Converge

A business blockchain for sophisticated asset management and cross-business transactions

By

ALEX GREIG

Software Design and Development - Task 3
ST AUGUSTINE'S COLLEGE

A project proposal to Big Time Software for the development of *Converge*. A business blockchain that utilises an intelligent consensus mechanism and concepts such as tokenisation, converge will efficiently create a complete, transparent, tamperproof history and facilitation of the information flows, inventory flows, and financial flows in transactions.

23 JUNE 2022

TABLE OF CONTENTS

i

PROBLEM DEFINITION

Currently, businesses globally have to communicate across multiple departments and management systems to facilitate cross-company transactions, leading to costly mistakes, opaque interaction histories, high administration costs and long delays for the transference of capital to be processed. Business blockchains, an innovative solution to this global issue, are reinventing how transactions are managed. They can take time and costs out of almost any process, enabling near real-time operations. And they deliver a high degree of accuracy and control, with much less risk than many alternatives. Due to the immutable nature of the blockchain, a transparent record of all transactions is kept which can be utilised for bookkeeping and taxation purposes while also preventing fraudulent behaviour as all transactions are recorded and can be reproduced for litigious reasons. The business blockchain that is being created, therefore, will help supply chain partners and corporations globally by creating a complete, transparent, tamperproof history and facilitation of the information flows, inventory flows, and financial flows in transactions. This permissioned blockchain will outperform current enterprise resource planning solutions as it is able to manage all transactions extremely quickly and efficiently, through sharing the load of computation across all nodes in the blockchain. An example of the differences between conventional record keeping and blockchain for a transaction is shown below.

Capturing the Details of a Simple Transaction: Conventional vs. Blockchain Systems

The financial ledgers and enterprise resource planning systems now used don't reliably allow the three parties involved in a simple supply-chain transaction to see all the relevant flows of information, inventory, and money. A blockchain system eliminates the blind spots.

The blockchain that is being developed is a proof of concept that will show the core functionality of permissioned blockchains and the benefits it produces. It will be compatible for all businesses as a node will be able to run on a server setup by the development team and users can interface with the system via a user-friendly web frontend to manage transactions and assets.

The assets, labour and even items such as orders / invoices within each company will be tokenised on the blockchain so that businesses can interact with other businesses over the blockchain. Though it most commonly refers to the tokenization of financial or fungible assets, such as shares in a company or a quantity of gold, asset tokenization can hypothetically refer to the tokenization of any material or nonmaterial thing possessing monetary value: everything from a piece of art to a patent to an hour of a skilled worker's time. This process of tokenization creates a bridge between real-world assets and their trading, storage and transfer in a digital world. Another example if the blockchain was to be used in Consultancy is we would enter the number of hours worked on the client for the month on the blockchain as a proposal for an invoice. When the client accepts the number of hours, the hours are posted to the blockchain, and the smart contract (explained in the following) executes the invoice creation based on the agreed hours and prices.

Although the functionality of transferring fungible and non-fungible assets will be implemented into the software solution, the legal processes surrounding the binding of assets will be implemented in the future. These legal proceedings are outside the scope of this project as they would require communication with governing institutions, intricate understanding of taxation law and connecting software solutions with the strenuous obligations of regulatory frameworks.

For the purpose of this conceptual design, I will utilise a main, "converge" token that is not truly backed by US Dollar or another asset, but in production, would be. The manner in which the tokens will be bound to physical assets is through the creation of a stable coin which reassures businesses will not have to consider the fluctuations of token value. To make this collateralised stable coin I would need to own the fiat or assets in which the token was based upon, as 'collateral' or conduct an exchange agreement with a bank in which the converge tokens could be converted into the local fiat currency.

2

ISSUES RELEVANT TO PROGRAM

Blockchain is an open digital ledger technology that has the capability of significantly altering the way that people operate in organizations. Blockchain's ethical issues for business catalyse from its three main promises: immutability, disintermediation, and automation. Immutability results in the permanency of a human past record and raises ethical issues such as privacy and transparency concerns. Disintermediation refers to horizontal decision-making and the numerosity of stakeholders in verifying outcomes which raises ethical issues related to accountability and equal opportunity. Automation refers to the self-executing features of coded agreements called smart contracts which raises issues related to the absence of human decision making and the inability for human intervention. The main ethical issue that will be discussed, however, is the inadvertent asymmetry of power that is being created, as large corporations have access to more bargaining power, information, and efficient transactions then smaller possibly local businesses.

