Journal of Network and Computer Applications 216 (2023) 103650

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Survey of prominent blockchain development platforms

Collin Connors *, Dilip Sarkar

University of Miami, Department of Computer Science, United States of America

ARTICLE INFO

Keywords:
Blockchain
Dapp
Web3
Smart contract
Blockchain development

ABSTRACT

Many developers have ideas to create blockchain applications but do not know where to begin. Often these developers default to using the first blockchain development platform they discover, which may not be the best platform for their project. Over 8000 new blockchain-related projects are added to GitHub a year. The near-constant influx of new projects can make it difficult for developers to search through existing projects and platforms to find the best platform for their projects. We considered 65 blockchain development platforms for this work and provided a brief yet comprehensive summary of the 23 most popular platforms. Our aim for this work is to assist developers in selecting the most appropriate platform for their blockchain projects.

1. Introduction

In the seminal paper, Bitcoin: A peer-to-peer electronic cash system (Nakamoto, 2008), Satoshi Nakamoto introduced the world to blockchain. While the system described in this paper is never explicitly called a blockchain, it has all the properties of modern blockchains. Blockchains, as we have now come to know them, are tamper-evident and tamper-resistant digital ledgers implemented in a distributed fashion (Yaga et al., 2018). This technology initially led to the rise of cryptocurrencies such as Bitcoin.

However, as blockchains became more popular, developers began to expand the use cases of blockchains. This second generation of blockchains allowed developers to create smart contracts, code that can be run on a blockchain (Zou et al., 2021). With smart contracts, developers could create decentralized applications governed by a blockchain. These decentralized applications are called dApps, which have become central in Web3 development. Web3 development depends on many ideas brought over from traditional web development, sometimes referred to as Web2 development. To create a traditional application in Web2 development, developers create and manage a front-end and a back-end layer. The front-end layer is responsible for displaying information to a user. This layer often relies on technology such as HTML, CSS, and JavaScript so a web browser can render a page. The back-end layer manages the data through databases, APIs, and web hosting. Web2 developers use the front-end and back-end layers to create dynamic web applications.

In Web3 development, a developer still has a front-end and a back-end layer in addition to a new smart contract layer. The smart contract layer allows Web3 developers to interact with a blockchain to create dApps. This new layer interacts with the front-end and back-end layers. However, the smart contract layer allows developers to create applications that do not have a centralized entity managing the data.

In Web2 development, developers have many choices of development platforms for the front-end and back-end layers, such as various JavaScript frameworks or back-end scripting languages. In Web3 development, developers have similar choices for the smart contract layer. Critically, to create a dApp, a developer must select an appropriate blockchain development platform for their project.

In Web2 development, one of a new developer's most challenging choices is selecting the appropriate front-end and back-end technology stack for a project. The same problem exists in Web3 when selecting a blockchain development platform. There are many blockchain development platforms, with new ones being created yearly. This can make it difficult for developers to select the platform that best complements their projects.

Our motivation for this work is to simplify the processes of selecting a blockchain development platform to make it easier for developers to start creating dapps. To do this, we have considered 65 blockchain development platforms and selected the 23 most popular platforms to analyze. Our analysis aims to highlight the differentiators between various blockchain development platforms so that developers can understand which platform will best synergize with their projects. Previous surveys, while extensive, need to provide everything a potential developer needs to know. These works only cover a few blockchain development platforms (Valenta and Sandner, 2017), only cover blockchain development platforms for a specific industry (Agbo and Mahmound, 2019), only cover specific types of blockchain development platforms (Quasim et al., 2020; Polge et al., 2021; Javaid et al., 2021), or do not provide enough details necessary to select the

* Corresponding author.
E-mail address: cdc104@miami.edu (C. Connors).

https://doi.org/10.1016/j.jnca.2023.103650
Received 27 October 2022; Received in revised form 27 February 2023; Accepted 19 April 2023
Available online 25 April 2023
1084-8045/

most appropriate platform (Tavares et al., 2019; Dernayka and Chehab, 2021). In previous work (Connors and Sarkar, 2022b), we performed a comparative study to the one presented here but only on 15 platforms. Since publishing that work, we have expanded the number of platforms analyzed, collected more recent data, and selected more metrics to analyze. Likewise, a preprint version of this paper is available (Connors and Sarkar, 2022a).

We have organized the rest of the paper as follows. In Section 2, we provide some background information on blockchain, blockchain development platforms, and some technical aspects that differentiate blockchains. In Section 3, we provide some details on related work. In Section 4, we present our 23 selected platforms and our methods for choosing those platforms. In Section 5, we analyze the selected blockchain development platforms on various metrics. Our analysis is broken into six subsections and covers the data presented in our seven tables. In Section 6, we discuss how developers can use our analysis to select a blockchain development platform for their projects. Lastly, we provide some concluding thoughts in Section 7.

