simply use the CLI.")
51
52     def is_port_in_use(self, port):
53         with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as
↳ s:
54             return s.connect_ex(('localhost', port)) == 0
55
56     def get_port(self):
57         if API.single_instance:
58             return self.port
59         return None
60
61     @app.route("/add-shipment", methods=["POST"])
62     @cross_origin()
63     def add_shipment():
64         response = jsonify(success=False, status_code=500)
65         if request.method == "POST":
66             shipment_id = request.form["shipment-id"]
67             src = request.form["src-location"]
68             dest = request.form["dest-location"]
69             qty = request.form["qty"]
70
71             print("\nAdding to blockchain...")
72
73             blockchain = Blockchain.instance()
74             node_comm = Communication.instance()
75             transaction =
↳ blockchain.create_distributor_transaction(shipment_id,
↳ blockchain.get_node_id(), src, dest, qty,
↳ blockchain.get_node_type(),
↳ blockchain.get_node_name())
76             node_comm.broadcast_transaction(transaction)
77             print("Shipment is now pending.\n> ", end="")
78
79             response = jsonify(success=True, status_code=200)
80         return response
81
82     @app.route("/request-shipment", methods=["POST"])
83     @cross_origin()
84     def request_shipment():
85         response = jsonify(success=False, status_code=500)
86         if request.method == "POST":
87             dest = request.form["dest-location"]

```

```

88         qty = request.form["qty"]
89         distributor = request.form["distributor"]
90
91         print("\nAdding to blockchain...")
92
93         blockchain = Blockchain.instance()
94         node_comm = Communication.instance()
95         transaction =
96             ↪ blockchain.create_client_transaction(blockchain
97             .get_node_name(), dest, qty, distributor,
98             ↪ blockchain.get_node_type())
99         node_comm.broadcast_transaction(transaction)
100
101         print("Request has been received and is now
102             ↪ pending.\n> ", end="")
103         response = jsonify(success=True, status_code=200)
104     return response
105
106 @app.route("/get-node-info", methods=["GET"])
107 @cross_origin()
108 def get_distributor():
109     blockchain = Blockchain.instance()
110     response = jsonify(success=True, status_code=200,
111         ↪ distributor_client=blockchain.get_node_name(),
112         ↪ node_id=blockchain.get_node_id(),
113         ↪ node_type=blockchain.get_node_type())
114     return response
115
116 @app.route("/get-pending-transactions", methods=["GET"])
117 @cross_origin()
118 def get_pending_transactions():
119     blockchain = Blockchain.instance()
120     pending_transactions =
121         ↪ blockchain.get_pending_transactions()
122     response = jsonify(success=True, status_code=200,
123         ↪ pending_transactions=[str(transaction).strip()[:-1]
124         ↪ for transaction in pending_transactions])
125     return response
126
127 @app.route("/get-block", methods=["GET"])
128 @cross_origin()
129 def get_block():
130     blockchain = Blockchain.instance().get_blockchain()
131     block_index = request.args.get("blockIndex", default=0,
132         ↪ type=int)
133     if abs(block_index) < len(blockchain):
134         block = str(blockchain[block_index - 1]).strip()
135         response = jsonify(success=True, status_code=200,
136             ↪ block=block[:-1])
137     else:
138         response = jsonify(success=True, status_code=404)
139     return response

```

7.3.7 style.css

```
1  #wrapper {
2      overflow-x: hidden;
3  }
4
5  #page-content-wrapper {
6      min-width: 100vw;
7  }
8
9  #sidebar-wrapper {
10     min-height: 100vh;
11     margin-left: -15rem;
12     -webkit-transition: margin 0.25s ease-out;
13     -moz-transition: margin 0.25s ease-out;
14     -o-transition: margin 0.25s ease-out;
15     transition: margin 0.25s ease-out;
16 }
17
18 #sidebar-wrapper .sidebar-heading {
19     padding: 0.875rem 1.25rem;
20     font-size: 1.2rem;
21 }
22
23 #sidebar-wrapper .list-group {
24     width: 15rem;
25 }
26
27 #wrapper.toggled #sidebar-wrapper {
28     margin-left: 0;
29 }
30
31 @media (min-width: 768px) {
32     #sidebar-wrapper {
33         margin-left: 0;
34     }
35
36     #page-content-wrapper {
37         min-width: 0;
38         width: 100%;
39     }
40
41     #wrapper.toggled #sidebar-wrapper {
42         margin-left: -15rem;
43     }
44 }
45
46 .navbar-toggler-icon:hover {
47     cursor: pointer;
48 }
49
50 .alert {
51     margin-top: 15px;
52 }
```

```

53
54 #distributor-client,
55 #status-indicator,
56 #node-type,
57 #node-id {
58     padding: 0 15px 0 0;
59 }
60
61 .break-all {
62     word-break: break-all;
63 }
64
65 #menu-toggle {
66     min-width: 35px;
67 }
68
69 @media (max-width: 1400px) {
70     .navbar-nav {
71         display: none;
72     }
73 }
74
75 .scrollable {
76     overflow-x: auto;
77 }

```

7.3.8 add-shipment.js