In this sense, this business blockchain might enable transactions that are the product of force or possibly even fraudulent activity that would normally be prevented through institutions such as banks or government bodies. These mediating institutions would normally identify and constrain the misuse of markets by large corporations, however, by managing their assets and transactions through the business blockchain, this might be circumvented. This would enable different types of illegal and immoral transactions by facilitating transactions without intermediaries who can personally be held accountable for those transactions.

A case in point is the "assassination markets." AUGUR is an Ethereum-based blockchain application for the creation of "peer-to-peer prediction markets" which allow people to place bets

secretly. AUGUR has created, invertedly, a utilisation of blockchain analogous to the "assassination market" in that it allows anonymous betting upon someone's death, which in turn may incentivize people to kill others so as to win these very bets.

Although this real-world example is slightly disconnected from business blockchains the same unintended use cases of the software may occur. The actualisation of this ethical issue in this business blockchain may come in the form of businesses trading slave labour that breaks jurisdiction regarding minimum wages and the fair work rights, however, due to the transaction being mediated through blockchain technology, the unethical actions might go unnoticed.

INTERFACE DESIGN

CONVERGE

Converge Node
Mainnet
ws://127.0.1.1:9944
Version 3.0.0 monthly
Build 2022-05-17

187,298,462 CVG

258
Current Block
3 Seconds Remaining

252
Finalised Block
11 Seconds Remaining

Business Name

Fungible Assets
Interaction Type: ● Creation   ○ Trading   ○ Ownership

Asset Type [v]

Product Name: iPhone 13 Pro Max 1TB

Price: 1849 CVG

Quantity: 10 000

STATUS
Finalised. Block Hash.
0x67aac349ff207153710le866

Events
System.ExtrinsicSuccess (Phase=("applyExtrinsic",0))...
DispatchInfo {"weight": 185790180, "pay fees": yes}
System.NFTCreated : (Phase=("applyExtrinsic",...

Non-Fungible Assets
Interaction Type: ○ Creation   ○ Trading   ● Ownership

consulting contract
Block: 4
Owner: 5GrWa4326ff2190f0caac34

Ownership rights to level 23 chiller tower
Block: 124
Owner: 5GrWa4326ff2190f0caac34

Working lease contract for 15 mighty cleaner
Block 235
Owner: 5GrWa4326ff2190f0caac34

Custom PCB circuit board
SN: 392197942
Block: 247
Owner: 5GrWa4326ff2190f0caac34

This sketch is a representation of the web frontend that each business will interact with to contribute to the blockchain. At the top the amount of converge tokens, fungible and non-fungible assets that a business currently owns is displayed. The user interface utilises radio buttons to select the different options associated with fungible and non-fungible assets while also implementing drop down and text boxes for input. The frontend also utilises scroll bars so more information can be fit into the design without cluttering its appearance. This interface is considerate of the user's needs as it does not require a lot of technical understanding to operate. This ease-of-use of the software interface is a testament to the consideration of the audience's needs.

QUALITY ASSURANCE CRITERIA

Quality Assurance is the processes that transpire during the development of a software solution that are taken to ensure the product meets contractual obligations and needs of the client. The quality of the product will be judged on the efficiency and effectiveness of facilitating transactions across businesses, and whether they can tokenise their assets, fungible, and transfer them across to other businesses.

The application needs to be reliable and not 'crash' during function as this would prove detrimental to the requirement for the blockchain to be always running. The system that is created needs to have very high security as it will manage a large quantity of transactions, possibly facilitating billions of dollars in transaction execution. This security is inherent to how the system is built as the consensus mechanisms, the runtime modules are all built in a way that makes it almost impossible to break, this is combined with the decentralised, private nature of the network, allowing businesses to not have to trust a central authorities server. Further, the business blockchain must have an easy-to-use front end that will be used by non-technical employees, however, setting up of the node will need to be undertaken by a development team. More specifically the buttons, forms and layout must be easily navigable and consistent. Finally, the system must go through modular and systematic testing to make sure all components work in unison and function in the way the client wishes.