2. Background

In its most basic form, a blockchain is a digital ledger that is tamper-resistant, tamper-evident, and often implemented in a distributed fashion. A blockchain maintains these properties through the use of cryptographic hashes. A blockchain is segmented into "blocks" that contain data. Each block contains the cryptographic hash of the previous block. Including the previous block's hash in a block creates a chain of blocks linked by hashes. If the data changes in any block, the hashes must be updated in all subsequent blocks. For a more detailed description of blockchain, see Yaga et al. (2018), Guo and Yu (2022), and Nofer et al. (2017).

This work does not look at specific blockchains but rather blockchain development platforms. A blockchain development platform is any platform that allows users to develop blockchain-based applications. For example, Ethereum is a blockchain development platform, whereas the Ethereum mainnet or the Ropsten testnet are examples of Ethereum blockchains. While this distinction may seem nuanced, it is an important distinction to make concerning permissioned blockchains. Platforms such as Hyperledger Fabric let developers build their own blockchains. In this case, it is clear that the blockchain development platform is different from the blockchain. Blockchains differentiate themselves through the Sybil control mechanism (Iqbal and Matulevičius, 2021). Many previous works refer to this as the consensus protocol or the consensus algorithm. However, these mechanisms specifically prevent Sybil attacks; thus are properly called Sybil control mechanisms. In contrast, the consensus protocol is how a network agrees when a dispute, such as a fork in the blockchain. In Bitcoin, the Sybil control mechanism is Proof of Work (PoW), and the consensus protocol is that the network selects the longest chain when there is a fork.

There are many different Sybil control mechanisms, and it is out of the scope of this work to discuss them here. However, readers are encouraged to see Mingxiao et al. (2017), Nguyen and Kim (2018), and Velliangiri and Karthikeyan (2020) for detailed surveys covering popular Sybil control mechanisms. Another differentiator of blockchains is the permission structure. A blockchain can be either public or private, as well as permissionless or permissioned. These properties define how users can interact with a blockchain network.

A public blockchain is a blockchain where any user can join and read the data on the blockchain. Likewise, a blockchain can be permissionless, where any user can join the Sybil control mechanism and attempt to add new blocks. In contrast, a private blockchain only allows a small group of users to read the data on the blockchain. Similarly, in a permissioned blockchain, users must be permitted to join the Sybil control mechanism and attempt to add new blocks.

Lastly, blockchain development platforms can be categorized by their generation (Tavares et al., 2019). Being the first blockchain, Bitcoin is a generation 1.X blockchain. This generation of blockchain focuses on a single use case for the blockchain and only allows for a small amount of expansion. The Bitcoin blockchain only focuses on recording $XBT transactions and does not allow developers to write expansive dapps.

Wanting to expand blockchain technology further than just cryptocurrency, developers created generation 2.X blockchain development platforms such as Ethereum. These platforms allowed developers to write smart contracts. This generation of blockchain, for the first time, allowed developers to use blockchain in a variety of use cases and led to developers creating dapps and Web3.

While generation 2.X development platforms are still popular with developers, generation 3.X platforms aim to create blockchains suited for enterprise applications. Some key features of generation 3.X blockchain development platforms include Layer 2 protocols, machine-to-machine (M2M) communication, mobile compatibility, the use of DAGs, and the use of compute protocols.

In the following sections, we will use the terminology discussed here to highlight the features of various blockchain development platforms.

3. Related work

As blockchain has become more popular, many have surveyed various blockchain development platforms. As early as 2017 (Valenta and Sandner, 2017), researchers have been comparing existing blockchain platforms with one another. This work compares Ethereum, Hyperledger Fabric, and Corda using characteristics such as Governance, Consensus, and native currency.

Other works, such as Polge et al. (2021), further analyzed the performance of Hyperledger Fabric, Ethereum, Quorum, Multichain, and Corda. This work does not look into the practical development considerations, such as how smart contracts are written but rather compares the performances of the given blockchains. This work found that Hyperledger Fabric and Ethereum performed best under their tests.

Similar performance comparison work such as Dernayka and Chehab (2021) have found that Ethereum performed best under their tests. This work compared additional features such as storage and resources needed. However, this work only analyzed Ethereum and EOS, leaving out many popular blockchains. Likewise, other comparisons only analyze blockchain platforms in the context of a specific domain such as healthcare (Agbo and Mahmound, 2019). This work compared features like the speed, security, and scalability of various blockchain development platforms in healthcare applications. Again this work only compared a few platforms.

More extensive work such as Tavares et al. (2019) looks at blockchain research as a whole. This work uses various metrics to compare block chains, such as Blockchain 3.0 features, Consensus Features, and Documentation. We also use some of these metrics in our work to compare the blockchain platforms. However, this work does not distinguish between blockchain development platforms and other blockchain projects, making it difficult for new developers to select an appropriate platform.

