WHITEPAPER

Blockchain

Technology overview and opportunities for Eaton Aerospace

Published by
Eric(Yixin) Feng, Bachelor of Applied Science

Table of Contents

1.0 Executive Summary 3
2.0 Foresight focus on blockchain 5
2.1 what is blockchain 5
2.2 How does it work 6
Transaction Definition 7
Transaction Authentication 7
Block Creation 7
Block Validation 7
Block Chaining 7
2.3 Types of Blockchain 8
Permissionless vs. Permissioned Blockchain 9
Public Blockchain 9
Private Blockchain 9
Consortium Blockchain 9
Hybrid Blockchain 9
2.4 Commercialized Blockchain Application 10
2.4.1 Cryptocurrency 10
2.4.2 Value Registry 11
2.4.3 Value Ecosystem 11
2.4.4 Value Web 12
Smart Contract 13
Domestic & International payment 13
Trade Finance 14
2.5 Future Blockchain Technology 15
Government data distribution 15
Greater transparency between industries 15
Institution-issued cryptocurrency 15
Blockchain supported identity 16
World Economy via blockchain 16
3.0 Full stack architecture to build blockchain applications 16
3.1 Internet Evolution 17
Web 1.0 17

Web 2.0 18
Web 3.0 18
3.2 General Blockchain Stack Architecture 18
3.3 Dependencies 20
Node.js – Development Environment/ libraries 20
Vite.js – front-end 20
MetaMask Ethereum Wallet – crypto wallet 21
Alchemy – Node provider 21
HardHat – Smart contract development environment 21
3.4 Project Setup 22
3.5 Build Smart Contract 25
3.6 Deploy Smart Contract 26
3.7 Client-side Application 27
3.8 Backend function 29
3.9 Frontend interfaces 30
3.10 Deployment 31

1.0 Executive Summary

Over the past two decades, rapid advances in internet technologies and wireless connectivity have impacted almost every aspect of our daily lives, changing the ways in which we work, learn, shop, and even interact with one other. Originating from an obscure whitepaper back in 2008 authored under the name Satoshi Nakamoto (a presumed pseudonym), blockchain is now widely viewed as one of the most prominent new technologies with the potential to unlock a new era of internet where data transfer is secure and happens in real time. Blockchain's promise is to provide a decentralised web environment and create a trustworthy network, where everyone can access information and trade assets safely.

While still in its early stages, the blockchain market is expected grow rapidly over the next decade, permeating through almost all key market segments including financial, retail, government and industrial, and is projected to reach a market size of close to $2 trillion by 2030. Potential use-cases for blockchain include any areas that require trade, digital registry, and/or creating a value ecosystem.

In this paper, we will dive deep the underlying technology of blockchain, discuss its potential, and show how blockchain can be used to develop a decentralised app. A sample application will be demonstrated to illustrate the full stack architecture of a blockchain application consisting of a crypto wallet, front-end interface, node provider, smart contract, and blockchain network. Lastly, we will provide examples of how blockchain is being utilized within the Aerospace industry and hypothesize potential use-cases where Eaton Aerospace could derive value through the utilization of blockchain technology.

2.0 Foresight focus on blockchain

Today, with the widespread advancement of internet connectivity across every industry, data and information trade has become incredibly easy and fast. However, most transactions are not real-time because trust between parties requires validation from central authorities, so there is a lack of a real-time online trade method to reach the optimal state of the process. To solve this problem, a new and potentially disruptive technology known as blockchain has emerged. The core of this technology is to create a distributed consensus ledger, where the ledger is maintained by a distributed network of computers. Blockchain technology is able to create new possibilities by letting individuals create, evolve, and observe immutable transactions and other successive events. Today there are already countless applications being powered by blockchain that are having a significant impact on numerous industries including credit, cryptocurrency, medical safety, real estate trading, identity protection, etc. Further proving that blockchain will likely be the future way to transfer information on the internet.

As current enterprises gradually integrate with blockchain technology, the technical complexity and the lack of acceptance of these far-reaching changes has prevented the development in private, public and commercial sectors. Therefore, it is less a question of whether blockchain will be used in the future but more a question of where and how it can be integrated efficiently.

This document will provide an informative explanation into the fundamentals of blockchain technology, the current state of the technology and how this technology will impact multiple industries in the future. A sample blockchain application will also be explored to demonstrate a typical technology stack and the work needed to deploy the application into the blockchain network. Lastly, the integration of this technology will be discussed by examining use cases showing how blockchain is used to enhance the current state of the online trade system.

2.1 what is blockchain

Blockchain is a globally distributed database which can facilitate transactions of assets across the world within seconds while incurring only a minimum transaction fee. These assets are usually represented in digital value and are validated using a smart contract.

One key difference between typical database and blockchain is that data is structured in groups, known as blocks. Each block can store sets of information and once created, it will be validated and linked to previously filled blocks. In addition, the traditional database stores data into tables, but blockchain stores data into blocks. Each block inherits its properties from previous blocks with a unique timestamp. Below is an example of blockchain architecture.

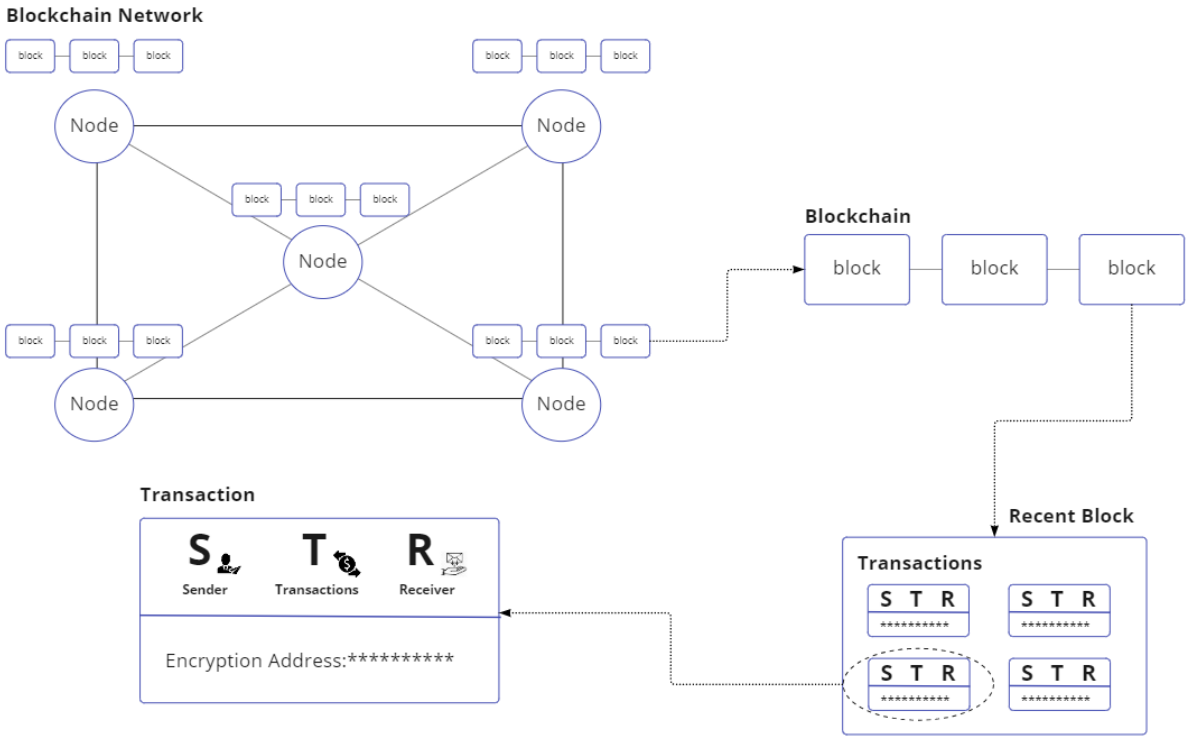


Figure 1. Simplified graph for a distributed ledger network, each node is a machine that tracks the history of all blocks created on the network. Each block consists of a set of transactions and each transaction provides the fundamental information about the sender, transactions, and receiver as well as the encryption address that locates the transaction inside the network.

Blockchain helps to solve the "double problem", which is related to digital currencies being viewed as unfeasible properties due to the fact that copies of information can be easily obtained and being difficult to prevent digital money being sent to the merchant while the owner still holds the original copy. Therefore, the traditional ways of mitigating this risk are usually to hire a third-party authentication service, such as banks, to act as a centralized authority keeping the safety of all transactions. Bitcoin, one of the first blockchain driven technologies developed, resolved this "double problem" by creating a distributed ledger to keep the history of all of transactions and require validation from its users to verify each chain inside the blockchain.

2.2 How does it work

The goal of blockchain is to allow digital information to be created while preventing any forms of alteration of data and being able to distribute the information to all stakeholders. Over the last decade numerous applications utilizing blockchain have been created such as cryptocurrencies, decentralized finance applications (DeFi), non-fungible tokens (NFTs), and smart contracts. [1] Below, we provide an overview of the key steps involved in the process of a blockchain transaction.

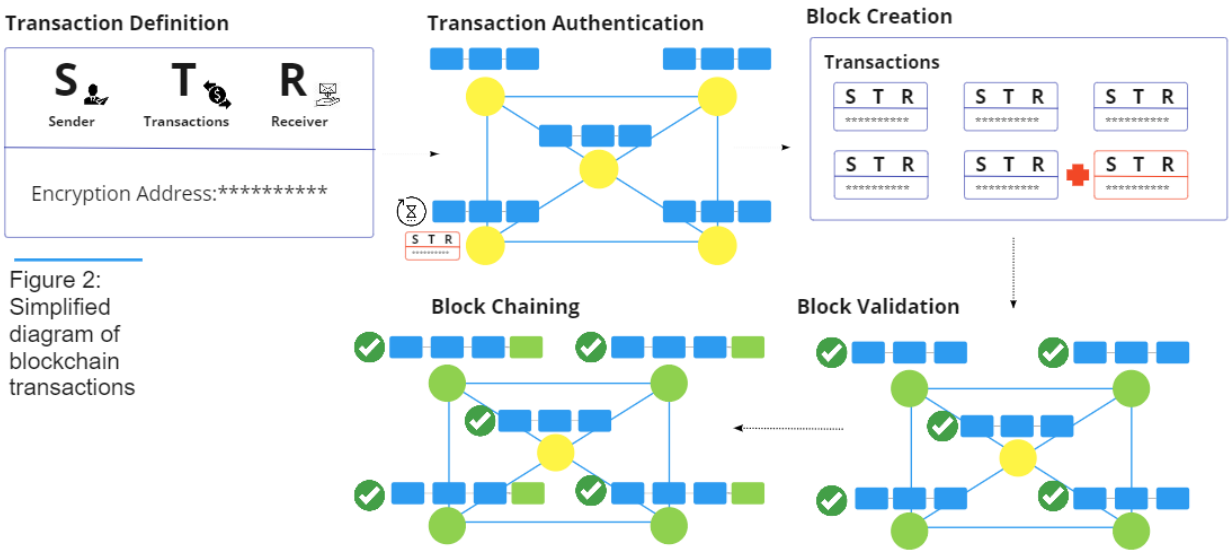


Figure 2: Simplified diagram of blockchain transactions

Transaction Definition

The sender creates a transaction on a local node(machine), which typically includes basic information such as the sender identity, the transaction amount, any forms of irreversible data, and the receiver's address. Each transaction is assigned a unique address to locate the transaction within the network for when it gets added to a block in a subsequent step.