5.1 Define The Problem

Currently, businesses globally have to communicate across multiple departments and management systems to facilitate cross-company transactions, leading to costly mistakes, opaque interaction histories, high administration costs and long delays for the transference of capital to be processed. Business blockchains, an innovative solution to this global issue, are reinventing how transactions are managed. They can take time and costs out of almost any process, enabling near real-time operations. And they deliver a high degree of accuracy and control, with much less risk than many alternatives. Due to the immutable nature of the blockchain, a transparent record of all transactions is kept which can be utilised for bookkeeping and taxation purposes while also preventing fraudulent behaviour as all transactions are recorded and can be reproduced for litigious reasons. The business blockchain that is being created, therefore, will help supply chain partners and corporations globally by creating a complete, transparent, tamperproof history and facilitation of the information flows, inventory flows, and financial flows in transactions. This permissioned blockchain will outperform current enterprise resource planning solutions as it is able to manage all transactions extremely quickly and efficiently, through sharing the load of computation across all nodes in the blockchain. An example of the differences between conventional record keeping and blockchain for a transaction is shown below.

Capturing the Details of a Simple Transaction: Conventional vs. Blockchain Systems

The financial ledgers and enterprise resource planning systems now used don't reliably allow the three parties involved in a simple supply-chain transaction to see all the relevant flows of information, inventory, and money. A blockchain system eliminates the blind spots.

The blockchain that is being developed is a proof of concept that will show the core functionality of permissioned blockchains and the benefits it produces. It will be compatible for all businesses as a node will be able to run on a server setup by the development team and users can interface with the system via a user-friendly web frontend to manage transactions and assets.

11

The assets, labour and even items such as orders / invoices within each company will be tokenised on the blockchain so that businesses can interact with other businesses over the blockchain. Though it most commonly refers to the tokenization of financial or fungible assets, such as shares in a company or a quantity of gold, asset tokenization can hypothetically refer to the tokenization of any material or nonmaterial thing possessing monetary value: everything from a piece of art to a patent to an hour of a skilled worker's time. This process of tokenization creates a bridge between real-world assets and their trading, storage and transfer in a digital world.

Although the functionality of transferring fungible and non-fungible assets will be implemented into the software solution, the legal processes surrounding the binding of assets will be implemented in the future. These legal proceedings are outside the scope of this project as they would require communication with governing institutions, intricate understanding of taxation law and connecting software solutions with the strenuous obligations of regulatory frameworks.

For the purpose of this conceptual design, I will utilise a main, "converge" token that is not truly backed by US Dollar or another asset, but in production, would be. The manner in which the tokens will be bound to physical assets is through the creation of a stable coin which reassures businesses will not have to consider the fluctuations of token value. To make this collateralised stable coin I would need to own the fiat or assets in which the token was based upon, as 'collateral' or conduct an exchange agreement with a bank in which the converge tokens could be converted into the local fiat currency.

5.2   Economic Feasibility

The economic feasibility of this project can be understood through a number of points of analysis. Firstly, the spending projection of the project needs to be addressed and how much working capital is required for the functioning of the software. Secondly, the cash flows need to be examined, including the revenues and the maintaining outflows of cash.

The spending projection for this project needs to be minimal as this scope of this project does not enable me to spend capital on expensive APIs, or high-end virtual private servers to host nodes for testing and compatibility studies. Thus, I have planned the project to use the most minimal costs possible as this fits within the boundaries of the project. I have utilised open-source frameworks to reduce costs and cut out any dependence on APIs. If this project was to be produced in reality, I would need to set aside 'collateral' in which the converge token was tied to so that it had a physical asset that it was bound to. This would be quite economically infeasible so a better method would be to conduct an exchange agreement with a bank in which

the converge tokens could be converted into the local fiat currency.

The projected income expected from this project will come in the form of small fees that are collected on each transaction, although a small percentage, the sheer volume will make the process very economically advantageous. For example, a percentage for transaction such as 0.1% will be a large incentive for continuing development and maintenance as if a trillion dollars passes through the network, that is around one billion dollars in revenue. If large volumes of transaction are occurring every day, then this will lead to a high profit. Further, as there are no central servers, the maintenance cost of the blockchain will be quite minimal, leading to lower expenditure and thus higher profits.

5.3   Technical Feasibility

Technical feasibility is the process of figuring out how you are going to produce your product or service to determine whether it is possible to create. Thus, with the aid of thorough research the software solution is feasible to create, however, it would require quite sophisticated technology that is only currently being developed. Firms around the globe are all attempting to understand and implement blockchain technology into their business, which reveals the pioneering needed to develop and implement such a software solution. Although it is technically feasible to create, the difficulty of creating will be high. Due to access to blockchain frameworks such as substrate, however, the ease of creating such a software is increased. Substrate is a open-source framework that provides different libraries and templates to create a blockchain. It provides clear documentation and starting guides which increases the technical feasibility as the pathway to creating the blockchain is transparent.