```

1  /**
2   * Triggered on submission of the "Add Shipment" form (Distributor
   ↪ Nodes)
3   *
4   * Submits the information to the relevant API endpoint.
5   */
6  $("#add-shipment-form").submit(function (event) {
7      event.preventDefault();
8
9      $.ajax({
10         type: "POST",
11         url: "http://localhost:" + window.apiPort + "/add-shipment",
12         dataType: "json",
13         data: $("#add-shipment-form").serialize(),
14         success: function () {
15             $("#add-shipment-success").removeClass("d-none");
16             $("#add-shipment-failed").addClass("d-none");
17             console.log("Request successful!");
18         },
19         error: function () {
20             $("#add-shipment-failed").removeClass("d-none");
21             $("#add-shipment-success").addClass("d-none");
22             console.log("Request failed. Check that the node is online
   ↪ and try again.");
23         },

```

```

24     });
25 });

```

7.3.9 api-port.js

```

1 window.apiPort = 5000;

```

7.3.10 get-block.js

```

1 window.blockIndex = 0; //0 - most recent, -1 = current block -1,
  ↪ -2 = current block -2, etc...
2
3 /**
4  * Render the current block on page load.
5  */
6 $(document).ready(function () {
7     getBlock();
8 });
9
10 /**
11  * Fetches a block, all of its information, and transactions from
12  ↪ the blockchain.
13  * Executed on page load, or on the press of the "Next" and "Prev"
14  ↪ buttons.
15  * Renders the transactions and block information on the screen if
16  ↪ in valid range.
17  */
18 function getBlock() {
19     $.ajax({
20         type: "GET",
21         data: { blockIndex: window.blockIndex },
22         dataType: "json",
23         url: "http://localhost:" + window.apiPort + "/get-block",
24         success: function (response) {
25             if (response.status_code == 404) {
26                 console.log("You're at the genesis block!");
27                 window.blockIndex += 1;
28             } else {
29                 block = JSON.parse(response.block.replaceAll(/}, (
30                 ↪ |\n)*\]/gi, "}]"));
31                 $("#explorer-block-number").html(block.block_number);
32                 $("#explorer-block-id").html(block.id);
33
34                 ↪ $("#explorer-block-previous-digest").html(block.previous_digest);
35                 block_timestamp = new Date(Math.trunc(block.timestamp) *
36                 ↪ 1000);
37                 $("#explorer-block-date-time").html(
38                 ↪ `${block_timestamp.getDate()}/${
39                 ↪     block_timestamp.getMonth() + 1
40                 ↪ }/${block_timestamp.getFullYear() }
41                 ↪     ${block_timestamp.getHours()}:
42                 ↪     ${block_timestamp.getMinutes()}:${block_timestamp.getSeconds()}`

```

```

36     );
37     if (window.blockIndex == 0) {
38         $("#explorer-block-number").append(" (Latest)");
39     }
40
41     console.log(block);
42     transactions = block.transactions;
43     console.log(transactions);
44     $("#block-explorer-table-body").html("");
45
46     for (transaction in transactions) {
47         transaction = transactions[transaction];
48         timestamp = new Date(Math.trunc(transaction.timestamp) *
↪      1000);
49
50         if (transaction.transaction_type == "DISTRIBUTOR") {
51             midway = Math.round(transaction.origin_node.length /
↪      2);
52             origin_node = transaction.origin_node.substr(0,
↪      midway) + "<wbr>" +
↪      transaction.origin_node.substr(midway + 1,
↪      transaction.origin_node.length);
53         }
54         $("#block-explorer-table-body").append(`
55             <tr>
56                 <th scope="row">${transaction.transaction_type ==
↪      "DISTRIBUTOR" ? transaction.shipment_id : "N/A"}</th>
57                 <td>${transaction.transaction_type ==
↪      "DISTRIBUTOR" ? origin_node : transaction.client}</td>
58                 <td>${transaction.transaction_type ==
↪      "DISTRIBUTOR" ? transaction.src_location : "N/A"}</td>
59                 <td>${transaction.dest_location}</td>
60                 <td>${transaction.qty}</td>
61                 <td>${transaction.distributor}</td>
62                 <td>${transaction.transaction_type}</td>
63                 <td>${timestamp.getDate()}/
64                 ${timestamp.getMonth() + 1}/
65                 ${timestamp.getFullYear()}
66                 ${timestamp.getHours()}:
67                 ${timestamp.getMinutes()}:
68                 ${timestamp.getSeconds()}</td>
69             </tr>
70         `);
71     }
72     if (transactions.length == 0) {
73         $("#block-explorer-table").addClass("d-none");
74         // ("no - pending - transactions").removeClass("d - none");
75     } else {
76         $("#block-explorer-table").removeClass("d-none");
77         // ("no - pending - transactions").addClass("d - none");
78     }
79 }
80 },
81 error: function () {

```

```

82         console.log("Request failed. Check that the node is online
      ↪ and try again.");
83     },
84     });
85 }
86
87 $("#explorer-prev-btn").click(function () {
88     window.blockIndex -= 1;
89     getBlock();
90 });
91
92 $("#explorer-next-btn").click(function () {
93     if (window.blockIndex < 0) {
94         window.blockIndex += 1;
95         getBlock();
96     }
97 });

```

7.3.11 get-pending-transactions.js