Transaction Authentication

The node (machine/user) receives the message of the block creation and authenticates the action. The authenticated transaction moves to a pending state among a "pool" of other pending transactions.

Block Creation

At a specific time interval, the node broadcasts all pending transactions to the network for validation, and an updated version of the ledger is created.

Block Validation

The validation is done in an iterative process that requires the consensus of the majority of participating network nodes. There are different ways to validate the block. For instance, Bitcoin's block chain uses a validation technique called "proof-of-work", while Ethereum uses a technique called "proof-of-stake". The different techniques each have their own advantages, but all in all they provide security and make sure each transaction is valid and authenticated through the distributed ledger.

Block Chaining

After validation, the new block in added to the block chain and broadcast to all the nodes in the network. This "chaining" process will usually take between 3-10 seconds.

2.3 Types of Blockchain

The initial idea of blockchain revolved around avoiding central authorities, and it has become a success because everyone can share data safely and anonymously without accessing the feature behind a paywall. This type of decentralized, publicly accessible network is referred to as a permissionless ledger. However, some organizations saw the opportunity to transfer company internal data while giving the validation authority to trusted individuals, in that way, nodes can be assigned to people with authenticated status, whiles others can still initiate and receive transactions in the blockchain. When developing blockchain applications, companies need to understand the trade-offs and choose type of blockchain that best fits their needs and complies with existing laws and regulations.[2]

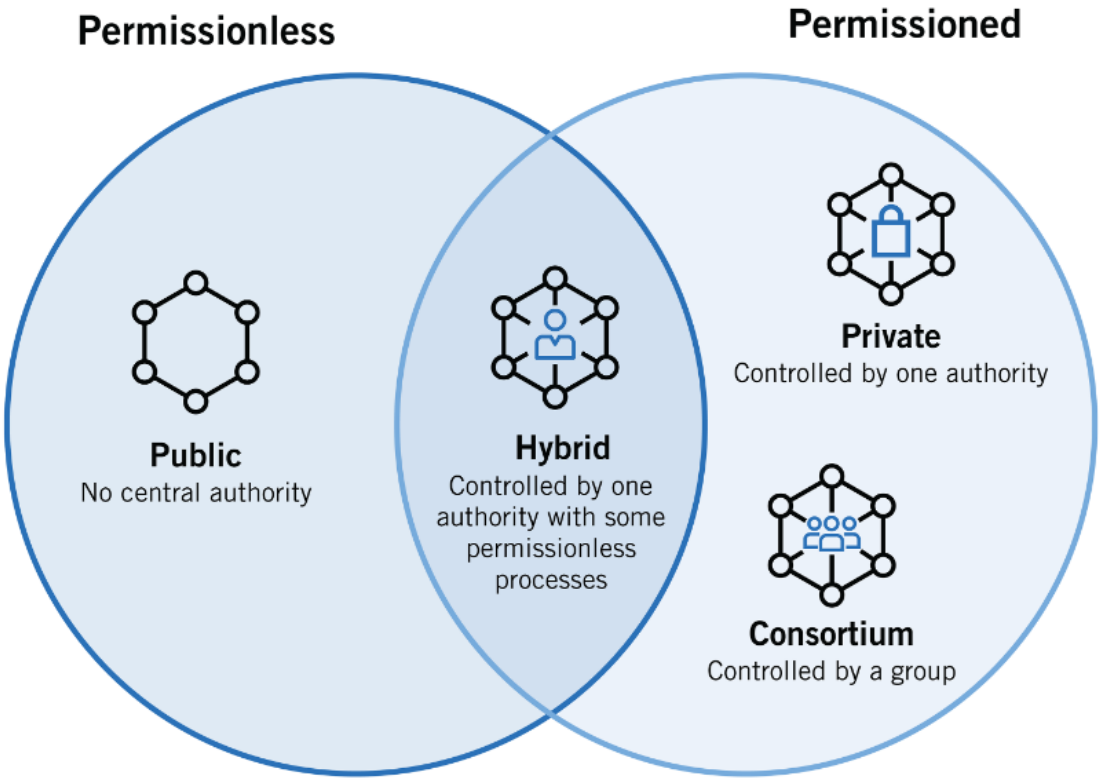


Figure 3. relationships between permissionless blockchain and permissioned blockchain, and how it evolves into public, hybrid, private and consortium blockchain based on who is able to validate the transactions and amount of action each node can take. [2]

Permissionless vs. Permissioned Blockchain

Blockchain can be categorized as permissioned, permissionless, or something in between. With a Permissionless blockchain any user is can interact with the blockchain anonymously and can receive, send, and validate transactions.

On the Other hand, permissioned blockchain will assign levels of accessibility to nodes in the network and restrict permission to join the network to specific nodes. Nodes in permissioned blockchain are also transparent meaning that anyone in the network will be able to know the identity of other nodes in the network.

Permissionless blockchains tend to be more secure since there are more nodes in the network to validate transactions. However, the processing time will be longer because of the large number of nodes and size of the transactions. In contrast, a Permissioned blockchain will have less nodes, meaning it will have faster processing time but will also be less secure because it is easier for bad actors to collude. Therefore, a private blockchain network must ensure all nodes inside the network are highly trusted and authenticated.

Public Blockchain

Public blockchain allows anyone to join and becomes completely decentralized. Anyone inside the network has equal rights to access the blockchain, create data, and validate data.

Currently, most public blockchains are used to exchange and mine cryptocurrency, such as Bitcoin, Ethereum, Litecoin, etc. On these public blockchains, the nodes "mine" for cryptocurrency by creating blocks for requested transactions through solving cryptographic equations. In return they are rewarded a small amount of cryptocurrency. These "miners" essentially act similar to bank teller that formulate transactions and receive "fee" for their efforts.

Private Blockchain

In private blockchain, the central authorities decide who can be a node in the network. Each node will also be granted limited access as determined by the central authorities, so it is partially decentralized where nodes in the network has restricted functionality. Some examples include B2B private blockchain networks such as Ripple and Hyperledger.

Consortium Blockchain

Consortium blockchain are governed by group of nodes, rather than one central authorization. This type of blockchain provides more decentralization and results in higher levels of security. However, these blockchains are underdeveloped and require complicated processes to integrate into current enterprises. Some challenges might include cooperation between organizations, which includes logistics and antitrust risks. In addition, shift the current infrastructure to implement blockchain technology requires significant time and cost.

One recent example of a consortium blockchain is the Global Shipping Business Network (GSBIN) originally formed by CargoSmart. The GSBIN is comprised of nine leading ocean carries & terminal operators working collaboratively with the goal of facilitating the sharing of verified logistics data and streamlining operations across the entire supply chain.

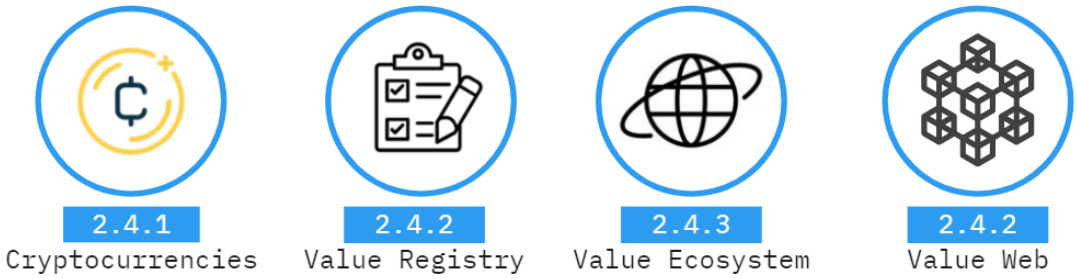
Hybrid Blockchain

hybrid blockchains function like private blockchains, which have a central authority that controls access to the network, but once authorized, all nodes have equal rights to create and validate data. Some examples of hybrid blockchains include IBM Food Trust and XinFin developed by Ramco Systems.

2.4 Commercialized Blockchain Application

When the first blockchain applications were developed, known as blockchain 1.0, they were used as a means to provide a standard digital currency. As the technology has matured, the focus has shifted to more advanced ownership and transaction authentication, known as blockchain 2.0, this technology starts to migrate physical contracts to smart contracts and executes orders once predetermined conditions are met. While blockchain technology is slowly matured for commercial use, more enterprises are starting to build services and enhance their transaction processes using blockchain.

This section will be an informative analysis into the current opportunities to implement blockchain and deep dive into how banks and payment industries integrate this technology.



2.4.1 Cryptocurrency

Cryptocurrency is a digital representation of currency that is not issued by central banks or any public authorities but is being accepted by two or more parties as a mean to exchange, store, and transfer electronically.

Cryptocurrencies create a trading system that provides the consumer and merchant necessary function to make trades. These payment transactions are typically completed in a matter of seconds (permissionless) or minutes (permissioned) regardless of the geographic distance. Another benefit blockchain provides is that since all transactions are completed in a direct manner, any service fee from third player can be neglected. Providing a cost-saving approach to make any transactions.