5.4   Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during the problem definition. The software solution that is being proposed will reduce costly mistakes through the utilisation of smart contracts and sophisticated consensus mechanisms, reduce the often-unintelligible transaction histories, and reduce long delays for the transference of assets through sophisticated networking protocols.

The software solution will be a proof of concept / prototype of a fully functioning business blockchain; thus, it will not be able to be utilised by business's globally. This slightly lowers its operational feasibility, however, if successful in demonstrating the capabilities of a business blockchain then further development would be more greatly supported. The software solution will be able to be operationally feasible as its design, although complex and requiring meticulous

pre-thought, is manageable. In conjunction, the maintainability of the software solution is quite high as, inherent to blockchain technology, servers are not needed due to its distributed nature. Through forkless runtime upgrades, the system can be updated and maintained without the need for it to ever be stopped, aiding to the unbeatable uptime. The hard part of this project in reality is establishing a sustainable group of trading partners, with transactions governed by effective smart contracts and clear rules of engagement.

5.5   Scheduling Feasibility

Schedule Feasibility is defined as the probability of a project to be completed within its scheduled time limits, by a planned due date. In comparison to the scale of this project, the timeframe that has been allocated is quite limited. The total time to design, develop, produce and test the software solution is 60 days. This does make completion of the project difficult and does introduce a few scheduling issues into the analysis of the feasibility. To solve this, scheduling tools such as Gantt charts will optimise workflows and time management, enabling development to be at its maximum efficiency. Although there is a very restrictive timeframe, the project is still feasible as long as time is effectively managed.

5.6   Recommendation

Due to each individual aspect of the feasibility report revealing an advantageous position in creating the software, the recommendation is to develop the software solution. This recommendation is based on the high results from each section; however, it will be acknowledged that the production of the software in the timeframe will be quite challenging, due to the sheer magnitude of the software solution proposed.

GANTT CHART

All tasks that are mentioned within the Gantt chart are essential for achieving the minimum time frame and minimum viable product except: Automated document creation, Zero Knowledge proofs and complex smart contracts. The Gantt Chart outlines the tasks that need to be completed in order to design the program, it also outlines the measurement of time and the duration of each of the tasks. By clicking on the link above the Gantt Chart will be opened.

7.1 Information Surrounding Blockchain Development

This section will explain concepts that are fundamental to blockchain development and will give an insight into how all the different parts work together to create a functioning network.

The runtime of a blockchain is the business logic that defines its behaviour. In Substrate-based chains, the runtime is referred to as the "state transition function"; it is where Substrate developers define the storage items that are used to represent the blockchain's state as well as the functions that allow blockchain users to make changes to this state. Thus, the runtime can simply be thought of as the business logic of the chain. It defines what transactions are valid and invalid and determines how the chain's state changes in response to transactions.

The "outer node", everything other than the runtime, does not compile to WASM, only to native. The outer node is responsible for handling peer discovery, transaction pooling, block and transaction gossiping, consensus, and answering RPC calls from the outside world. While performing these tasks, the outer node sometimes needs to query the runtime for information or provide information to the runtime. A Runtime API facilitates this kind of communication between the outer node and the runtime.

An extrinsic is a piece of information that comes from outside the chain and is included in a block. Extrinsics fall into three categories: inherents, signed transactions, and unsigned transactions. A block in Substrate is composed of a header and an array of extrinsics. The header contains a block height, parent hash, extrinsics root, state root, and digest. Extrinsics are bundled

16

together into a block as a series to be executed as each is defined in the runtime.

Blockchains must agree on:

- Some initial state, called "genesis",

- A series of state transitions, each called a "block", and

- A final (current) state.

In decentralised systems, the nodes will see transactions in different orders, and thus they must use elaborate method to exclude transactions. As a further complication, blockchain networks strive to be fault tolerant, which means that they should continue to provide consistent data even if some participants are not following the rules.

Blockchain nodes use consensus engines to agree on the blockchain's state. It has some internal state, and state transition function that allows it to transition from its current state to a future state. In most runtimes there are states that have valid transitions to multiple future states, but a single transition must be selected.

Some nodes in a blockchain network are able to produce new blocks, a process known as authoring. This is decided by the consensus engine that is being used. For this permissoned blockchain, the consensus method that will be used is Aura (round-robin). Aura provides a slot-based block authoring mechanism where a known set of authorities take turns producing blocks. Further, a fork choice rule is an algorithm that takes a blockchain and selects the "best" chain, and thus the one that should be extended. The longest chain rule simply says that the best chain is the longest chain.