```

1  /**
2   * Triggered when the user loads the "Pending Transactions" page
   ↪ (Distributor)
3   *
4   * Retrieves transactions from the appropriate endpoint.
5   * Adds all of the transactions to the pending transactions table.
6   */
7  $.ajax({
8      type: "GET",
9      url: "http://localhost:" + window.apiPort +
   ↪ "/get-pending-transactions",
10     success: function (response) {
11         pending_transactions = response.pending_transactions;
12         for (transaction in pending_transactions) {
13             console.log(pending_transactions[transaction]);
14             transaction = JSON.parse(pending_transactions[transaction]);
15             timestamp = new Date(Math.trunc(transaction.timestamp) *
   ↪ 1000);
16
17             if (transaction.transaction_type == "DISTRIBUTOR") {
18                 midway = Math.round(transaction.origin_node.length / 2);
19                 origin_node = transaction.origin_node.substr(0, midway) +
   ↪ "<wbr>" + transaction.origin_node.substr(midway + 1,
   ↪ transaction.origin_node.length);
20             }
21             $("#pending-transactions-table-body").append(`
22                 <tr>
23                     <th scope="row">${transaction.transaction_type ==
   ↪ "DISTRIBUTOR" ? transaction.shipment_id : "N/A"}</th>
24                     <td>${transaction.transaction_type == "DISTRIBUTOR" ?
   ↪ origin_node : transaction.client}</td>
25                     <td>${transaction.transaction_type == "DISTRIBUTOR" ?
   ↪ transaction.src_location : "N/A"}</td>

```



```

26         <td>${transaction.dest_location}</td>
27         <td>${transaction.qty}</td>
28         <td>${transaction.distributor}</td>
29         <td>${timestamp.getDate()}/${timestamp.getMonth() +
↪ 1}/
30         ${timestamp.getFullYear()} ${timestamp.getHours()}:
31 ↪ ${timestamp.getMinutes()}:${timestamp.getSeconds()}</td>
32     </tr>
33     `);
34 }
35 if (pending_transactions.length == 0) {
36     $("#pending-transactions-table").addClass("d-none");
37     $("#no-pending-transactions").removeClass("d-none");
38 } else {
39     $("#pending-transactions-table").removeClass("d-none");
40     $("#no-pending-transactions").addClass("d-none");
41 }
42 },
43 error: function () {
44     console.log("Request failed. Check that the node is online and
↪ try again.");
45 },
46 });

```

7.3.12 get-status.js

```

1  /**
2   * Periodically check the status of the node every 10 seconds.
3   * Show red icon and appropriate information if the node is
↪ offline. Green if all okay.
4   * Adapts the UI based on whether the node is a Client or
↪ Distributor.
5   */
6  let statusCheck = window.setInterval(
7      (function getStatus() {
8          $.ajax({
9              type: "GET",
10             url: "http://localhost:" + window.apiPort +
↪ "/get-node-info",
11             success: function (response) {
12                 $("#status-indicator").attr("style", "color: green
↪ !important");
13                 $(".node-status-block").removeClass("d-none");
14                 $("#node-status-error").addClass("d-none");
15                 $("#node-id").html(response.node_id);
16                 $("#distributor-client").html("<strong>" +
↪ (response.node_type == "CLIENT" ? "Client: " :
↪ "Distributor: ") + "</strong>" +
↪ response.distributor_client);
17                 $("#distributor-client-index").html((response.node_type ==
↪ "CLIENT" ? "Client: " : "Distributor: ") +
↪ response.distributor_client);

```

```

18     $("#node-id-index").html(response.node_id);
19     $("#node-type").html(response.node_type);
20     $("#node-type-index").html(response.node_type);
21     $("#node-status-index-error").addClass("d-none");
22     $("#node-status-index").removeClass("d-none");
23     if (response.node_type == "CLIENT") {
24         $(".client-nav").removeClass("d-none");
25         $(".distributor-nav").addClass("d-none");
26     } else {
27         $(".client-nav").addClass("d-none");
28         $(".distributor-nav").removeClass("d-none");
29     }
30 },
31 error: function () {
32     $("#status-indicator").attr("style", "color: red
33     ↪ !important");
34     $(".node-status-block").addClass("d-none");
35     $("#node-status-error").removeClass("d-none");
36     $("#node-id").html("");
37     $("#node-status-index").addClass("d-none");
38     $("#node-status-index-error").removeClass("d-none");
39 });
40 })(),
41 10000
42 );

```

7.3.13 request-shipment.js

```

1  /**
2   * Triggered on submission of the "Request Shipment" form (Client
3   * ↪ Nodes)
4   *
5   * Submits the information to the relevant API endpoint.
6   */
7   $("#request-shipment-form").submit(function (event) {
8       event.preventDefault();
9
10      $.ajax({
11          type: "POST",
12          url: "http://localhost:" + window.apiPort +
13          ↪ "/request-shipment",
14          dataType: "json",
15          data: $("#request-shipment-form").serialize(),
16          success: function () {
17              $("#request-shipment-success").removeClass("d-none");
18              $("#request-shipment-failed").addClass("d-none");
19              console.log("Request successful!");
20          },
21          error: function () {
22              $("#request-shipment-failed").removeClass("d-none");
23              $("#request-shipment-success").addClass("d-none");
24              console.log("Request failed. Check that the node is online
25              ↪ and try again.");

```