Bitcoin, first developed in 2008, is currently the most popular cryptocurrency, having the largest market capitalization, an estimated 106 million bitcoin owners, and 270,000 daily bitcoin transactions. [3]

While continuing to grow in popularity, Cryptocurrencies still have many challenges to overcome before gaining widespread, public acceptance. These include:

- High volatility for the worth of the currency leads to fluctuating value over time
- Risk of deflation and inflation cannot be controlled, no stability of tangible asset
- Lack of authorities and policies.

2.4.2 Value Registry

As blockchain matures, the development shifted from handling cryptocurrencies into using public ledger to handle the ownership of tangible assets. This shift in development is also known as blockchain 2.0.

In many financial and legal processes, Proof of ownership is usually validated using any form of signed documents. Relying on a central authority creates risks related to transferring documents, inadvertent disclosure of documents, and deterioration of the documents.

By integrating blockchain technology into the existing ownership system, the user can simply store the document, with the associated signature and timestamp, and be able to retrieve from the blockchain anywhere anytime with its decentralized nature. The owner of the assets can then prove their owner ship by not only the signature, but also the private key tied to the public record that is created when the transaction is made. Thus, providing a secure and easily accessible way to store any types of digital goods or services, as well as contract, password and many more items that can be stored online.

Currently one of the most popular applications of value registry is via a Non-Fungible Token (NFT), which includes any form of digital asset such as art, music, videos, ownerships that can be bought and sold online. NFTs were first created in 2014 and have since gained a significant amount of popularity with the rising demand of online content. The market for NFTs was worth $15.7 billion USD in 2021 and expect to reach $122 billion USD by 2028.[4]

Essentially, NFTs are similar to a collector's item, but in digital form. Each NFT, meaning each digital assets are tied to a single user and they holds the private key for that asset to access detailed information. In addition, they can use blockchain's mechanism to verify and transfer ownership easily. Creator can also store specific information in a NFT's metadata, this could include signature, regulation policies, etc.

This is currently the most potential commercialized usage for blockchain application. This decentralized nature helps customer easily trade digital assets without any central authorities, so everyone in the network can participate and start their own business, significantly reduce the entry difficulty.

2.4.3 Value Ecosystem

Blockchain value ecosystems are blockchain platforms that allow all firms and services to perform rapid integration of blockchain technology using its distributed ledger in the ecosystem.

Blockchain is a technology with limitless potential because of its unique way of storing data and the innovative ways to secure data without a central authority. Therefore, a group of developers aims to

create a flexible blockchain framework to allow other developers to use their imagination to create any decentralized blockchain application in the network. These systems encompass governing structures like individual participation, data ownership, funding, exit and entrance criteria, and information shared within the networks. Essentially, these ecosystems provide a fast development platform to accelerate adoption of blockchain technology in the current market.

One of the most popular public blockchain ecosystems is called Ethereum. It allows developers to set up infrastructure and services on the network, and each service can interact with other services in the same network in a seamless manner. Therefore, Ethereum aims to provide a platform powered by blockchain where firm can build their own service and market on the platform, which is similar to current ecosystem such as eBay, Amazon, Facebook who provide the tools for everyone to start their own business, where more firms can catch up the trend to integrate blockchain into its services and applications.

Ethereum prioritise the usage of smart contract and provide a simple way to program smart contract using solidity. Smart contracts act as a means to execute transaction protocol once specific, pre-defined conditions are met. Essentially the parties involved would agree on terms and set-up the smart contract accordingly. Once the conditions are met the smart contract would automatically proceed with the transaction, eliminating the need for any intermediary. In the future, Ethereum aims to provide a decentralized system, where anyone can create their own service on the platform, run maintenance, and prevent other entities from censoring the content or usage.

2.4.4 Value Web

Blockchain value web is collection of service that focus on trade of financial assets between different banking and payment domains. This is by far the most prevalent area of investment and could change regulation of finance and trade in the future.

The value web revolves around the idea of “Internet of Value”, which is potentially the next evolution of internet that integrates a combination of technologies into the current trade of all forms of assets. Blockchain will be an important key to enhance existing trade process, changing both consumer and financial institution into a new era.

Some of the key benefits that blockchain brings to the value web include its abilities to accelerate process, simplify transaction complexity, and increase security. As a result, many existing financial services are trying to integrate blockchain services to offer systems that provide lower cost, better quality and faster time to market.

There are two main players in value web, which include the trade system managed by authorised financial institution and the people who make the exchanges with these systems. Trade systems will run in permissioned ledger, meaning that access is restricted until authentication. Nodes in the network can publish digital representations of physical assets, such as U.S dollars or gold with the purpose of creating a “bridge” between the digital and physical world. This is expected to be the responsibility of all financial institution as they hold a strong trust in society.

The second player in the web is the "market maker", who is responsible to exchange one digital asset for another. An example would be a forex trading institution exchange between different currencies.

There are five main use-cases for the value web, described in the context of applicability for financial institutions, which include smart contracts, domestic payments, international payments, and trade finance.

Smart Contract

The definition of a smart contract describes a computer program that can automatically execute the parameters of a contract once certain conditions are met. Due to its self-executing natures and the fact that -ownership information is embedded in its system, a smart contract becomes a self-sufficient tool to provide a transparent process to execute contractual agreements without requiring the upfront effort to build trust between both parties.

Another benefit of using smart contracts is that it allows the easy generation of contracts due to its code-based nature. Simplifying the contract drafting and negotiating processes. In addition, machine learning can also gradually be implemented into the smart contract to add the ability to debug the contract logic, which could ultimately be used to alert the drafter of any inconsistencies that would otherwise be ignored by human drafters.

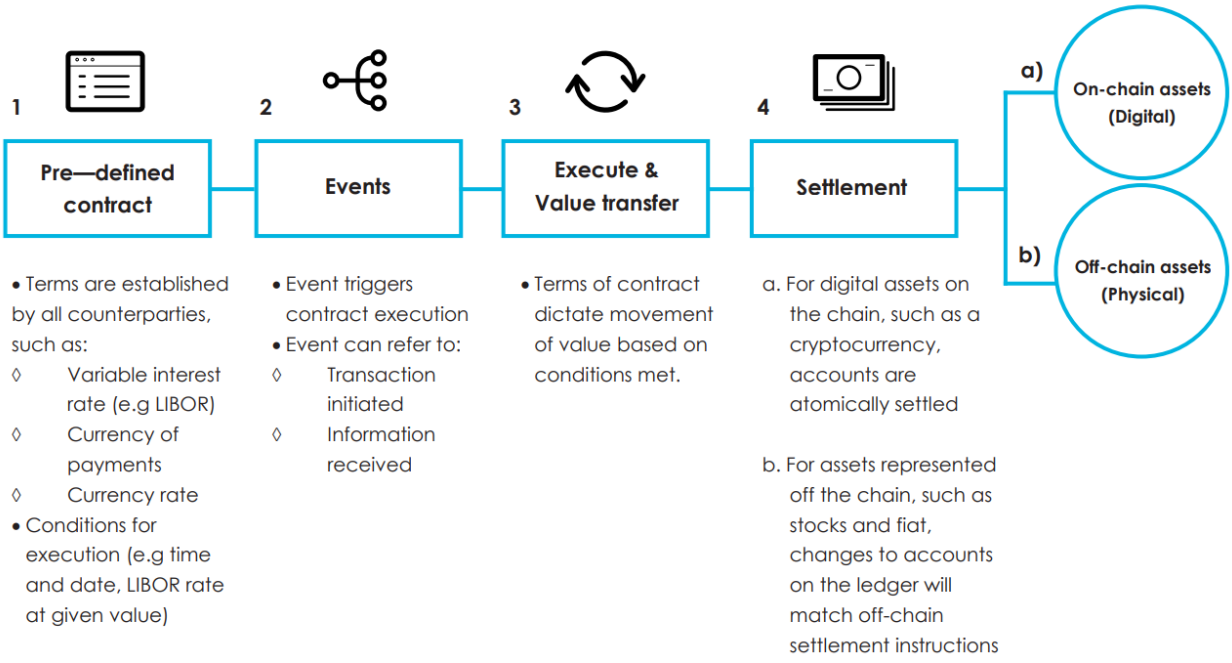


Figure 4: smart contract workflow[7]

Domestic & International payment

The use of blockchain for domestic & international payments has the potential to increase economic growth as faster transaction speeds enabled by blockchain technology will reduce the time associated

with the cash conversion cycle, generating greater working capital and reducing the need for short-term financing. Faster transactions will also improve client experience and prevent competitive threats from new entrance. Therefore, financial institutions are racing to develop instant payment solutions in order to achieve competitive advantage. However, integrating blockchain is difficult because most financial systems are built around complex IT infrastructures. In addition, at the procedural level for transactions, the process of inter-bank clearing is required to have an intensive coordination of resources between commercial banks, clearing houses, and central banks. This process usually takes place few times a day, which is vulnerable to changes in foreign exchange, changes in interest rates, and changes in regulations, especially over weekends and holidays. Therefore, payment service provider is looking for a solution for real-time payment both domestically and internationally.

Blockchain is a direct solution to providing real-time payment between commercial banks, and central banks. This transaction would occur through the use of a digital asset, typically cryptocurrencies. Blockchain would also allow central banks to leverage big data analysis to more effectively balance and manage cash flow in real-time., as they can now leverage big data analysis more effective with real-time cash flow. These technologies are being integrated gradually by financial institutions and it is estimated, based on a World Economic Forum (WEF) report, that by 2025 10% of global GDP will be stored on the blockchain.

Trade Finance

Currently trade finance involves the logistics and handling many types of physical documents as well as the use of several different IT systems. These procedures are similar between banks but typically use completely different IT systems. Example includes:

1. Extension of credit to customer
2. Informing credit status
3. Banks open communication channel to customer
4. Updating status of goods
5. Execution of full or partial payment based on contract

Blockchain would help automate the trade system by using cryptographic keys and crypto wallets, to replace all of the traditional documents and store them on the blockchain as a smart contract.

Cryptographic keys are pieces of information, usually a combination of words, that are used to access blockchain transactions. It comes with a pair of keys corresponding to a public and private address for the transaction. The public key acts as a signature to validate the transaction, whereas the private key will provide access to the digital asset in the network.

Crypto wallets are services that store cryptographic keys without storing any actual digital assets. Part of the service the crypto wallet also provides is to ensure the involved parties reach a desired consensus before proceeding with a transaction by requiring multiple signatures.