Users in any system want to know when their transactions are finalised, and blockchain is no different. In some traditional systems, finality happens when a receipt is handed over, or papers are signed. Using the block authoring schemes and fork choice rules described so far, transactions are never entirely finalised. There is always a chance that a longer (or heavier) chain will come along and revert your transaction. However, the more blocks are built on top of a particular block, the less likely it is to ever be reverted. In this way, block authoring along with a proper fork choice rule provides probabilistic finality. When deterministic finality is desired, a finality gadget can be added to the blockchain's logic. Members of a fixed authority set cast finality votes, and when enough votes have been cast for a certain block, the block is deemed final. In this system, this threshold is 2/3. Blocks that have been finalised by such a gadget cannot be reverted without external coordination such as a hard fork. The gadget that will be used is GRANDPA or GHOST-based Recursive Ancestor Deriving Prefix Agreement.

GRANDPA validators vote on chains, not blocks, i.e. they vote on a block that they consider "best" and their votes are applied transitively to all previous blocks. Once more than 2/3 of the GRANDPA authorities have voted for a particular block, it is considered final. A diagram of this process is below:
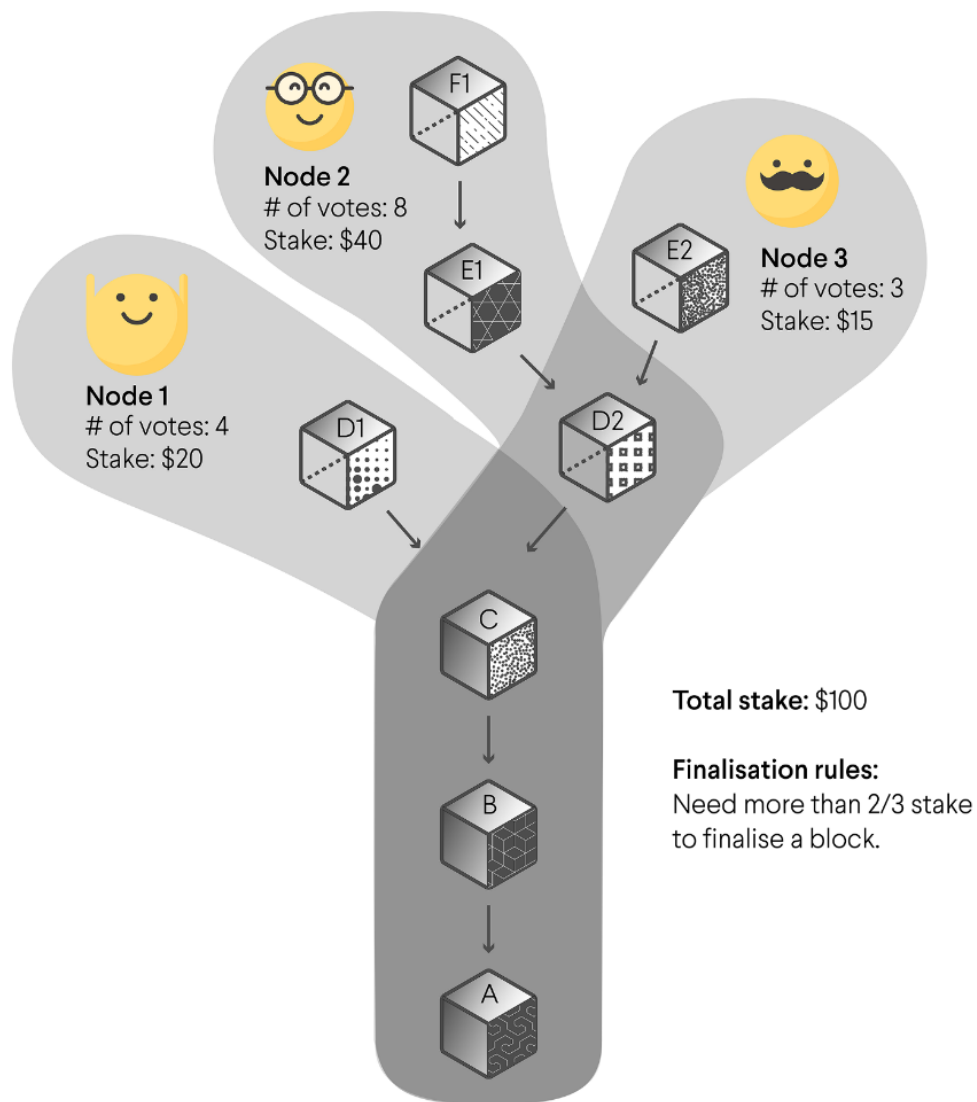
GRANDPA figures out which blocks more than 2/3 of nodes have in their chain, and finalizes them. It can also give different nodes different weights. These weights could be determined by amount staked in the protocol. In this diagram, blocks C, B, and A are finalized.