```

23     },
24   });
25 });

```

7.3.14 add-shipment.html

```

1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8" />
6    <meta name="viewport" content="width=device-width,
    ↪   initial-scale=1, shrink-to-fit=no" />
7
8    <title>VaxDist - Add Shipment</title>
9
10   <link rel="stylesheet"
    ↪   href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/
11   bootstrap.min.css"
    ↪   integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsL0n8PA
12   Rn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
13   crossorigin="anonymous" />
14
15   <link href="css/style.css" rel="stylesheet" />
16   <link rel="shortcut icon" type="image/jpg"
    ↪   href="images/favicon.ico" />
17 </head>
18
19 <body>
20   <div class="d-flex" id="wrapper">
21     <div class="bg-light border-right" id="sidebar-wrapper">
22       <a class="list-group-item list-group-item-action bg-light"
    ↪   href="index.html"></img></a>
23       <div class="list-group list-group-flush">
24         <div class="distributor-nav d-none">
25           <a href="add-shipment.html" class="list-group-item
    ↪   list-group-item-action bg-light">Add Shipment</a>
26           <a href="pending-transactions.html"
    ↪   class="list-group-item list-group-item-action
    ↪   bg-light">Pending Transactions</a>
27           <a href="block-explorer.html" class="list-group-item
    ↪   list-group-item-action bg-light">Block Explorer</a>
28         </div>
29         <div class="client-nav d-none"><a
    ↪   href="request-shipment.html" class="list-group-item
    ↪   list-group-item-action bg-light">Request
    ↪   Shipment</a></div>
30       </div>
31     </div>
32
33     <div id="page-content-wrapper">
34       <nav class="navbar navbar-expand-lg navbar-light bg-light
    ↪   border-bottom">

```

```

35 <span class="navbar-toggler-icon border-0"
    ↳ id="menu-toggle"></span>
36
37 <div class="collapse navbar-collapse"
    ↳ id="navbarSupportedContent">
38   <ul class="navbar-nav ml-auto mt-2 mt-lg-0">
39     <li class="nav-item" id="status-indicator"><i
        ↳ class="fas fa-circle"></i></li>
40     <li class="nav-item d-none"
        ↳ id="node-status-error">Node Offline. Check console
        ↳ for errors.</li>
41     <li class="nav-item node-status-block"><span
        ↳ id="distributor-client">Querying Node
        ↳ Status...</span></li>
42     <li class="nav-item node-status-block
        ↳ d-none"><strong>Node Type:</strong> <span
        ↳ id="node-type"></span></li>
43     <li class="nav-item node-status-block
        ↳ d-none"><strong>Node ID:</strong> <span
        ↳ class="break-all" id="node-id"></span></li>
44   </ul>
45 </div>
46 </nav>
47
48 <div class="container-fluid">
49   <h1 class="mt-2">Add Shipment</h1>
50   <p>When a shipment is ready to add to the network, enter
    ↳ the details in the form below. This page is designed
    ↳ for Distributor Nodes.</p>
51
52   <form id="add-shipment-form">
53     <div class="form-group">
54       <label for="shipment-id">Shipment Id</label>
55       <input required type="text" class="form-control"
        ↳ id="shipment-id" name="shipment-id"
        ↳ aria-describedby="shipment-id" placeholder="Enter
        ↳ Shipment Id" />
56       <small class="form-text text-muted">The id of the
        ↳ shipment.</small>
57     </div>
58     <div class="form-group">
59       <label for="src-location">Source Location</label>
60       <input required type="text" class="form-control"
        ↳ id="src-location" name="src-location"
        ↳ aria-describedby="src-location" placeholder="Enter
        ↳ Source Location" />
61       <small class="form-text text-muted">The source
        ↳ location of the vaccine shipment.</small>
62     </div>
63     <div class="form-group">
64       <label for="dest-location">Destination
        ↳ Location</label>

```