2.5 Future Blockchain Technology

Based on a recent Blockchain market size report, blockchain activities are forecasted to reach a value of highest value of $10 billion in 2022 and grow to almost $2000 billion by 2030. [5] The main reason for this rapid growth is due to the efforts being made within the financial industry to capitalize on this highly promising technology. At the current projected growth rate, blockchain will soon overcome other technological development (including cloud computing, data analysis, IOT) in terms of venture capital activities and reach a level of popularity similar to other prominent topics such as artificial intelligent and robotics. As a result, continuing rise in both investment and interest, we can expect more enterprise and financial institution are starting to build services around blockchain and changing the infrastructure for multiple industry including finance, commerce, supply chain, cybersecurity. [6]

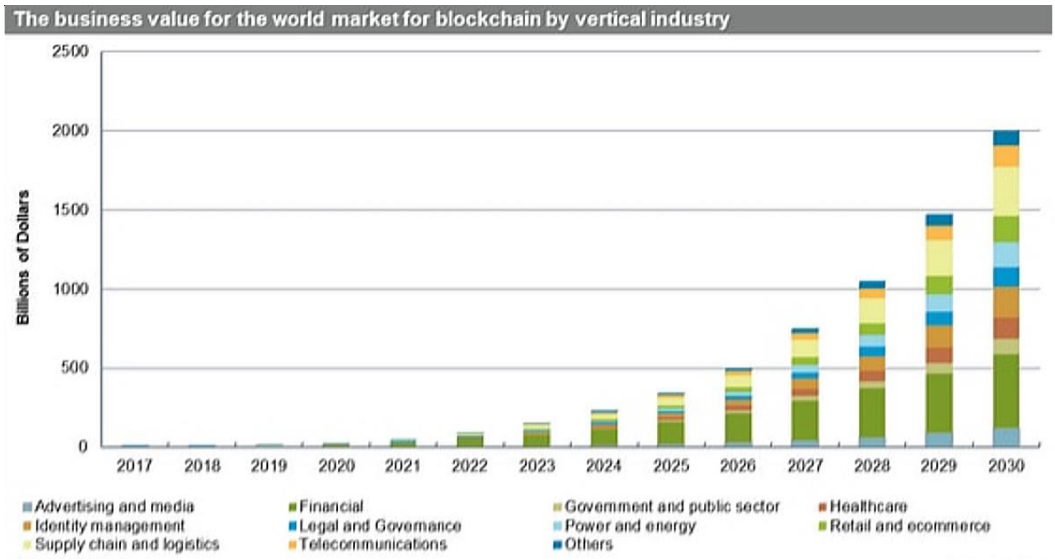


Figure 5: forecast of market size for blockchain to 2030[8]

It will only be a matter of time before blockchain becomes more integrated into people's daily lives. When considering the future of blockchain, there are 6 key areas where it will likely have the most impact including government data distribution, transparency between industries, institution-issued cryptocurrency, blockchain suppled identity, and replacement of infrastructure for world economy.

Government data distribution

Traditional paper-based systems will eventually be migrated to distributed ledger technology (DLT), where DLT provides greater trust, transparency, and security through encryption and validation functions.

Greater transparency between industries

Blockchain will allow all data to be shared through distributed legers in the same network, significantly reducing efforts to interconnect different tools. This interconnected network of distributed ledgers, will make data more accessible to the public, provide greater transparency, and offer the inherent security of blockchain.

Institution-issued cryptocurrency

Cryptocurrency will be a great alterative to flat cash as it can be more traceable, has reduced settlement time and overall is more efficient.

Like flat currency, cryptocurrency can also be backed by real assets, and its worth can be manipulated through various controls. This

prevents practice of printing money to deflate the value of flat cash, and overall provide an accurate worth of the currency.

As of today, central bank such as Bahamas, Nigeria, Grenada, who has already integrated blockchain technology to its digital currency to act as its country's flat currency, also know as central bank digital currency (CBDC). CBDC simplifies the implementation of monetary and fiscal policy which include a central bank's activities related to influencing the quantity of money and influencing government's decision about taxation and spending.

Blockchain supported identity

Current identity management methods have numerous flaws. As many nefarious actors looking to steal other's identity and private information, blockchain becomes a perfect solution to transform online identity management. It offers a tremendous level of security because of its unique way of validating information stored on each block. Validation process will approve a transaction before it is added to the chain, so identity verification and other verification procedures can be easily applied.

There are numerous examples of potential applications where utilizing blockchain for identify management can provide real benefits. One example would be the use of blockchain to aid in maintaining voter information and providing tools to complete electoral process.

Blockchain could also be used to maintain and record real estate ownership, titles and more due to its nature to store and transfer data across different systems as long as it is in the same network.

World Economy via blockchain

Currently, international trade practices are inefficient. These inefficiencies slow down commerce and break trust between nations. International trade is also frequently impacted, by fraud, counterfeiting, changing policies, and language misinterpretations.

By integrating blockchain into the trade process, methods of payment will be unified, and paperwork will be replaced with digital smart contracts. These features can help to reduce fraud while also improving transparency for the status of goods. The reduction of fraud and increased transparency will ultimately lead to an increase in international commerce and foster greater trust in international trade.

3.0 Full stack architecture to build blockchain applications

As blockchain technology matures, more value ecosystem and blockchain specific frameworks/programs are being developed, making it much easier for developers to build blockchain applications and bring blockchain innovations to different industries. In this section we will provide an

overview of the popular concepts, stacks, frameworks, and tools used to build and deploy blockchain applications.

3.1 Internet Evolution

The evolution of the world wide web is generally partitioned into successive iterations that mark major shifts in technology, content and/or use. The earliest version of the web, Web 1.0, was developed in the 1990s and consisted of mainly static pages connected by a system of hyperlinks and hosted on centralized servers. Web 2.0 is the most current evolution of the web and introduced a more dynamic, participative & social web that includes everyday web tools such as Facebook, twitter, google drive, amazon, etc. The next generation of the web, Web 3.0, is just now starting to come into realization and is promised to provide features including decentralization, govern by public, Integrated machine learning and limitless connectivity.

The following subsection contain further details on each stage of web evolution introduced above as well as the role blockchain will play in Web 3.0 and beyond.

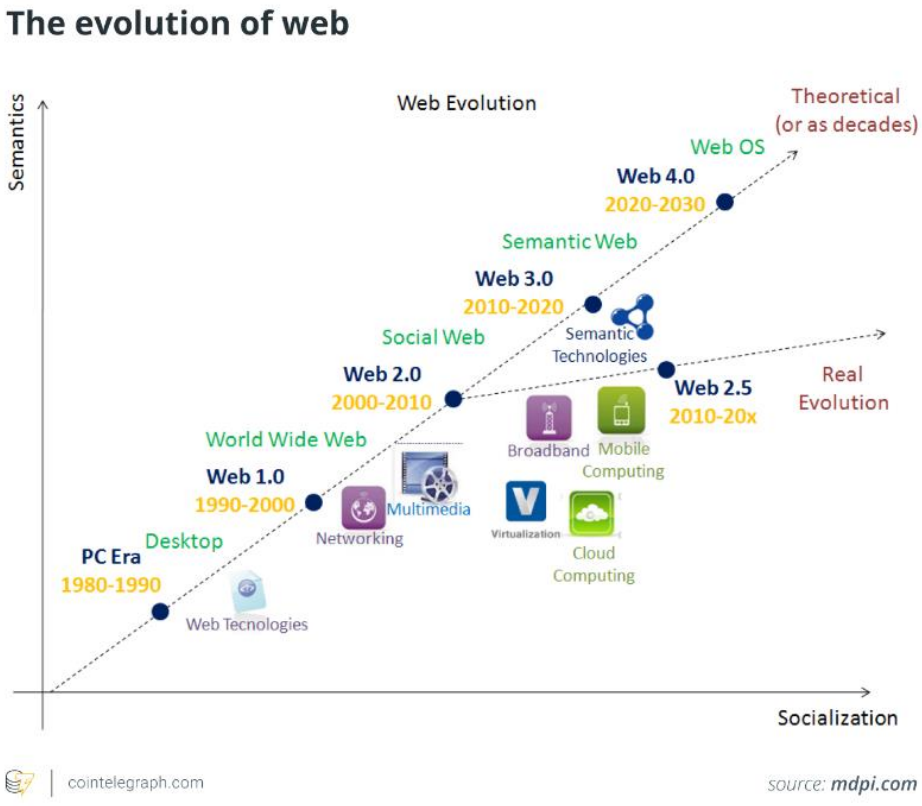


Figure 7: Evolution and prediction of web from 1980 to 2030. Current web 2.5 integrates broad band, mobile computing, virtualization and cloud computing, and Web 3.0 is going to integrate semantic feature in the future. [32]

Web 1.0

Web 1.0 is the foundation for internet and is first developed in 1990 by Bernner-Lee at European researcher CERN. Web 1.0 has three fundamental pieces, which include:

- HTML: a markup language to formalize the web.

- URL: stand for Uniform Resource identifier, a unique address to identify resource on the web
- HTTP: Stand for Hypertext Transfer Protocol, which allows access to specific web on the internet

During this time, web development was still at its infancy, but web applications started to integrate features such as email, real-time news retrieval, and helped with online banking.

Web 2.0

Web 2.0 refers to the shift in interactivity, social connectivity, and user generated content offered through dynamic web applications. This means that nowadays user-generated content can be shared and viewed by millions of people in an instant.

The innovations of Web 2.0 have also driven the growth for mobile internet access and social networks. This development has greatly expanded online interactivity and enabled Web 2.0-centric companies such as Apple, Amazon, Google, Facebook, and Netflix (AKA FANNG), to grow into some of the largest companies in the world by market capitalization.

Web 3.0

Web 3.0 represents the next evolution of internet that builds upon the core concepts of decentralization, openness, and greater user utility. Web 3.0 focuses on 2 main features:

- Decentralization: create a platform where posting data becomes permissionless, and there is no central node to censor or survey information.
- Bottom-up Design: Instead of code written to output data that is used by small number of experts, a bottom-up design is used to make individual users participate and experiment with the code, while also creating a semantic web to make internet data machine readable.