Functions can be automated through smart contracts, in which lines of computer code use data from the blockchain to verify when contractual obligations have been met and payments can be issued. Smart contracts can be programmed to assess the status of a transaction and automatically take actions such as releasing a payment, recording ledger entries, and flagging

exceptions in need of manual intervention.

Finally, a common architecture pattern for privacy-preserving is to only store a hash of the transaction data. For example, an invoice can be shared without exposing any details of the invoice. This allows companies to trace and exchange information with blockchain technology without exposing any data as the data are kept safely off-chain.

The core algorithms that will be used in the application are displayed below, written in the pseudo-code syntax. The program is split into two sections: the node that will be apart of the network, and the front-end that will be accessed by business's and clients after they have set up the node. For the purpose of the documentation each module will be represented by a single sub-routine, however, in production each module will have multiple files with many sub-routines for design organisation and efficiency.

7.2 Algorithms - Node

7.2.1 Node Main Program

Listing 7.1: Main Program

```
1  BEGIN MAINPROGRAM(node_key, port, ws_port, rpc_port, ed25519_key,
      sr25519_key, bootnodes)
2      services(node_key, port, ws_port, rpc_port, ed25519_key,
          sr25519_key, bootnodes) // This module starts a thread that spins
           up the network, client and extrinsic pool.
3      non_fungible_tokens(rpc_port) // Runtime "Pallet" that handles non-
          fungible token issuing and transference within the business
          blockchain. The RPC channel is the transmission line in that the
          interaction from the front end will go through.
4      fungible_tokens() // Runtime "Pallet" that handles fungible tokens
          and their minting, transference and architecture patterns.
5      runtime() // This function/module handles all aspects of the runtime
           as described above.
6  END MAINPROGRAM
```

7.2.2 Runtime

Listing 7.2: Runtime

```
1  BEGIN runtime(rpc_port)
```

```
2      runtime = new dataStructure with types BlockNumber, Signature,
           AccountId, Balance, Index, Hash
3
4      let static const BLOCK_LENGTH = 5 * 1024 * 1024 // 5Mb
5      let static const SLOT_DURATION = 6000 // This variable determines
           the average expected vlock time that is being targeted, this is
           600 milliseconds or 6 seconds.
6
7      configure_aura_consensus()
8      configure_grandpa_finality()
9
10     configure_transaction_payments()
11     validate_transactions()
12
13     configure_node_authorisation() // This will enable nodes to be
           authorised while the blockchain is running, therefore the
           blockchain does not need to be stopped to allow people to join.
14
15     FOR i=0 to rpc_port.extrinsics().length() DO
16         apply_extrinsic(rpc_port.extrinsics()[i])
17     NEXT i
18
19 END runtime()
```

7.2.3 Non-fungible Tokens

Listing 7.3: Non Fungible Tokens

```
1  BEGIN non_fungible_tokens(rpc_port)
2      let NFT = new dataStructure implementing config trait with fields
           price, owner and proof // Creates the NFT data structure that
           will represent each NFT item in the business blockchain.
3      let Config = new Trait with types event, currency, MaxBytesInHash //
            The config trait that defines the parameters and types that the
           NFT pallet depends on.
4      let NFTs = new StorageMap of NFT // Defines the storage structure in
            which NFT will be stored on the blockchain.
5      let GenesisConfig = new empty NFTs Map // This defines the config
           for NFTs in the genesis block that all nodes must agree on, all
           nodes will start with an empty storage map for NFTs.
6
7      IF rpc_port.recieve == "Create NFT" THEN //rpc_port is the remote
           procedural call that is received from the web interface.
```

```
 8          create_nft(rpc_port.user_recieved(), rpc_port.proof())
 9      ENDIF
10
11      IF rpc_port.recieve == "Set Price" THEN
12          set_price(rpc_port.user_recieved())
13      ENDIF
14
15      IF rpc_port.recieve == "Transfer" THEN
16          transfer(rpc_port.user_recieved(), rpc_port.other())
17      ENDIF
18
19      IF rpc_port.recieve == "Buy NFT" THEN
20          buy_nft(rpc_port.user_recieved(), rpc_port.other())
21      ENDIF
22  END non_fungible_tokens()
```