```

65         <input required type="text" class="form-control"
        ↪ id="dest-location" name="dest-location"
        ↪ aria-describedby="dest-location"
        ↪ placeholder="Enter Destination Location" />
66     <small class="form-text text-muted">The destination
        ↪ location of the vaccine shipment.</small>
67 </div>
68 <div class="form-group">
69     <label for="qty">Quantity</label>
70     <input required min="1" value="1" type="number"
        ↪ class="form-control" id="qty" name="qty"
        ↪ aria-describedby="qty" placeholder="Enter
        ↪ Quantity" />
71     <small class="form-text text-muted">The number of
        ↪ vaccines in the shipment.</small>
72 </div>
73 <button type="submit" class="btn btn-primary
        ↪ submit-btn">Submit</button>
74 </form>
75 <div class="alert alert-success d-none"
        ↪ id="add-shipment-success" role="alert">Successfully
        ↪ added shipment to the blockchain! Click <a
        ↪ href="pending-transactions.html">here</a> to view the
76     transaction.</div>
77 <div class="alert alert-danger d-none"
        ↪ id="add-shipment-failed" role="alert">Failed to add
        ↪ shipment to blockchain. Check the node is online and
        ↪ try again.</div>
78 </div>
79 </div>
80 </div>
81
82 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
        ↪ integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
        ↪ crossorigin="anonymous"></script>
83 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
84     js/bootstrap.bundle.min.js"
        ↪ integrity="sha384-LtrjvnR4Twt/qOuYxE721ul19sVFLVSA4hf
85     /rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
        ↪ crossorigin="anonymous"></script>
86 <script src="https://kit.fontawesome.com/4e701a347d.js"
        ↪ crossorigin="anonymous"></script>
87 <script src="js/api-port.js"></script>
88 <script src="js/add-shipment.js"></script>
89 <script src="js/get-status.js"></script>
90
91
92 <script>
93     $( "#menu-toggle" ).click( function (e) {
94         e.preventDefault();
95         $( "#wrapper" ).toggleClass( "toggled" );
96     } );
97 </script>
98 </body>
99

```

100 </html>

7.3.15 block-explorer.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="utf-8" />
6      <meta name="viewport" content="width=device-width,
7          ↪ initial-scale=1, shrink-to-fit=no" />
8
9      <title>VaxDist - Block Explorer</title>
10
11     <link rel="stylesheet"
12         ↪ href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
13         css/bootstrap.min.css"
14         ↪ integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsL0n8PA
15         Rn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
16         ↪ crossorigin="anonymous" />
17
18     <link href="css/style.css" rel="stylesheet" />
19     <link rel="shortcut icon" type="image/jpg"
20         ↪ href="images/favicon.ico" />
21 </head>
22
23 <body>
24     <div class="d-flex" id="wrapper">
25         <div class="bg-light border-right" id="sidebar-wrapper">
26             <a class="list-group-item list-group-item-action bg-light"
27                 ↪ href="index.html"></img></a>
28             <div class="list-group list-group-flush">
29                 <div class="distributor-nav d-none">
30                     <a href="add-shipment.html" class="list-group-item
31                         ↪ list-group-item-action bg-light">Add Shipment</a>
32                     <a href="pending-transactions.html"
33                         ↪ class="list-group-item list-group-item-action
34                         ↪ bg-light">Pending Transactions</a>
35                     <a href="block-explorer.html" class="list-group-item
36                         ↪ list-group-item-action bg-light">Block Explorer</a>
37                 </div>
38                 <div class="client-nav d-none"><a
39                     ↪ href="request-shipment.html" class="list-group-item
40                     ↪ list-group-item-action bg-light">Request
41                     ↪ Shipment</a></div>
42             </div>
43         </div>
44         <div id="page-content-wrapper">
45             <nav class="navbar navbar-expand-lg navbar-light bg-light
46                 ↪ border-bottom">
47                 <span class="navbar-toggler-icon border-0"
48                     ↪ id="menu-toggle"></span>
```

```

36
37 <div class="collapse navbar-collapse"
    ↳ id="navbarSupportedContent">
38 <ul class="navbar-nav ml-auto mt-2 mt-lg-0">
39 <li class="nav-item" id="status-indicator"><i
    ↳ class="fas fa-circle"></i></li>
40 <li class="nav-item d-none"
    ↳ id="node-status-error">Node Offline. Check console
    ↳ for errors.</li>
41 <li class="nav-item node-status-block"><span
    ↳ id="distributor-client">Querying Node
    ↳ Status...</span></li>
42 <li class="nav-item node-status-block
    ↳ d-none"><strong>Node Type:</strong> <span
    ↳ id="node-type"></span></li>
43 <li class="nav-item node-status-block
    ↳ d-none"><strong>Node ID:</strong> <span
    ↳ class="break-all" id="node-id"></span></li>
44 </ul>
45 </div>
46 </nav>
47
48 <div class="container-fluid">
49 <h1 class="mt-2">Block Explorer</h1>
50 <p>Navigate through the blocks in the VaxDist blockchain
    ↳ by using the buttons.</p>
51 <h3>Block <span id="explorer-block-number"></span></h3>
52 <h5>ID: <span id="explorer-block-id"></span></h5>
53 <h5>Previous Digest: <span class="break-all"
    ↳ id="explorer-block-previous-digest"></span></h5>
54 <h5>Date/Time: <span
    ↳ id="explorer-block-date-time"></span></h5>
55
56 <div class="scrollable">
57 <table class="table d-none mt-2 "
    ↳ id="block-explorer-table">
58 <button type="button" class="btn btn-secondary btn
    ↳ mr-1" id="explorer-prev-btn">Prev</button>
59 <button type="button" class="btn btn-primary btn"
    ↳ id="explorer-next-btn">Next</button>
60 <thead>
61 <tr>
62 <th scope="col">ID</th>
63 <th scope="col">Origin/Client</th>
64 <th scope="col">Source</th>
65 <th scope="col">Destination</th>
66 <th scope="col">Qty</th>
67 <th scope="col">Distributor</th>
68 <th scope="col">Type</th>
69 <th scope="col">Date/Time</th>
70 </tr>
71 </thead>
72 <tbody id="block-explorer-table-body"></tbody>
73 </table>