However, many web applications still need restriction and censorship, and it is very expensive and monumentally difficult to convert human language to machine readable language with its different purpose and variation. In addition, Web 3.0 will require the reinvention of the way we store data over the internet in order to provide a universal layer. For these reasons blockchain, along with artificial intelligence and machine learning, are viewed as key areas of technological innovation necessary to enable the Web 3.0 evolution.

Web 2.0 vs Web 3.0

	Web 2.0	Web 3.0
Target reach	Community development	Empowering individual users
Focus	Tagging and end-user experience	User empowerment through trust, security and privacy
Types of applications	Web applications	Smart applications (based on ML and AI)
State of data	The network owns the data	Entities have ownership over the data, including its ownership and use
Driving technologies	AJAX, Javascript, HTML5, CSS3	Artificial intelligence, machine learning and decentralized protocols
Advertising	Interactive advertising	Behavioural advertising
Use of 3D graphics	No	Yes

cointelegraph.com

Figure 8: key differences between Web 2.0 and Web 3.0[33]

3.2 General Blockchain Stack Architecture

In a traditional Web 2.0 application, the user interacts with the applications frontend via a browser, while the frontend inputs translate into queries (such as json) and send to the backend APIs. The

backend performs necessary actions and outputs responses to the user while also storing relevant data into databases.

In Web 3.0, the full stack includes crypto wallet and client-side application as the front end, smart contract as the backend that is deployed on the blockchain network, and a node provider to connect smart contract and the client-side application. The frontend behaves somewhat the same, but with the added feature of being able to identify the user inside blockchain network by interacting with crypto wallet. Then, node provider is connected to frontend that serve as a mean to provide an API key to connect to the blockchain network. It will locate the smart contract that is deployed on the network, where the smart contract is written in programming languages such as Solidity, and act as a backend API. Once transaction is completed, the transaction detail and state changes is stored on blockchain using cryptographic verifiable method.

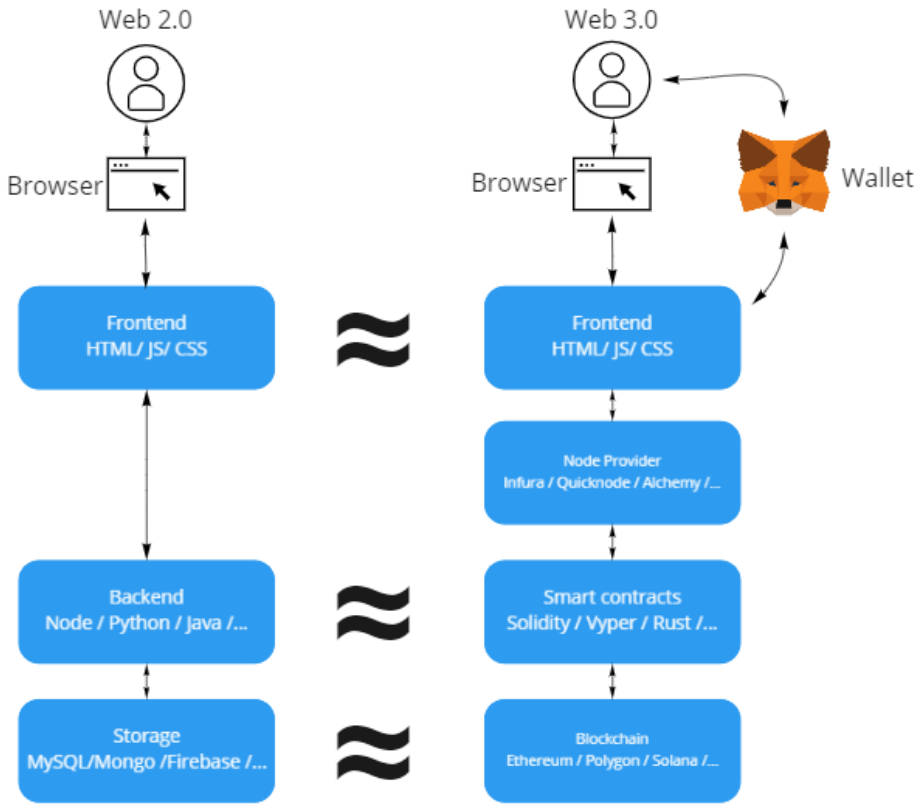


Figure 9: comparison between Web 2.0 and 3.0 stack architecture

Smart contract is used to handle most data transfer using blockchain. They are completely decentralized and publicly accessible compared to traditional backend APIs. Meaning that anyone can see the code, use the code, and add on top of the code, which makes smart contract very accessible and composable to publics. In addition, each transaction is completely immutable once verified, and contains metadata on specific requests, giving transparency to the public and simplifying data collection. Experts can then utilize this data to implement machine learning and open up different

opportunities to bring insight to all types of incoming data, provide rapid decisions and bring more business.

3.3 Dependencies

Before starting the development of a blockchain/web 3.0 application, several dependencies will need to be installed that will help to accelerate development time. Dependencies are essentially different module build for specific function (in this case a blockchain web application) that provides opensource framework and functions instead of building everything from scratch. [9] Below are the potential dependencies used to build blockchain application stack.

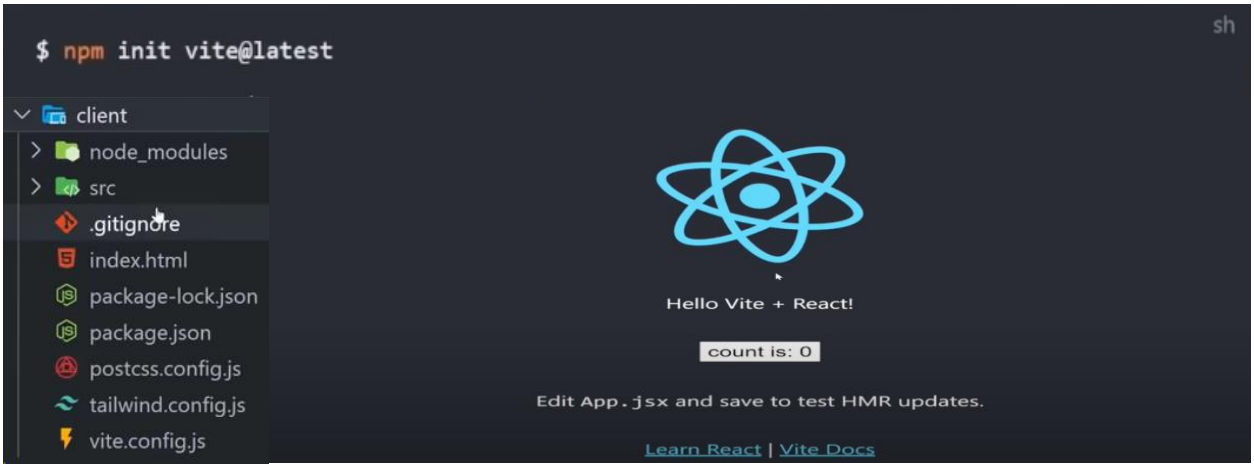
Node.js – Development Environment/ libraries

Node.js is an asynchronous event-driven JavaScript runtime used to build scalable network applications. One of the benefits for using Node.js is that it provides popular frontend/backend framework that automatically sets up sample websites and allows other well-built dependencies to be integrated. By downloading Node.js, we will gain access to "npm", known as the node package provider, which is the world's largest Software Registry, containing over 800,000 code packages. [10]

Vite.js – front-end

Vite is front-end framework-agnostic written in JavaScript. For instance, it offers official templates for React, Vue, Preact, Svelte, Lit and even vanilla JavaScript and TypeScript. [11]

VIte.js can be accessed using "npm", this framework will be set up once running script in the terminal.



Sample web application will start running after deployment from "npm" to the local machine, and user can add additional front-end component and change web behavior in "src" folder.

MetaMask Ethereum Wallet – crypto wallet

Metamask is a web extension that turns a normal web browser into a blockchain browser. This type of wallet will store the public and/or private keys for cryptocurrency transactions, allows the web to connect with a public blockchain network, and stores a user's personal information. The public key is where anyone can get the history for the transactions and the private key is where a user can access the actual digital asset on the blockchain network and is encrypted into large, randomly generated numbers with hundreds of digits. [12]

Alchemy – Node provider

Alchemy is a node provider that connects local blockchain applications to the public blockchain network. Once a user is set up the application and chooses a network to connect to, Alchemy will provide a key in HTTP form to allow the blockchain application to be deployed in the network.

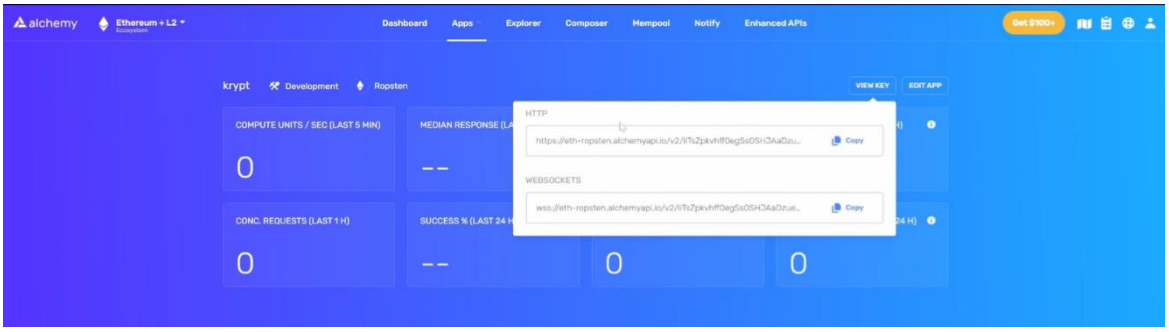


Figure 10: an example of setting up a blockchain application in alchemy. Here the application is connected to the Ropsten test network and the HTTP is used to deploy the application developed in solidity smart contract on this specific location within the network

HardHat – Smart contract development environment

HardHat is a development environment that provides framework to develop solidity smart contracts and provides functions to compile, deploy, test, and debug Ethereum software. It also helps the developer manage features that are inherent to developing dApps and smart contracts.