Listing 7.4: Create NFT

```
 1  BEGIN create_nft(user, proof) //proof here a list of bytes which
         represent an image file, invoice, contract document that will be the
          foundation for the NFT.
 2      ensure_signed(user)?
 3      let nft = new NFT using proof, user and block.
 4      let nft_id = hash(nft)
 5      IF NFTs.contains(nft_id) == TRUE THEN
 6          OUTPUT("NFT already exists and cannot be replicated")
 7      ELSE
 8          NFTs.insert(nft_id)
 9      ENDIF
10  END create_nft()
```

7.3 Algorithms - Front-end

7.3.1 Main Program

Listing 7.5: Main Program

```
 1  BEGIN frontend_main(target_url, api_state, account)
 2      Connect to websocket using target_url
 3      IF api_state == "Error" THEN
 4          OUTPUT "API Error
 5      ENDIF
 6
```

```
 7      IF api_state == "READY" THEN
 8          OUTPUT "Connecting to node"
 9          load_net(target\_url)
10          load_acc(account)
11      ENDIF
12
13      load_frontend_page(api) // API is the data structure that represents
                the api that the front end is interacting with.
14      load_nfts(api)
15      load_fts(api)
16
17  END frontend_main()
```

7.3.2 Display Non-fungible Tokens Module

Listing 7.6: Load NFTs

```
 1  BEGIN load_nfts(api)
 2      let NFTs = api_query_nfts()
 3      let fileReader = new fileReader // A new fileReader object which
            will enable the user to upload files on the web front end and for
             them to be read into a byte array.
 4
 5      let elements = load_nft_ui_elements()
 6
 7      let CreateNFT = new Button
 8      IF CreateNFT.pressed == TRUE THEN
 9          rpc.callable_function("create_nft(user, fileReader.file)")
10      ENDIF
11
12      IF element.transfer_button.pressed == TRUE THEN
13          rpc.callable_function("transfer(user, to, nft_id)")
14      ENDIF
15
16      IF element.buy_nft.pressed == TRUE THEN
17          rpc.callable_function("buy_nft(user, from, nft_id)")
18      ENDIF
19
20  END load_nfts()
```

PROJECT WORK EVIDENCE

In conjunction with my commits on github at the link: Converge Github, I have created a logbook documentation to highlight major milestones in the process of creating the software application and my thought process behind the decisions I made along the way.

8.1 10/4/22

I began the project with the idea of Unison. A software solution that sophisticatedly shares computational load across website visitors to achieve a common goal. Utilising WebAssembly and the Rust Programming Language, Unison efficiently shares computational load and would demonstrate its capability through community cryptocurrency mining. This was the beginning point for this project and was a major milestone as it marked the start of this ever evolving and shifting project.

8.2 27/4/22

At this point in time, I had researched quite a bit into how WebAssembly functions and how one might achieve a sharing of computational load. I was going to create a front-end interface to access a WebAssembly(WASM) binary that would solve computational problems for point of work cryptocurrency mining, the solutions and which cryptographic nonces to check would be transmitted between a user and a central server and thus would be, "sharing computational load." Although this idea interested me and the technology was fascinating, I wanted to move towards a more financial related software project as I did not see a lot of value in what I was currently

creating. This is where I moved onto Converge. This project idea was to sophisticatedly display the global economy using data-driven machine learning. It would provide a macroeconomic view of the economy using sentiment analysis of market news and utilise statistical methods to devise an arbitrary value of economic potential for each country. This was more financial than my current project and I could see how this would be used for financial institutions to see a macro view of each country's economies. I began researching how I would retrieve market information, undergo dimensionality reduction, and then produce an arbitrary value for each economy. I then researched the different web frameworks, such as ReactJS, to build a visually appealing front end to display all this data.