This previous work leaves out many popular blockchain platforms in their comparisons. New blockchain developers should be aware of all the major blockchain development platforms to decide which platform best suits their project. Likewise, these surveys do not provide the necessary details for new developers to choose which blockchain platform to use. While knowing the performance of various blockchains is helpful, new developers need to consider more aspects, such as the features provided, the learning resources provided, or the development environment. In this work, we look to improve by comprehensively comparing the 23 most popular blockchain development platforms. We show how each platform differs and provide the necessary information for developers to select a platform that best suits their projects.

The following section describes how we selected our platforms to analyze. We then provide a series of metrics comparing the various platforms. Finally, we discuss how developers can use our metrics to select an appropriate platform.

4. Selected projects

There are over 146,000 blockchain projects on GitHub. Most of the projects were created only in the last few years. Since there are so many projects, it can take time for new developers to find a development platform that best suits their project's goals. This work aims to provide developers with the information to select a platform to develop blockchain applications. Since there are so many blockchain development platforms, with many new ones being created each year, this work only analyzed popular open-source platforms allowing dynamic smart contract development.

We consider a project sufficiently popular if it has more than one thousand (1000) stars on GitHub. The popularity restraint ensures that our analyzed blockchain development platforms are of high quality. We set our popularity requirement low enough to include many platforms yet high enough to ensure all of our discussed platforms are well known. However, this popularity restraint favors older blockchains as they have had more time to amass a following. Likewise, we only analyzed open-source platforms so developers could see the underlying code. The added transparency of open-source projects allows for more trusted code.

Lastly, we only looked at platforms that allowed for dynamic smart contract development since our goal is to assist developers in creating new distributed applications. This restraint limits the projects analyzed to only platforms that allow developers to use generation 2.X or 3.X blockchains. In total, we considered 65 platforms; however, only 23 of those platforms meet our requirements. Some well-known platforms that did not satisfy our requirements include Cardano (David et al., 2023), Avalanche (Sekniqi et al., 2020), IOTA (Popov, 2018), and Lisk (Anon, 0000b). The complete list of 65 platforms considered is listed in our GitHub Repository (Connors, 2023).

Section 4 shows the projects we selected. We marked projects that are forks of older projects with an asterisk. Quorum and BNBChain are forks of Ethereum, and Qtum is a fork of Bitcoin. Note that the data for all of our tables were collected in September 2022. Many whitepapers defining blockchain development platforms have not been submitted to peer-reviewed publications. Instead, many platforms publish their whitepaper directly to their website or GitHub page. We have provided our own Github Repository (Connors, 2023) with links to each project's website, whitepaper, and other helpful links related to the project.

In the following section, we will analyze our 23 selected platforms and provide metrics to assist developers in selecting a platform that best complements their projects.

5. Analysis of blockchain projects

5.1. Popularity

In the first stage of our analysis, we compared the popularity of our selected projects using the popularity score metric, shown in Table 2. Because of our popularity restraint, all 23 selected platforms have at least 1000 stars on GitHub. Thus, even the least popular of our selected platforms are well known. Developers can use our popularity measures to gauge how large the community surrounding a blockchain development platform is and how many developers are using a given platform.

Our popularity score metric, Eq. (1), is generated from various sources measuring the popularity of a project. We used GitHub Stars, GitHub Forks, the number of questions on the projects Stack Exchange, and the number of papers on Google Scholar mentioning each project. Some popularity metrics, such as the number of DApps developed, were left out of our popularity score since not all of our projects were listed in DApp repositories.

Since all our projects have GitHub repositories, we could compare the number of GitHub Stars. A Star in GitHub allows a developer to like a project and save it for later. Thus this metric reflects how many developers are interested in each blockchain development platform. Likewise, we looked at the GitHub forks. The number of GitHub forks lets us know how many developers are interested in creating projects based on a blockchain development platform.

Similarly, we examined the number of closed GitHub issues to see how many people were asking questions about the blockchain development platform. However, some platforms delete old issues when a new update to the blockchain development platform is released. Deleted issues explain why platforms like Hyperledger Fabric have fewer closed GitHub issues than expected. Because of the deleted issues, we do not include this metric when calculating popularity scores. We also measured the popularity of the projects outside of GitHub using Stack Exchange. Stack Exchange is a website where users can ask questions and receive responses from others with expertise in the platform. For all of our platforms, we searched Stack Exchange using the blockchain tag followed by the platform's name to eliminate questions unrelated to the blockchain. For example, when searching Waves in Stack Exchange, we revived many answers related to physics and not blockchain; thus, we searched for "[blockchain] Waves". While this may have removed some questions about each platform that did not include the blockchain tag, it ensured that all the questions returned were about the relevant blockchain development platform.