Below is an overview of all the functionality provided in the HardHat development environment

- Smart Contract Management – write smart contracts with solidity programing language and compile into byte code to feed into Ethereum Virtual Machine (EVM)
- Contract Deployment – ability to migrate and deploy smart contracts onto any public Ethereum blockchain network
- Network Management – ability to connect to any public Ethereum blockchain network and the ability to develop in a personal blockchain network

- Development Console – interact with smart contracts inside the JavaScript runtime environment. It can connect & provide interactions between various blockchain applications and client-side applications.
- Automated testing – tests can be written in JavaScript or Solidity to make sure a smart contract behaves as expected before deployment

Figure 11: Generated folder to set up solidity smart contract framework from HardHat

3.4 Project Setup

Blockchain application can be used in most trade application, which include application for E-commerce, or just as simple as transfer funds between two accounts.

In this section we will be going over a sample application that is powered by web 3.0 and blockchain technology. We will create a website that allows a user to transfer cryptocurrencies between different wallets and provides a history of all transactions completed on the site. The web application will provide following functions:

- User can login to their crypto wallet
- User can enter the receiver address, Ether amount, keyword and a message for the transaction.
- Web page Display notification for the transaction and gas needed(minimum fee to pay in cryptocurrency to make transaction)
- User can confirm the transaction
- Transaction history is displayed on the webpage and can be verified in EtherScan

To see the detailed code and complete web application please refer to the Github page and deployed website in appendix A.

As part of this project we will develop, test, and deploy a smart contract using the HardHat development environment. To set up the environment, we can use any JavaScript code editor. In this case we will be using VSCode and deploy the project using "npm" to install the hardhat package from the terminal.

```
1.  npm install –save-dev hardhat
2.  npx hardhat
```

After running the script, a sample application will be created which includes the project folder structure.

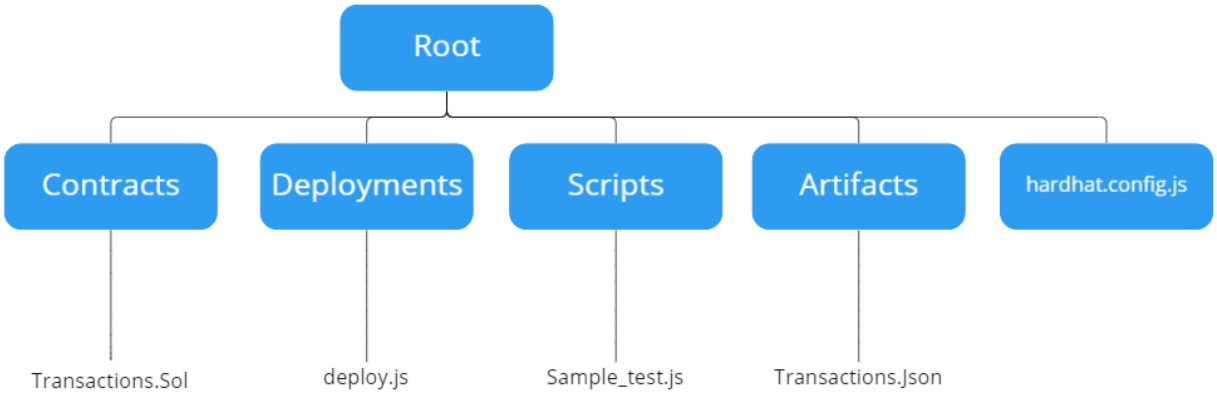


Figure 12: Sample smart contract project structure

The purpose of each directory is as follows:

Contracts -> This folder contains all contracts, interfaces, and external libraries created by the developer. Each contract is written in solidity and contains information about each block as well as functions needed for the contract to execute.

Deployments -> scripts to deploy contract to the network

Scripts -> test cases for each smart contract

Artifacts -> all the relevant information such as ABI used to connect to the client-side application

Hardhat.config.js -> configuration of hardhat, usually includes information such as the version of solidity, connected network, and account used to push the smart contract to the blockchain network.

After creating the sample project, the first thing we need to do is set up the configuration file for our smart contract project. In this case, we need to push the smart contract to Ropsten test network, which permits testing of blockchain development prior to mainnet release. [13] The test network behaves the same as mainnet and provides free test Ether to use to perform transactions on the network. Each test net either uses proof-of-work or proof-of-authority as their validation algorithms. Each test net are usually supported by different web3.0 libraries such as geth and parity, potentially provide better network connection and more blockchain framework to work with. Below are some other examples of test net. [14]

Network	Network ID	Detail
Main	1	
Ropsten	3	POW, support by geth and parity
Rinkeby	4	POA, supported by geth
Kovan	42	POA, supported by parity

Table 2: comparison between different blockchain network

To access to the network, we will need to connect our smart contract to a node provider. In this case we use alchemy, where we can choose different blockchains and different networks we want to connect to. Once a node is created, it will provide an HTTP key that will be used to locate our smart contract to the network. We simply copy the key and paste it into the config file.

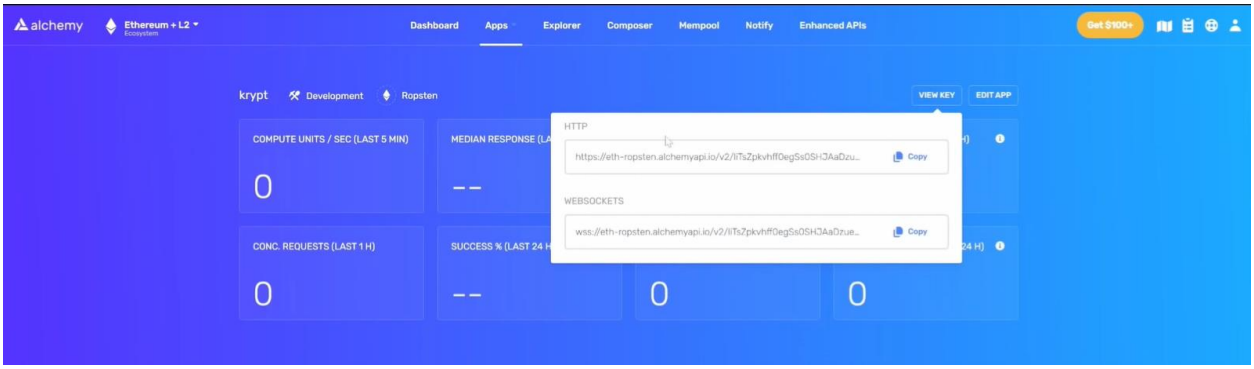


Figure 13: node set up in Alchemy

The second piece of information needed to deploy our contract to the network is the account to use to push the contract. The main reason we need an account is that in order to create any item on the blockchain network, "gas" is needed and is based on the supply and demand for the computational power needed to process the deployment in the network. "gas" is based on the existing supply and demand for the computational power needed to process the deployment of the smart contract to the network. Therefore, an account is needed to pay a gas fee to deploy the contract to the network. [15]

An account can be created in crypto wallet. In this case we will be using MetaMask as our wallet. An account will be a string with random letters and digit, and these are information we need to set up the config file. To create the wallet, a secret recovery phrase will be created which contains a combination of vocabulary. This secret recovery phrase is used to back up all cryptocurrencies and access to all accounts contained in the wallet. Once the wallet set-up is finished, it can connect to the Ropsten test network. Free test Ether can be obtained from any faucet, which are websites that can earn crypto by performing some simple tasks such as watching ads, solving quizzes, etc. After earning some Ether, we can then use those Ether for deployment of the smart contract. [16]

```
3.  require('@nomiclabs/hardhat-waffle')
4.
5.  module.exports= {
6.    solidity: '0.8.0',
7.    networks:{
8.      ropsten:{
9.        url: 'https://eth-ropsten.alchemyapi.io/v2/J2x0OA_WjOqtE-dy7DpvVt9aJzx1Il65',
10.       accounts: ['a0ae23cd6659c94984236894ced0764687e217d4ddecc7638982d822aaa227e2']
```

```
    }
  }
}
```

Figure 14: Hardhat.config.js

After completing the set up, smart contract can be developed in "Contract" folder and be tested in the Ethereum test network.

3.5 Building the Smart Contract

The first step for developing our smart contract is to create a file named 'transactions.sol" and store it in the Contracts folder within the development environment. As part of each transaction, we want the user to be able to see the detail of the transaction together with additional information such as a message and keyword. This additional information can also be a description or signature if each block is treated as a separate product.

```
pragma solidity ^0.8.0;
contract Transactions {

}
```

Within the 'transactions.sol' file, first, the version of solidity and contract name needs to be declared. Next, we will set up the basic data structure and global variables needed for our smart contract.

```
pragma solidity ^0.8.0;
contract Transactions {
    uint256 transactionCount;

    event Transfer(address from, address receiver, uint amount, string message, uint256 timestamp, string keyword);

    struct TransferStruct {
        address sender;
        address receiver;
        uint amount;
        string message;
        uint256 timestamp;
        string keyword;
    }

    TransferStruct[] transactions;

    event Transfer(address from, address receiver, uint amount, string message, uint256 timestamp, string keyword);
}
```

transactionCount -> global variable which counts the number of blocks added to the blockchain. Functions for this variable will be developed inside the contract as well.

TransferStruct -> contains the data structure for each block, and stores information related to each transaction. In this case it provides information about the sender, receiver, amount, message,

timestamp and keyword. Data structures vary and appropriate attributes can be created for different projects.

Transactions -> list of "Transferstruct", and in this case this is our blockchain to store all "Transferstruct" block.

Transfer -> a blockchain event, which is used to communicate with the client-side application. This event can be emitted in contract functions and client-side applications can receive information from the event to make useful insights.

Next, functions need to be created to help transfer Ether in the front end. These functions include:

- Adding a transaction to blockchain, while triggering an event to notify the user that a transaction has been made
- Getting all history for the blockchain
- Returning the total number of transactions

```
1.  function addToBlockchain(address payable receiver, uint amount, string memory message,
    string memory keyword) public {
2.      transactionCount += 1;
3.      transactions.push(TransferStruct(msg.sender, receiver, amount, message,
    block.timestamp, keyword));
4.
5.      emit Transfer(msg.sender, receiver, amount, message, block.timestamp, keyword);
6.  }
7.
8.  function getAllTransactions() public view returns (TransferStruct[] memory) {
9.      return transactions;
10. }
11.
12. function getTransactionCount() public view returns (uint256) {
13.     return transactionCount;
```

"addToBlockchain" function will receive input including receiver, amount, message, and keyword. The "transactionCount" variable will increment by 1. A new "transferStruct" block will be created and added to the transactions blockchain. It will also emit input information to the client-side application for access.