```

```

74     </div>
75 </div>
76 </div>
77 </div>
78
79 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    ↳ integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
    ↳ crossorigin="anonymous"></script>
80 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
81 js/bootstrap.bundle.min.js"
    ↳ integrity="sha384-LtrjvnR4Twt/qOuYxE721ul9sVFLVSA4hf/
82 rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
83     crossorigin="anonymous"></script>
84 <script src="https://kit.fontawesome.com/4e701a347d.js"
    ↳ crossorigin="anonymous"></script>
85 <script src="js/api-port.js"></script>
86 <script src="js/get-status.js"></script>
87 <script src="js/get-block.js"></script>
88
89 <script>
90     $( "#menu-toggle" ).click(function (e) {
91         e.preventDefault();
92         $( "#wrapper" ).toggleClass("toggled");
93     });
94 </script>
95 </body>
96
97 </html>

```

7.3.16 index.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8" />
6     <meta name="viewport" content="width=device-width,
    ↳ initial-scale=1, shrink-to-fit=no" />
7
8     <title>VaxDist - Home</title>
9
10    <link rel="stylesheet"
    ↳ href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
11    css/bootstrap.min.css"
    ↳ integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsL0n8PA
12    Rn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
13    crossorigin="anonymous" />
14
15    <link href="css/style.css" rel="stylesheet" />
16    <link rel="shortcut icon" type="image/jpg"
    ↳ href="images/favicon.ico" />
17 </head>
18

```



```

19 <body>
20   <div class="d-flex" id="wrapper">
21     <div class="bg-light border-right" id="sidebar-wrapper">
22       <a class="list-group-item list-group-item-action bg-light"
23         ↪ href="index.html"></img></a>
24       <div class="list-group list-group-flush">
25         <div class="distributor-nav d-none">
26           <a href="add-shipment.html" class="list-group-item
27             ↪ list-group-item-action bg-light">Add Shipment</a>
28           <a href="pending-transactions.html"
29             ↪ class="list-group-item list-group-item-action
30             ↪ bg-light">Pending Transactions</a>
31           <a href="block-explorer.html" class="list-group-item
32             ↪ list-group-item-action bg-light">Block Explorer</a>
33         </div>
34         <div class="client-nav d-none"><a
35           ↪ href="request-shipment.html" class="list-group-item
36           ↪ list-group-item-action bg-light">Request
37           ↪ Shipment</a></div>
38       </div>
39     </div>
40     <div id="page-content-wrapper">
41       <nav class="navbar navbar-expand-lg navbar-light bg-light
42         ↪ border-bottom">
43         <span class="navbar-toggler-icon border-0"
44           ↪ id="menu-toggle"></span><wbr>
45
46         <div class="collapse navbar-collapse"
47           ↪ id="navbarSupportedContent">
48           <ul class="navbar-nav ml-auto mt-2 mt-lg-0">
49             <li class="nav-item" id="status-indicator"><i
50               ↪ class="fas fa-circle"></i></li>
51             <li class="nav-item d-none"
52               ↪ id="node-status-error">Node Offline. Check console
53               ↪ for errors.</li>
54             <li class="nav-item node-status-block"><span
55               ↪ id="distributor-client">Querying Node
56               ↪ Status...</span></li>
57             <li class="nav-item node-status-block
58               ↪ d-none"><strong>Node Type:</strong> <span
59               ↪ id="node-type"></span></li>
60             <li class="nav-item node-status-block
61               ↪ d-none"><strong>Node ID:</strong> <span
62               ↪ class="break-all" id="node-id"></span></li>
63           </ul>
64         </div>
65       </nav>
66
67       <div class="container-fluid">
68         <h1 class="mt-2">Home</h1>
69         <p>Use the navigation menu on the left to move around the
70           ↪ UI. Confirm the node information listed below is
71           ↪ correct before continuing.</p>

```

```

51     <h3 class="d-none" id="node-status-index-error">Node
    ↳ Offline! Check console for errors.</h3>
52     <span id="node-status-index">
53         <h3>Node ID: <span class="break-all"
    ↳ id="node-id-index">Querying...</span></h3>
54         <h3>Node Type: <span
    ↳ id="node-type-index">Querying...</span></h3>
55         <h3 id="distributor-client-index"></h3>
56     </span>
57 </div>
58 </div>
59 </div>
60
61 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    ↳ integrity="sha256-/xUj+3OJU5yExlq6GSYGGShk7tPXikynS7ogEvDej/m4="
    ↳ crossorigin="anonymous"></script>
62 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
63 js/bootstrap.bundle.min.js"
    ↳ integrity="sha384-LtrjvnR4Twt/qOuYxE721u19sVFLVSA4hf
64 /rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
65 crossorigin="anonymous"></script>
66 <script src="https://kit.fontawesome.com/4e701a347d.js"
    ↳ crossorigin="anonymous"></script>
67 <script src="js/api-port.js"></script>
68 <script src="js/get-status.js"></script>
69
70 <script>
71     $('#menu-toggle').click(function (e) {
72         e.preventDefault();
73         $('#wrapper').toggleClass("toggled");
74     });
75 </script>
76 </body>
77
78 </html>

```

7.3.17 pending-transactions.html

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="utf-8" />
6     <meta name="viewport" content="width=device-width,
    ↳ initial-scale=1, shrink-to-fit=no" />
7
8     <title>VaxDist - Pending Transactions</title>
9
10    <link rel="stylesheet"
    ↳ href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
11    css/bootstrap.min.css"
    ↳ integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsL0n8PA
12    Rn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"