Likewise, using Google Scholar, we measured the number of academic papers related to each blockchain development platform. Again for all of our selected platforms, we needed to restrict the results to only papers containing the platform followed by the word blockchain. For example, when searching for Optimism, Blockchain returns many papers where the authors state their optimism about blockchain technology; thus, instead, we searched only for papers that included Optimism, followed by blockchain, which returned only papers about the platform. Again this likely dropped some papers related to the platform, but it ensured that all the papers found were relevant to the platform.

BNB Chain did not have any Google Scholar results. The lack of papers is because the blockchain was formally called the Binance Smart Chain (BSC). However, Binance merged the BSC with other blockchain technologies they were hosting and renamed it BNB Chain. We did not search for papers related to Binance Smart Chain because there are significant differences between BNB Chain and its predecessor.

Lastly, we looked at how many dapps each blockchain development platform had posted to DappRadar. DappRadar is a website that aggregates and ranks known dApps. Naturally, none of the platforms that create private blockchains have any entries in DappRadar since the applications create on these platforms are not public. However, some public blockchain platforms, such as Stellar, have no entries in DappRadar. Since DappRadar does not collect information on all our selected projects, we did not include the number of dApps when calculating the popularity score.

To calculate the popularity score shown in Eq. (1), we took the normalized sums of GitHub forks, GitHub stars, StackExchange questions, and Google Scholar papers. The most popular platform can have a score of 100.

$$PopularityScore_i = \frac{forks_i + stars_i + questions_i + papers_i}{forks_{max} + stars_{max} + questions_{max} + papers_{max}} * 100 \quad (1)$$

Ethereum is the most popular platform, with a score of 100. Hyperledger Fabric is the most popular private blockchain development platform and the second most popular platform, scoring 32.93. While our least popular project Zilliqa only has a score of 1.76. The popularity score only allows us to compare the selected project to each other. All platforms shown have at least 1000 stars on GitHub, meaning they are all well known.

A popular platform is not the best platform for every project. Developers should consider the popularity of a project only as a means to see how much support is available. We hope that by using this table in conjunction with our other tables, developers can choose the platform that best supports their projects.

5.2. Properties of blockchain development platforms

Our subsequent table, Table 3, describes the fundamental differentiators between blockchains tied to the blockchain development platforms. The table includes the blockchain generation, whether the blockchain is permissioned or permissionless, what Sybil control mechanism the blockchain uses if developers need to learn a framework-specific programming language to develop smart contracts and if a cryptocurrency governs the blockchain. Some of our selected platforms are tied to a well-established mainnet like Ethereum. In contrast, other platforms will have developers create their own blockchains, such as Hyperledger Fabric.

Most of our selected platforms are linked to generation 2.X blockchains. Of generation 3.X blockchains, both Optimism and Harmony are linked to layer 2 (L2) blockchains. An L2 blockchain performs transactions on a smaller side chain but uses a more extensive blockchain for finality. Both Optimism and Harmony finalize their transactions on the Ethereum mainnet blockchain. Developers should note that while L2s aim to increase transaction speed, they give up some security by using a smaller network. This trade-off of security for speed is essential in applications requiring exceptionally high network trust. However, the speed gained by using an L2 may outweigh the reduced security for many applications.

Likewise, NEO and IoTeX allow for M2M communication, another property of generation 3.X blockchains. Developers wishing to create IoT applications should consider using these blockchain development platforms.

Our selected projects contain a mix of permissioned and permissionless blockchains. Often enterprise or other private applications will require a permissioned blockchain. However, developers should be aware that when creating a premissioned blockchain, they may also need to create and operate blockchain nodes, create an authority to grant permission to the blockchain network, or even choose the consensus algorithm the network will use. For example, in Hyperledger Fabric, a developer needs to create their own private network before they can begin working on developing their dApp. While Hyperledger Fabric offers a suite of tools to make this step less complex, it still complicates the dApp development process.

In contrast, public-facing applications will often require developers to use permissionless blockchains. While developers will not need their own networks, they will often need to pay a fee to use the public mainnet. Likewise, developers will have little control over the network; thus, a permissionless blockchain may not be suitable for applications handling sensitive information. Developers should consider which permission structure is best for their application. Some selected platforms, such as EOS, NEO, and NEAR, allow developers to use a permissionless network or create a private network. Access to two types of networks allows developers to switch which type of network they are using without switching platforms. For example, a developer may start off using a public mainnet. However, the developer may want to release a premium version of their application on a private blockchain.

Our selected projects use various Sybil control mechanisms, often called consensus algorithms. Even though the original blockchain, Bitcoin, uses Proof of Work (PoW) as its Sybil control mechanism, none of our selected projects use PoW. PoW has been shown to have many negative drawbacks compared to modern algorithms such as Proof of Stake (PoS), including high energy consumption (Zhang and Chan, 2020) and slow performance (Nair and Dorai, 2021). Thus PoS and algorithms similar to PoS are the most popular on our list.