"GetAllTransactions" function will return a transactions array and "getTransactionCount" function will return the total number of blocks in transactions.

3.6 Deploy Smart Contract

The next step would be to test the contract function, but because our function is very straight forward, we will skip ahead to deploy our smart contract. The deployment process will be done in deployment folder.

```
1.  const main = async () => {
2.    const transactionsFactory = await hre.ethers.getContractFactory("Transactions");
3.    const transactionsContract = await transactionsFactory.deploy();
4.
5.    await transactionsContract.deployed();
6.
7.    console.log("Transactions address: ", transactionsContract.address);
```

```
};

const runMain = async () => {
  try {
    await main();
    process.exit(0);
  } catch (error) {
    console.error(error);
    process.exit(1);
  }
};

runMain();
```

Here the constant "main" will contain the deployment process. "hre.ethers" is a javascript plugin to interact with the ether object. "ContractFactory" is an abstraction used to deploy new smart contracts. By calling "deploy()" on a "ContractFactory" the deployment process will start. The deployment process will spend gas from the account and send the smart contract to the address given by node provider.

```
Compiling 1 file with 0.8.0
Compilation finished successfully
Transactions deployed to: 0xF747CFC3291fdC2B4066d3e782384B8C65035583
```

This address is where the contract is stored in the blockchain network, and we can use this address to access the smart contract together with all the contract detail in the client-side application. Additionally, an "artifact" folder will be created, which contains all the useful information to a contract including ABI, contract bytecode, and deployment details in a JSON bundle.

ABI -> stands for Contract Application Binary Interface, which is used to interact with smart contacts in the Ethereum ecosystem including client-side applications or contract-to-contract interactions. [17]

Contract bytecode -> a compiled smart contract into EVM bytecode used to deploy to blockchain networks as the Ethereum runtime environment only understands and executes bytecode. The main benefit to using bytecode is that other programing languages can also create smart contracts and contract bytecode can convert into other programing languages, giving developers significant amounts of freedom when creating blockchain applications.[18]

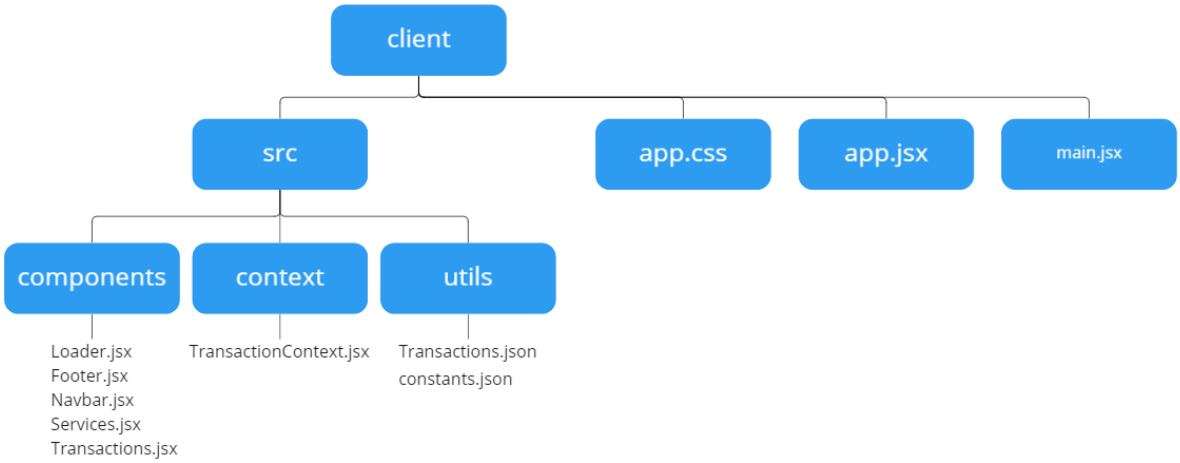
Deployment Details -> includes all inputs and outputs for the smart contract functions

3.7 Client-side Application

The General client-side stack is written in a combination of JavaScript, html, css files, and can be set up using various web development frameworks such as React, Vite, Angular, Vue, etc. For this project we will be using Vite as our framework, which can be imported using the "npm" command.

```
npm init vite@latest
```

The client-side application will consist of the three files (main.jsx, app.jsx, and app.css), as well as a "src" folder to contain all front-end and backend logic.



Src -> contains frontend and backend components

Components -> frontend component, written in Jsx file, which contains JavaScript and html

Context -> backend component, decides how to receive information from the front- end and process it together with the connected smart contract using "Ether.js"

Utils -> includes constants, transaction details, and global variables. Constants include the smart contract address and ABI. Transaction details include all functions of the smart contract as well as the block detail stored in json format. Global variables can include commonly used variables on the web application.

App.css -> setting for aesthetics on the website

App.jsx -> layout of the app

Main.jsx -> deployment program for the application

After setting up the sample application, the first task is to set up the connection with the smart contract. This can be accomplished by putting the "transaction.json" file generated from the artifact folder in the smart contract and put "transaction.json" into utils and creating variables for ABI and the contract address.

```
1.  import abi from './Transactions.json';
2.
3.
4.  export const contractABI = abi.abi;
5.  export const contractAddress = '0xF747CFC3291fdC2B4066d3e782384B8C65035583';
```

In order to connect to the client-side application, ABI and the address for the smart contract must be imported from the artifact folder and be used as two separate constants. These constants can be interpreted by "Ether.js", a JavaScript package, that allows us to interact with the Ethereum object and its ecosystems. As a result, we get access to below features[19]:

- Importing and exporting JSON wallets(Geth, Parity)

- Importing and exporting BIP 39 mnemonic phrases (12-word backup phrase) and HD Wallets (Metamask)
- Meta-classes that create JavaScript objects from any contract ABI, including ABIv2 and Human-Readable ABI
- Connecting to Ethereum nodes over JSON-RPC, INFURA, Etherscan, Alchemy, Cloudflare or MetaMask.
- MIT License (including ALL dependencies); completely open source to do with as you please.

By accessing the contract ABI and location, backend functions in the context folder are able to access Ethereum objects and interact with their attributes as well as the functions created in the smart contract.

```
1.  const {ethereum} = window;
2.
3.  const getEthereumContract = ()=> {
4.      const provider = new ethers.providers.Web3Provider(ethereum);
5.      const signer = provider.getSigner();
6.      const transactionContract = new
    ethers.Contract(contractAddress,contractABI,signer);
7.      return transactionContract;
```

In the example code above, three variables are created: the provider, the signer, and the transactionContract. The provider is created using "Ethers.providers.Web3Provider", which is a read-only abstraction to access the blockchain. [20] A signer is an Ethereum account that created the transactions and signed the transactions to the Ethereum network to execute state change. [20] The transactionContract is the smart contract we created which can be accessed with the contract address, contract ABI and the signer. Once we get access, we can see all the inputs and outputs for the smart contract and use its built-in functions.

3.8 Backend function

Backend functions can be stored in the context folder and since the application is using Vite framework, the logic will be written in JavaScript. Various functions need to be developed to complete the back-end features; however, only the main function "sendTransactions" will be discussed. This function allows the user to send an amount of Ether to another user anywhere in the world along with the keyword and a message for the transaction. To view the full set of backend functions please refer to appendix A.

```
1.  const sendTransaction =async () => {
2.      try {
3.          if(!ethereum) return alert('Please install metamask');
4.          //get the data from the form...
5.          const {addressTo,amount,keyword,message} = formData;
6.          const transactionContract = getEthereumContract();
7.          // this constant us to parse ether to GWEI
8.          const parsedAmount = ethers.utils.parseEther(amount);
9.
10.         await ethereum.request({
11.             method: 'eth_sendTransaction',
12.             params:[{
```

```
                from:currentAccount,
                to:addressTo,
                gas: '0x5208', //21000 GWEI, around 0.00002 ETH
                value:parsedAmount._hex, //0.00001
            }]
        });
        console.log({
            addressTo,
            parsedAmount,
            message,
            keyword,
        });
        const transactionHash = await transactionContract.addToBlockchain(addressTo,
    parsedAmount, message, keyword);
        // this part make sure the asyncreous process finish
        setIsLoading(true);
        console.log(`Loading - ${transactionHash.hash}`);
        await transactionHash.wait();
        setIsLoading(false);
        console.log(`Success - ${transactionHash.hash}`);

        const transactionCount = await transactionContract.getTransactionCount();
        setTransactionCount(transactionCount.toNumber());

        window.reload();
    } catch (error) {
        console.log(error);
        throw new Error('No ethereum object')
    }
}
```

First the input for the 'sendTransaction' function is received from "formData", which is a variable in the front-end which gives details about the address, amount, keyword, and message. Meanwhile, the smart contract can be called as a Ethereum object from the "getEthereumContract()" function. In order to send a transaction to the network, by using "ethereum.request" function will submit an RPC (Remote Procedure Call) to Ethereum via Metamask. [21] The transaction will determine the current account, the receiver account, the "gas" necessary to complete the transaction and the number of Ethers sent in the transaction. Then transaction history can be documented by calling out the smart contract object, execute "addToBlockchain" and "getTransactionCount" to complete the transaction.

```
return (
      <TransactionContext.Provider value = {{handleChange,sendTransaction}}>
          {children}
      </TransactionContext.Provider>
  )
```

The backend function can be exported to the frontend and called when the conditions are met. They can also act as external variables and functions.

3.9 Frontend interfaces

The Frontend is written in html to receive the input from the user and create the frontend design using an external CSS package such as "tailwind.css". In this sample app we will simplify the frontend with only necessary components. For more detail, please refer to appendix A.

```
<div>
    <Input placeholder="Address To" name="addressTo" type='text' handleChange={handleChange}/>
    <Input placeholder="Amount (ETH)" name="amount" type='number' handleChange={handleChange}/>
    <Input placeholder="Keyword (GIF)" name="keyword" type='text' handleChange={handleChange}/>
    <Input placeholder="Enter Mesage" name="message" type='text' handleChange={handleChange}/>

    <div/>
        <button
        type = 'button'
        onClick={handleSubmit}
        >
        Send Now
        </button>
    )}

</div>
```

The webpage will have four input boxes for user to enter information about the receiver address, the amount, keyword, and a message for a single transaction. These inputs will get transferred after clicking a button and triggering the "handleSubmit" function.

```
import { TransactionContext } from '../context/TransactionContext';
const {formData, ,sendTransaction} = useContext(TransactionContext);
const handleSubmit = (e) => {
    const {addressTo,amount,keyword,message} = formData;

    e.preventDefault();

    if(!addressTo || !amount || !keyword || !message) return;

    sendTransaction();
};
```

"handleSubmit" will use functions and variables from the backend which can be imported from the context folder. This function will basically store all inputs into the "formData" variable, and pass it to the "sendTransaction" function, which then will be processed by the backend and complete the transaction.