```

```

13     crossorigin="anonymous" />
14
15     <link href="css/style.css" rel="stylesheet" />
16     <link rel="shortcut icon" type="image/jpg"
17     ↪ href="images/favicon.ico" />
18 </head>
19 <body>
20     <div class="d-flex" id="wrapper">
21         <div class="bg-light border-right" id="sidebar-wrapper">
22             <a class="list-group-item list-group-item-action bg-light"
23             ↪ href="index.html"></img></a>
24             <div class="list-group list-group-flush">
25                 <div class="distributor-nav d-none">
26                     <a href="add-shipment.html" class="list-group-item
27                     ↪ list-group-item-action bg-light">Add Shipment</a>
28                     <a href="pending-transactions.html"
29                     ↪ class="list-group-item list-group-item-action
30                     ↪ bg-light">Pending Transactions</a>
31                     <a href="block-explorer.html" class="list-group-item
32                     ↪ list-group-item-action bg-light">Block Explorer</a>
33                 </div>
34                 <div class="client-nav d-none"><a
35                 ↪ href="request-shipment.html" class="list-group-item
36                 ↪ list-group-item-action bg-light">Request
37                 ↪ Shipment</a></div>
38             </div>
39         </div>
40         <div id="page-content-wrapper">
41             <nav class="navbar navbar-expand-lg navbar-light bg-light
42             ↪ border-bottom">
43                 <span class="navbar-toggler-icon border-0"
44                 ↪ id="menu-toggle"></span>
45
46                 <div class="collapse navbar-collapse"
47                 ↪ id="navbarSupportedContent">
48                     <ul class="navbar-nav ml-auto mt-2 mt-lg-0">
49                         <li class="nav-item" id="status-indicator"><i
50                         ↪ class="fas fa-circle"></i></li>
51                         <li class="nav-item d-none"
52                         ↪ id="node-status-error">Node Offline. Check console
53                         ↪ for errors.</li>
54                         <li class="nav-item node-status-block"><span
55                         ↪ id="distributor-client">Querying Node
56                         ↪ Status...</span></li>
57                         <li class="nav-item node-status-block
58                         ↪ d-none"><strong>Node Type:</strong> <span
59                         ↪ id="node-type"></span></li>
60                         <li class="nav-item node-status-block
61                         ↪ d-none"><strong>Node ID:</strong> <span
62                         ↪ class="break-all" id="node-id"></span></li>
63                     </ul>
64                 </div>

```

```

46     </nav>
47
48     <div class="container-fluid">
49         <h1 class="mt-2">Pending Transactions</h1>
50         <p>Below are the pending transactions for this node.</p>
51         <div class="alert alert-warning d-none"
52             ↪ id="no-pending-transactions" role="alert">
53             This node currently has no pending transactions. Click
54             ↪ <a href="pending-transactions.html">here</a> to
55             ↪ refresh.
56         </div>
57         <div class="scrollable">
58             <table class="table d-none"
59                 ↪ id="pending-transactions-table">
60                 <thead>
61                     <tr>
62                         <th scope="col">ID</th>
63                         <th scope="col">Origin/Client</th>
64                         <th scope="col">Source</th>
65                         <th scope="col">Destination</th>
66                         <th scope="col">Qty</th>
67                         <th scope="col">Distributor</th>
68                         <th scope="col">Date/Time</th>
69                     </tr>
70                 </thead>
71                 <tbody id="pending-transactions-table-body"></tbody>
72             </table>
73         </div>
74     </div>
75     </div>
76 </div>
77
78 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
79     ↪ integrity="sha256-/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4="
80     ↪ crossorigin="anonymous"></script>
81 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
82     ↪ js/bootstrap.bundle.min.js"
83     ↪ integrity="sha384-LtrjvnR4Twt/qOuYxE721ul19sVFLVSA4hf/
84     ↪ rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
85     ↪ crossorigin="anonymous"></script>
86 <script src="https://kit.fontawesome.com/4e701a347d.js"
87     ↪ crossorigin="anonymous"></script>
88 <script src="js/api-port.js"></script>
89 <script src="js/get-status.js"></script>
90 <script src="js/get-pending-transactions.js"></script>
91
92 <script>
93     $( "#menu-toggle" ).click( function (e) {
94         e.preventDefault();
95         $( "#wrapper" ).toggleClass( "toggled" );
96     } );
97 </script>
98 </body>
99