Developers should understand the pros and cons of the Sybil control mechanism. We suggest developers read their selected project's whitepaper to understand how a specific blockchain development platform implements the Sybil control mechanism on their blockchains.

Some blockchains, such as Hyperledger Fabric, allow the user to pick which mechanism the network will use when the user creates the network. This modular Sybil control mechanism layer may be desirable to developers who require specific control mechanisms for a given application.

Before starting a project, developers should know if a blockchain requires a framework-specific programming language for developing smart contracts. Solidity is the most well-known framework-specific programming language, created initially to develop Ethereum dapps. Table 6 shows all the languages available for each project. Since smart contracts can be responsible for handling financial transactions, developers should make sure they understand the quirks of any language they choose. A small mistake in code can easily lead to a significant financial loss for the dApp's developers and users.

Lastly, permissionless blockchains are governed by cryptocurrencies. Developers must pay a fee to run their applications on a permissionless blockchain. Some platforms have multiple cryptocurrencies. NEO has $NEO for participating in PoS and $GAS for paying the block creation fees. Since cryptocurrency prices are volatile, we do not report them here; however, developers should research the cost of creating dApps on their selected platform.

Developers should understand the fundamentals of each blockchain development platform before starting a project. Developers should ensure that the platform aligns with their project's goals. For example, a developer should select a permissionless blockchain if they plan to create an application that anyone can access. Understanding the fundamentals will aid developers in picking the best platform for their projects.

5.3. Existing applications

Table 4 analyzes the various applications built using each blockchain development platform. Before starting an application, developers should be aware of the types of projects each platform is designed for. While many platforms are designed for general applications, some have particular use cases.

The most common application area of blockchain is in FinTech; however, blockchain has many use cases in other areas. Many private blockchain platforms, such as HyperLedger Fabric, focus on allowing users to create secure enterprise applications. Likewise, some blockchain development platforms, such as Exonum, have focused on allowing users to create government applications. Stellar is focused on interbank transfers, a particular FinTech use case. Developers must know which platforms are designed to support their projects sector.

Likewise, some blockchain development platforms focus on connectivity with existing blockchains. Optimism and Harmony are L2 blockchains; thus, they both focus on compatibility with the Ethereum mainnet blockchain. Similarly, Stacks and Qtum allow developers to create dApps that interact with the Bitcoin blockchain. Developers building applications that need to interact with these blockchains should consider what support is available within their chosen development platform.

To help developers best understand what types of applications they can build on each platform, we list some common use cases and some popular dApps for each project. Some unique projects include the F1 team Red Bull Racing using Tezos for team management and the country of Georgia using Exonum for land registry. While it is out of the scope of this work to discuss each of these dApps, developers are encouraged to research existing dApps. In particular, dApp code is available for any developer to read on public blockchains, which may help new developers understand common coding paradigms.

Developers should familiarize themselves with some existing applications on each platform. This will help developers understand what applications are possible on each platform. For example, if a developer wishes to create an IoT application, they should consider platforms that are designed for IoT. The synergy between the platform and the project can make applications easier to build and more accessible to users.

5.4. Development considerations

Table 5 highlights key areas developers should understand before starting a project. The table covers the privacy features of each platform, the documentation available to developers, if there is a test network for developers, if there are tools available for developers, and how much control developers have over the Sybil control mechanism.

In addition to this paper, we have created a GitHub repository (Connors, 2023), where we have posted links to the projects. Our GitHub repository includes links to the project's documentation, learning resources, tools such as testing frameworks, and community resources.

We assigned a privacy score to each platform. The privacy score lets developers know the privacy features available on each platform. Business applications may require more privacy than general applications such as games. Thus, developers must select a platform that will allow for the appropriate level of privacy. A project received 1 point if it allowed for private transactions, 2 points for Zero-Knowledge Proofs (ZKP), and 4 points if it allowed developers to create their own private blockchain. A ZKP is a way for two actors to interact with each other without giving up their identities. ZCash (Ben Sasson et al., 2014) is a popular cryptocurrency that implements ZKP to allow for private blockchain transactions. For detailed information on ZKP, see Goldreich and Oren (1994) and Sun et al. (2021).

Since we assign points as powers of two, a privacy score can easily be decomposed into its parts by converting the score to binary. For example, Tron scored 5 points, which in base 2 is bin(101). Reading the binary from left to right, we see that it does allow for the creation of private blockchains (bin(100)), it does not have support for ZKP (bin(00)), and it does support private transactions (bin(1)).

Using a similar point system, we assign each project a documentation score. This metric aims to show developers how much support is available from the platform. A project was given 1 point for having official written documentation, 2 points for having a simple tutorial on how to create an application, 4 points if the documentation showed developers how to build a complete example application, 8 points if the documentation included videos, and 16 points if the platform provided extensive documentation. Some examples of extensive documentation include Ethereum and Tezos having games developers can play to learn the basics of the platform or EOS having an official training course developers can take.