8.3   10/5/22

At this point in time, I had made some progress on my application. Finding how to retrieve data from the market and news sources using web-scraping and certain API's such as NASDAQ's Data Link. Further I had researched how to intertwine WebAssembly and Javascript to be able to develop a system that figures out the arbitrary value of the economy and then display it by communicating to JavaScript. Although this was an interesting project with lots of potential and full of interesting technology such as machine learning, statistical models, WebAssembly and frontend development it deviated from the area that I wished to develop a project for. The project was a lot more economical in nature and converge, "diverged," from the business / financial industry that I wanted to develop for. At this point in time, I was unsure of whether to just continue and finish the project and be not truly satisfied with what I had created. This left me in a state of ambivalence as I did not have another idea to shift to and there was not much time left to be switching ideas.

This is where I bought a car, and unfortunately (or fortunately looking back in hindsight as without it this project may never have been undertaken) I had to pay a stamp duty to change the registration or ownership of the car. This gave me the idea to create a blockchain in which people could transfer assets that were tokenised with the efficiency and security inherent with blockchain technology. Further this would allow buyers to see the full history of the car and its previous owners, a tamperproof history of each car around the world. This project idea included sophisticated, bleeding edge technology, however, it was not for the financial businesses industry that I wanted to create software for. This is when the idea of Converge was created. A decentralised blockchain for sophisticated asset management and cross-business transactions. Utilising an intelligent consensus mechanism and concepts such as tokenisation, converge will efficiently create a complete, transparent, tamperproof history and facilitation of the information flows, inventory flows, and financial flows in transactions. I decided to keep the name Converge as

it represented how the businesses would come together, "Converge" into one business blockchain where they could transact with each other globally, with minimal costs, blazing fast execution and with the utmost security. At this point I began the long process of researching how blockchain technology worked, the complex and intricate nature of how a business block was created, how asset transactions worked, how assets could be tokenised, the legal proceedings and regulations, the consensus mechanisms, the intricate runtime and networking mechanisms and finally how I would connect it all together into a replicable node that formed a global network that could be accessed, from anywhere, on visually appealing, easy-to-use front-end.

8.4   11/5/22

The next day after my idea creation I began to try and build a foundational structure for the blockchain which would take multiple parameters such as block_id, hash... etc. At this point I realised the sheer size of the software application that I was trying to create. I was competing with the biggest firms in the world who have contributed billions of dollars to research: McKinsey, PwC, Deloitte, EY, Accenture, KPMG... I was creating a global network that would connect the top business's together on one business blockchain to facilitate transactions, a network that would become a complete, transparent, tamperproof history and facilitation of the information flows, inventory flows, and financial flows across the globe. Trillions of dollars (in concept) could be flowing through the network at a single point in time. The enormity of the project sunk in and I decided that I wouldn't be able to start from scratch, create all the intricate runtime details, all the peer to peer protocol implementation and networking logic, I needed to work with a framework to actualise my idea.

8.5   23/5/22

This was a major milestone in my project as I found the framework that I would use for my project. Substrate is a blockchain framework that enables developers to build customised blockchains while providing the backbone utility to help it run. It provides abstractions over complex processes such as peer to peer networking, remote procedure calls and WASM Runtime modules. This enabled me to focus on what I wanted to create rather than all the complex, intricate modules that was needed to get a basic, foundational model running.

8.6 8/6/22

At this point in time, I had finished all of my research surrounding how business blockchains operate and how the technology interconnects with each other to create a self-sustaining system. I had also finalised all the intricate details surrounding the product I was creating, including how the tokenisation process would work, how all the business's would connect with each other and how they would share assets.

8.7 10/6/22

This was a major milestone in the project as I added the ability for the blockchain to utilise non-fungible tokens, which will be used to transfer assets between businesses. The assets specifically that businesses will use this token for is things like contracts, consultancy hours, a specific building, or anything that is unique in nature. I also updated the front-end to display these operations and updated the chain specification to be further targeted towards the business blockchain.

8.8 12/6/22

This was another milestone in the development of Converge as I added the ability for businesses to mint and issue fungible assets that represented their physical assets. This was then added to the front end of the blockchain so that businesses could be added. I also added the ability for businesses to be added to the blockchain while it is running through the utilisation of extrinsics. Instead of having to have all the business public and private keys before the blockchain was started, which was infeasible, if business's want to join the blockchain the network does not need to be shut down but they can be added and confirmed by well-known nodes in the network.

8.9 19/6/22

This was another major milestone as I had fixed all errors with the node software and a minimal viable product had been created, although there was some features that I wanted to add to the software solution. At this point I needed to ensure that the front end worked in tandem with the node software and after this was done I could implement more complex smart contracts and possibly zero knowledge proofs if time allowed.