```

92 </html>

7.3.18 add-shipment.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5    <meta charset="utf-8" />
6    <meta name="viewport" content="width=device-width,
    ↳ initial-scale=1, shrink-to-fit=no" />
7
8    <title>VaxDist - Request Shipment</title>
9
10   <link rel="stylesheet"
    ↳ href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
11   css/bootstrap.min.css"
    ↳ integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpsSL0n8PA
12   Rn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
13   crossorigin="anonymous" />
14
15   <link href="css/style.css" rel="stylesheet" />
16   <link rel="shortcut icon" type="image/jpg"
    ↳ href="images/favicon.ico" />
17 </head>
18
19 <body>
20   <div class="d-flex" id="wrapper">
21     <div class="bg-light border-right" id="sidebar-wrapper">
22       <a class="list-group-item list-group-item-action bg-light"
    ↳ href="index.html"></img></a>
23     <div class="list-group list-group-flush">
24       <div class="distributor-nav d-none">
25         <a href="add-shipment.html" class="list-group-item
    ↳ list-group-item-action bg-light">Add Shipment</a>
26         <a href="pending-transactions.html"
    ↳ class="list-group-item list-group-item-action
    ↳ bg-light">Pending Transactions</a>
27         <a href="block-explorer.html" class="list-group-item
    ↳ list-group-item-action bg-light">Block Explorer</a>
28       </div>
29       <div class="client-nav d-none"><a
    ↳ href="request-shipment.html" class="list-group-item
    ↳ list-group-item-action bg-light">Request
    ↳ Shipment</a></div>
30     </div>
31   </div>
32
33   <div id="page-content-wrapper">
34     <nav class="navbar navbar-expand-lg navbar-light bg-light
    ↳ border-bottom">
35       <span class="navbar-toggler-icon border-0"
    ↳ id="menu-toggle"></span>
```

```

36
37 <div class="collapse navbar-collapse"
    ↳ id="navbarSupportedContent">
38 <ul class="navbar-nav ml-auto mt-2 mt-lg-0">
39 <li class="nav-item" id="status-indicator"><i
    ↳ class="fas fa-circle"></i></li>
40 <li class="nav-item d-none"
    ↳ id="node-status-error">Node Offline. Check console
    ↳ for errors.</li>
41 <li class="nav-item node-status-block"><span
    ↳ id="distributor-client">Querying Node
    ↳ Status...</span></li>
42 <li class="nav-item node-status-block
    ↳ d-none"><strong>Node Type:</strong> <span
    ↳ id="node-type"></span></li>
43 <li class="nav-item node-status-block
    ↳ d-none"><strong>Node ID:</strong> <span
    ↳ class="break-all" id="node-id"></span></li>
44 </ul>
45 </div>
46 </nav>
47
48 <div class="container-fluid">
49 <h1 class="mt-2">Request Shipment</h1>
50 <p>When you are ready to request a vaccine shipment, enter
    ↳ the details in the form below. This page is designed
    ↳ for Client Nodes.</p>
51
52 <form id="request-shipment-form">
53 <div class="form-group">
54 <label for="dest-location">Destination
    ↳ Location</label>
55 <input required type="text" class="form-control"
    ↳ id="dest-location" name="dest-location"
    ↳ aria-describedby="dest-location"
    ↳ placeholder="Enter Destination Location" />
56 <small class="form-text text-muted">The destination
    ↳ location of the vaccine shipment.</small>
57 </div>
58 <div class="form-group">
59 <label for="distributor">Vaccine Distributor</label>
60 <input required type="text" class="form-control"
    ↳ id="distributor" name="distributor"
    ↳ aria-describedby="distributor" placeholder="Enter
    ↳ Distributor Name" />
61 <small class="form-text text-muted">The vaccine
    ↳ distributor to request vaccines from.</small>
62 </div>
63 <div class="form-group">
64 <label for="qty">Quantity</label>
65 <input required min="1" value="1" type="number"
    ↳ class="form-control" id="qty" name="qty"
    ↳ aria-describedby="qty" placeholder="Enter
    ↳ Quantity" />

```

```

66         <small class="form-text text-muted">The number of
        ↪ vaccines in the shipment.</small>
67     </div>
68     <button type="submit" class="btn btn-primary
        ↪ submit-btn">Submit</button>
69 </form>
70 <div class="alert alert-success d-none"
    ↪ id="request-shipment-success"
    ↪ role="alert">Successfully added request to the
    ↪ blockchain!</div>
71 <div class="alert alert-danger d-none"
    ↪ id="request-shipment-failed" role="alert">Failed to
    ↪ add shipment to blockchain. Check the node is online
    ↪ and try again.</div>
72 </div>
73 </div>
74 </div>
75
76 <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    ↪ integrity="sha256-/xUj+3OJU5yExlq6GSYGSXh7tPXikynS7ogEvDej/m4="
    ↪ crossorigin="anonymous"></script>
77 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/
78 js/bootstrap.bundle.min.js"
    ↪ integrity="sha384-LtrjvnR4Twt/qOuYxE721u19sVFLVSA4hf/
79 rRt6PrZTmiPltdZcI7q7PXQBYTKyf"
80 crossorigin="anonymous"></script>
81 <script src="https://kit.fontawesome.com/4e701a347d.js"
    ↪ crossorigin="anonymous"></script>
82 <script src="js/api-port.js"></script>
83 <script src="js/get-status.js"></script>
84 <script src="js/request-shipment.js"></script>
85
86 <script>
87     $("#menu-toggle").click(function (e) {
88         e.preventDefault();
89         $("#wrapper").toggleClass("toggled");
90     });
91 </script>
92 </body>
93
94 </html>

```