The next column notes if the platform has a test network. Like in Web2 development, developers often only want to deploy their code to production once they are done testing it. A test network is a blockchain where developers can test their dApps. Like all other blockchains, a testnet can be either public or private. A private testnet will require developers to create their own environment for testing.

Often a faucet will accompany a public test. A faucet will give users cryptocurrency to pay block creation fees on a public testnet. The cryptocurrency given by the faucet is different from the cryptocurrency for the mainnet. Likewise, faucets rely on community support and can become dry; this may lead to issues for developers hoping to test on public testnets.

Next, we note which platforms have official development tools. Development tools are created by a blockchain development platform to assist developers in creating dApps. For example, Ethereum has the Remix IDE, an online IDE, and many frameworks for testing dApps. Developers should be aware of development tools available that can assist them in creating dApps.

Lastly, we cover the modularity of each project's Sybil Control layer, often called the consensus layer. Developers may need to optimism the Sybil control mechanism to best fit their application. Few platforms allow developers the flexibility to change the Sybil control layer. Hyperledger Fabric and Sawtooth allow developers to select from a set of Sybil control mechanisms (Shown in Table 3). Likewise, developers can modify part of the Sybil control mechanism if they create a private EOS network. Notice that no other platforms, including the platforms for private blockchains, allow developers to modify the Sybil Control layer, making the Hyperledger platforms and EOS unique.

Table 1
We considered 65 platforms and selected the 23 most popular platforms for our analysis. Dates marked with a * represent packages that are forks of older packages. A full list of the 65 considered platforms is available on our GitHub repository.

Platform	Github package name	Initial commit date	White paper
Ethereum	ethereum/go-ethereum	Dec 2013	Buterin (2014)
Hyperledger Fabric	hyperledger/fabric	May 2016	Androulaki et al. (2018), Hyperledger Foundation (2018)
EOS	EOSIO/eos	Apr 2017	Larimer et al. (2018)
Solana	solana-labs/solana	Feb 2018	Yakovenko (2018)
Tendermint	Tendermint/Tendermint	Apr 2014	Kwon (2014)
Quorum	ConsenSys/quorum	Dec 2013*	ConsenSys (2018)
Corda	corda/corda	Nov 2015	Brown (2018), Hearn and Brown (2019)
Neo	neo-project/neo	May 2015	Anon (0000a)
Tron	tronprotocol/java-tron	Dec 2017	TRON (2018)
Stellar	stellar/stellar-core	Nov 2014	Mazieres (2016)
Stacks	stacks-network/ stacks-blockchain	Jan 2014	Ali (2020)
BNB Chain	bnb-chain/bsc	Dec 2013*	Bnb-Chain (2020)
Hyperledger Sawtooth	hyperledger/ sawtooth-core	Mar 2016	Olson et al. (2018), Hyperledger Foundation (2018)
NEAR	NEAR/nearcore	Oct 2018	NEAR Foundation (2022)
Tezos	tezos/tezos	Sep 2016	Goodman (2014)
Optimism	ethereum-optimism/ optimism	Sep 2020	Optimism Team (2023)
IoTeX	iotexproject/iotex-core	Apr 2018	The Iotex Team (2018)
Harmony	harmony-one/harmony	May 2018	The Harmony Team (2019)
Waves	wavesplatform/Waves	Jan 2015	Waves Enterprise (2022)
Algorand	algorand/go-algorand	Jun 2019	Chen and Micali (2016)
Qtum	qtumproject/qtum	Aug 2009*	Qtum Org (2020)
Exonum	exonum/exonum	Apr 2016	Yanovich et al. (2018)
Zilliqa	Zilliqa/Zilliqa	Dec 2017	Anon (2017)

Developers should consider the privacy features and Sybil control layers' flexibility to ensure their selected platform supports their project. Likewise, newer developers may consider platforms with plenty of documentation, an easy-to-use testnet, and development tools. Developers can best select a platform that fits their skill level and project goals using our table.

5.5. Smart contract features

The following tables highlight the features of smart contracts provided by each platform. Table 6 shows which languages developers can use to write smart contracts. Languages marked with a * are languages designed for smart contract development.

Table 2
This table shows various popularity metrics for our selected platforms. Popularity Score is the normalized sum of the Forks, Stars, Stack Exchange Questions, and Google Scholar Articles.