3.10 Deployment

Deployment can be done using the "build" function, which creates a build directory with a production version for the web application. HTTP server can also be set up by customize "index.html", where the deployment service can access all the source file in an encrypted static directory that contains all JavaScript and CSS file. Each file in build/static directory will be encrypted with a unique hash of the file content which avoids the browser re-downloading assets and prevent others accessing the source code. [22]

```
1.  npm run build
```

The above command generates a deployment version of the web application, which can be deployed to public network by choosing a hosting service such as AWS, Azure, Heroku, etc. Each hosting service varies with regards to the specific deployment process, but overall contains a process to allow the developer to insert the build directory and push the web application to the internet.

4.0 Leverage blockchain technology in Eaton Aerospace

[REMOVED]

5.0 Conclusion

Blockchain technology has provided safety for data transfer without any involvement of external security, and opens up endless possibility to create decentralised application, reduce both time and cost. it is currently used in many commercialised applications such as cryptocurrency, value registry, value economy and value web and is predicted to integrates and evolve current trade ecosystem for all types of customers. Blockchain technology is also essential to build decentralised app, also known as Web 3.0, that brings decentralization and machine-readable language (for machine learning) within the application.

Currently there are many blockchain development framework and most industry standard way to develop an Blockchain application stack consist of crypto wallet, frontend, node provider, smart contract, and blockchain network.

Those enterprises who have a strong drive to innovate first will most likely gain the largest benefits. The blockchain is at the immature phrase and is the perfect catalyst for business process changes, and this technology could be the next iPhone at launch or the start of internet. By utilizing the decentralized nature, real-time transfer between data and transactions, it will be what the consumer and business want before they know it.

Blockchain technology is still an emerging area of innovation but has already been noticed by various industry and its impact will vary based on future cooperation as well as the adoption of blockchain application by current market players.

As a newcomer into the industry, Eaton should first choose the strategy between investment, in-house development, partnership. Blockchain technology should be carefully considered and addressed with many workshops and training for key stakeholders to understand. Opportunities will be addressed in all division of aerospace and integrate blockchain when decentralization is beneficial

Bibliography

[1]
A. Hayes, “Blockchain Explained,” *Investopedia*, Mar. 05, 2022. https://www.investopedia.com/terms/b/blockchain.asp
[2]

K. E. Wegrzyn and E. Wang, “Types of Blockchain: Public, Private, or Something in Between | Foley & Lardner LLP,” *www.foley.com*, Aug. 19, 2021. https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between
[3]
“How Many People Own, Hold & Use Bitcoins? (2022),” *buybitcoinworldwide.com*. https://buybitcoinworldwide.com/how-many-bitcoin-users/
[4]
“What Is An NFT? Non-Fungible Tokens Explained,” *Forbes Advisor Canada*, Aug. 16, 2022. https://www.forbes.com/advisor/ca/investing/cryptocurrency/nft-non-fungible-token/#:~:text=An%20NFT%20is%20a%20digital
[5]
“Blockchain Technology Market Size, Share | Industry Report, 2019-2025,” *www.grandviewresearch.com*. https://www.grandviewresearch.com/industry-analysis/blockchain-technology-market
[6]
L. Insights, “ConsenSys’ Truffle partners with Microsoft to improve Enterprise Ethereum developer experience,” *Ledger Insights - blockchain for enterprise*, Jun. 12, 2019. https://www.ledgerinsights.com/blockchain-enterprise-ethereum-consensys-truffle-microsoft-azure/ (accessed Aug. 25, 2022).
[7]
“Blockchain: Powering the Internet of Value WHITEPAPER PAGE 2 LABS.” [Online]. Available: https://blockchainlab.com/pdf/bank-2020---blockchain-powering-the-internet-of-value---whitepaper.pdf
[8]
L. Insights, “IHS Markit bullish about blockchain, predicts $2 trillion market by 2030,” *Ledger Insights - blockchain for enterprise*, Jun. 13, 2019. https://www.ledgerinsights.com/ihs-market-blockchain-forecast-2-trillion/ (accessed Aug. 25, 2022).
[9]
“What is Dependency Injection?,” *SearchAppArchitecture*. https://www.techtarget.com/searchapparchitecture/definition/dependency-injection#:~:text=In%20object%2Doriented%20programming%20(OOP (accessed Aug. 25, 2022).
[10]
Node.js Foundation, “About | Node.js,” *Node.js*, 2014. https://nodejs.org/en/about/
[11]
“What’s Vite: The Guide to Modern, Super-Fast Project Tooling,” *Telerik Blogs*, Jan. 13, 2022. https://www.telerik.com/blogs/whats-vite-guide-modern-super-fast-project-tooling (accessed Aug. 25, 2022).
[12]
“What is a private key?,” *SearchSecurity*. https://www.techtarget.com/searchsecurity/definition/private-key
[13]
“Ethereum’s Ropsten Testnet is live! What’s next,” *Business Today*, Jun. 09, 2022. https://www.businesstoday.in/crypto/story/ethereums-ropsten-testnet-is-live-whats-next-336893-2022-06-09 (accessed Aug. 25, 2022).

[14]
Kenneth胡家維 H., “Ethereum Test network,” *Coinmonks*, Jun. 26, 2022. https://medium.com/coinmonks/ethereum-test-network-21baa86072fa (accessed Aug. 25, 2022).
[15]
“Gas (Ethereum),” *Investopedia*. https://www.investopedia.com/terms/g/gas-ethereum.asp#:~:text=Key%20Takeaways
[16]
“Explained | What is a crypto faucet and what are its advantages?,” *cnbctv18.com*, Apr. 14, 2022. https://www.cnbctv18.com/cryptocurrency/explained-what-is-a-crypto-faucet-and-what-are-its-advantages-13159802.htm (accessed Aug. 25, 2022).
[17]
“Contract ABI Specification — Solidity 0.8.13 documentation,” *docs.soliditylang.org*. https://docs.soliditylang.org/en/v0.8.13/abi-spec.html#:~:text=Contract%20ABI%20Specification- (accessed Aug. 25, 2022).
[18]
“Byte Code,” *www.zastrin.com*. https://www.zastrin.com/courses/ethereum-primer/lessons/2-7#:~:text=Smart%20contract%20code%20is%20usually (accessed Aug. 25, 2022).
[19]
“ethers,” *docs.ethers.io*. https://docs.ethers.io/v5/single-page/ (accessed Aug. 25, 2022).
[20]
“01 Understanding Blockchain with `Ethers.js`: 4 Tasks of Basics and Transfer,” *DEV Community*. https://dev.to/yakult/01-understanding-blockchain-with-ethersjs-4-tasks-of-basics-and-transfer-5d17#:~:text=A%20Provider%20in%20ethers%20is (accessed Aug. 25, 2022).
[21]
“Ethereum Provider API | MetaMask Docs,” *docs.metamask.io*. https://docs.metamask.io/guide/ethereum-provider.html#methods
[22]
“Deployment | Create React App,” *create-react-app.dev*. https://create-react-app.dev/docs/deployment/#:~:text=npm%20run%20build%20creates%20a (accessed Aug. 25, 2022).
[23]
“GE Aviation’s Digital Group streamlines tracking of aircraft parts, reduces inefficiencies with Azure Blockchain technologies,” *Microsoft Customers Stories*. https://customers.microsoft.com/da-dk/story/755328-ge-aviation-manufacturing-azure (accessed Aug. 25, 2022).
[24]
“Monetizing blockchain A tailwind for aviation IBM Institute for Business Value.” [Online]. Available: https://www.ibm.com/downloads/cas/5JLAYLZE
[25]
“Aeron - Blockchain for Aviation Safety,” *aeron.aero*. https://aeron.aero/
[26]
“Ozone,” *www.ozone-platform.com*. https://www.ozone-platform.com/
[27]
“Aviation,” *SIMBA Chain*. https://simbachain.com/industries/aviation/ (accessed Aug. 25, 2022).
[28]

"Honeywell Uses Blockchain To Digitize Aircraft Records, Parts Pedigree Data," *www.honeywell.com*. https://www.honeywell.com/us/en/press/2020/08/honeywell-uses-blockchain-to-digitize-aircraft-records-parts-pedigree-data (accessed Aug. 25, 2022).
[29]
M. Brohan, "An aerospace parts marketplace flies high with blockchain," *Digital Commerce 360*, Jan. 11, 2021. https://www.digitalcommerce360.com/2021/01/11/an-aerospace-parts-marketplace-flies-high-with-blockchain/ (accessed Aug. 25, 2022).
[30]
"Digital Narrative for Aerospace and Defense | Accenture," *www.accenture.com*. https://www.accenture.com/ca-en/insights/aerospace-defense/digital-narrative-aerospace-defense (accessed Aug. 25, 2022).
[31]
"Loyyal," *Loyyal*, 2018. https://loyyal.com/
[32]
Vivek Madurai, "Web Evolution from 1.0 to 3.0," *Medium*, Feb. 17, 2018. https://medium.com/@vivekmadurai/web-evolution-from-1-0-to-3-0-e84f2c06739
[33]
"What is Web 3.0: A beginner's guide to the decentralized internet of the future," *Cointelegraph*. https://cointelegraph.com/blockchain-for-beginners/what-is-web-3-0-a-beginners-guide-to-the-decentralized-internet-of-the-future

Appendix A:

Full stack code:

- https://github.com/EricFeng20001120/blockchain-web3.0-eth-transfer-application

- https://github.com/EricFeng20001120/blockchain-web3.0-eth-transfer-application.git

deployed web application:

- https://kryptomasteryforportfolio.com/