Project	Github stars	Github forks	Closed Github issues	Stack exchange	Google scholar	Dapps	Popularity score
Ethereum	39163	14840	6391	6317	18500	3452	100.00
Hyperledger Fabric	13950	8190	162	1502	2310	–	32.93
EOS	11346	3700	4751	81	275	579	19.54
Solana	9372	2329	3491	383	78	105	15.43
Tendermint	5142	1786	2800	68	138	–	9.05
Quorum	4234	1184	757	206	379	–	7.62
Corda	3878	1069	579	500	225	–	7.20
Neo	3381	1000	1020	32	194	1	5.84
Tron	3132	1139	1030	121	43	1397	5.63
Stellar	3000	992	1157	33	190	–	5.35
Stacks	2688	545	1749	219	20	–	4.40
BNB Chain	1685	788	652	86	0	4067	3.25
Hyperledger Sawtooth	1393	762	4	113	135	–	3.05
NEAR	1822	364	2064	170	13	43	3.01
Tezos	1501	209	0	46	212	80	2.50
Optimism	1397	450	434	4	4	25	2.35
iotex	1431	291	987	1	18	45	2.21
Harmony	1447	268	971	12	12	117	2.21
Waves	1164	419	343	67	72	16	2.18
Algorand	1132	346	5	21	112	12	2.04
Qtum	1162	389	292	11	28	–	2.02
Exonum	1176	245	340	5	57	–	1.88
Zilliqa	1110	263	264	1	15	–	1.76

Our list of platforms contains a variety of languages, with the most common languages being Javascript and Solidity. Very few platforms allow for mobile smart contracts. Only Stellar and Hyperledger Sawtooth allow developers to use mobile languages like Swift.

Framework-specific languages, such as Solidity, offer the benefit of being designed for blockchain development at the cost of having developers learn a new programming language. When selecting a platform, developers should recognize the pros and cons of using a framework-specific language. For example, take a developer who wants to create a financial application. A small error in the code could lead to a significant financial loss. On the one hand, the developer may want to program in a language they already know since they already understand the nuances of the language, but they may have a blockchain-related bug in their code. On the other hand, they may choose a framework-specific language since it will help the developer avoid blockchain errors. Still, the developer must learn the language and understand its nuances.

Table 7 shows which platforms are directly upgradable after deployment. Since smart contracts are written on a blockchain, which is immutable, developers can never change the original code after being published. However, some platforms have systems to ensure that developers can update their code, and the old code is ignored. While developers have been able to replicate these systems on most platforms, only a few of our selected platforms allow developers to update their code directly after deployment.

Some of our selected platforms allow developers to create Ricardian Contracts. While smart contracts are often compared to legal agreements, they are not legal contracts. A Ricardian Contract is a legal contract that machines and people can read (Grigg, 2004). Developers wishing to create applications requiring legal contracts should consider a platform that allows for Ricardian Contracts.

Lastly, we highlight the implementation of smart contracts on each platform. The execution model shows when transactions are ordered and when the smart contract is executed. For example, in the Order-Execute model, transactions are first ordered, then the code is executed. In contrast, in the Execute-Order-Validate model, transactions are executed, and the output of those transactions is then ordered into a block. The Hyperledger Fabric Whitepaper (Androulaki et al., 2018) details each type of execution model.

Likewise, we note the OS model behind each blockchain development platform. Most of our platforms run on a distributed virtual machine, such as the Ethereum Virtual Machine (EVM). Each project's whitepaper contains information on how its virtual machine is implemented.

Lastly, we show the compiled language that each model uses. While developers do not need to be able to write code in the compiled language, it can be helpful to understand the entire model when debugging. In traditional applications, developers do not need to know how to write in assembly to create apps. However, understanding that the system will compile their program to assembly can make debugging easier for developers. The same idea applies to Web3 development; developers should understand how their applications are compiled so they can more easily debug their code.

By understanding the languages allowed by each platform, developers can best select a platform whose language they are comfortable with. Likewise, developers should understand what features the smart contracts provide to ensure they select a platform that synergies with their project. Using this table, developers can understand how smart contracts work on each platform to select a platform that fits their projects' needs.

5.6. Community engagement

Table 8 summarizes the community around each blockchain development platform. Developers should be aware of the size and engagement of a community so they know where to turn when they need technical assistance. Likewise, since blockchain focuses on decentralization, the community is often the decision-maker for substantial changes to the platform.

We first recorded who the supporters of the blockchain development platform are. Blockchain development platforms supported by companies will have different end goals than platforms supported by individuals or foundations. Developers should keep in mind the goals of the development platform backers.

Table 3
This table shows key properties of our selected blockchain development platforms. The Prog. Lang. column represents if the project requires a framework-specific language, such as Solidity.

Platform	Generation	Permissionless	Consensus	Prog. Lang.	Crypto-currency
Ethereum	2.0	Yes	PoS	Yes	$ETH (Ether)
Hyperledger Fabric	2.x	No	Kafka, Solo, BFT-SMaRt	No	–
EOS	2.x	Yes	aBFT + DPoS	No	$EOS (EOSIO)
Solana	2.x	Yes	PoH (Similar to PoS)	No	$SOL (Solana)
Tendermint	2.x	No	pBFT	No	–
Quorum	2.x	No	PoA	Yes	–
Corda	2.x	No	Notary Nodes	No	–
Neo	3.x	Yes	dBFT (Similar to PoS)	No	$NEO (Neo) $GAS (NeoGas)
Tron	2.x	Yes	DPoS	Yes	$TRX (TRON)
Stellar	2.x	Yes	FBA (Stellar)	No	$XML (Lumen)
Stacks	2.x	Yes	PoX (Proof of Transfer)	Yes	$STX (Stacks)
BNB Chain	2.x	Yes	pBFT & PoSA	Yes	$BNB (BNB)
Hyperledger Sawtooth	2.x	No	PoET, Raft	No	–
NEAR	2.x	Yes	Nightshade	No	$NEAR (NEAR)
Tezos	2.x	Yes	PoS	No	$XTZ (Tezos)
Optimism	3.x	Yes	DPoS	Yes	$OP (Optimism)
iotex	3.x	Yes	DPoS	Yes	$IOTX (IoTeX)
Harmony	3.x	Yes	pBFT	Yes	$ONE (Harmony)
Waves	2.x	Yes	LPoS	Yes	$WAVES (Waves)
Algorand	2.x	Yes	PPoS	No	$ALGO (Algorand)
Qtum	2.x	No	MPoS	No	$QTUM (Qtum)
Exonum	2.x	No	pBFT	No	–
Zilliqa	2.x	Yes	pBFT	Yes	$ZIL (Zilliqa)

Next, we record when each platform last made a commit to GitHub. We recorded our dates in September 2022. In addition, we recorded the total number of commits. These two metrics show how active a blockchain development platform community is. When we recorded our data, only EOS had yet to make a commit within the last year. Similarly, we record if the platform has an official road map. The road map outlines future changes the platform plans to make and when it aims to make them. Developers should know how active a platform is and what updates are planned. Some applications will require no modifications to the platform after deployment, while others will benefit from continued upgrades.

We measured how active each platform's community is on social media. We note which platforms have an official Telegram or Discord channel. These are popular messaging applications where developers can join a group and ask others in the community questions about the platform. Likewise, we recorded how many YouTube videos were posted by the platform. Lastly, we recorded the number of members on each platform's subreddit. A subreddit is a forum on the social media platform Reddit dedicated to a specific topic. The platforms tied to a cryptocurrency have significantly more subreddit members than platforms without a cryptocurrency. This disparity is so drastic that the least popular platform Zilliqa, linked to the $ZIL cryptocurrency, has more subreddit members than our second most popular project Hyperledger Fabric, which has no cryptocurrency.

Developers should use our community table to judge how active a platform's community is to understand how much support will be available. Projects with a larger community will have more people to assist a new developer. Likewise, developers should understand how active the community is and who the stakeholders are so they know where the project is going and when they can expect the new features to be implemented.

6. Discussion

In addition to our tables we have also created a GitHub repository containing links to each projects website, whitepaper, tools, learning resources, community, and sample dApps. This GitHub page can be found here (Connors, 2023)

Our first table, Table 1, list our selected projects. We only selected projects that met certain popularity requirements to ensure that all of our selected projects were well-known enough that new developers

Table 4
This table shows examples of existing applications for our selected blockchain development platforms. Links to the example applications can be found in our GitHub Repository.

Project	Application area	Example use cases	Sample DApps
Ethereum	FinTech, general dApps	Decentralized Exchange, NFTs, Games	Uniswap, Crypto Kitties
Hyperledger Fabric	GovTech, Enterprise Applications, Healthcare	Voteing, B2B Supply Chains	BRUINchain, Healthchain
EOS	general dApps, Enterprise Applications	Business Applications, Games, Trading Digital Assets	Defibox, Upland
Solana	general dApps, FinTech	Gaming, DAO, DeFi Payment	Magic Eden, Gameta
Tendermint	FinTech	DEX, StableCoin, Carbon Trading	Terra, Regan Network
Quorum	FinTech	Interbank Transfers, Marketplace for Loans	Project Ubin, Skeps
Corda	FinTech, Healthcare, Construction	Capital Markets, Claims Management, Supply Chain	HSBLOX
Neo	FinTech, IoT	Lottery, Social Network	Effect, Naritive
Tron	FinTech, general dApps	Decentralized Exchange, NFTs, Games	SunSwap, BSG
Stellar	FinTech	Interbank Transfers	IBM BWW
Stacks	FinTech, Bitcoin dApps	Decentralized Exchange, Domain Registrar	ALEX, Gamma
BNB Chain	FinTech, general dApps	Decentralized Exchange, NFTs, Games	PancakeSwap, Era7
Hyperledger Sawtooth	Healthcare, Enterprise Applications	EHR, Enterprise Management	PokitDok, Sextant
NEAR	FinTech, general dApps	Decentralized Exchange, NFTs, Games	Paras, Burrow
Tezos	FinTech, GovTech, general dApps	Stable Assets, Team Management	lugh, Red Bull Racing
Optimism	Fintech,general dApps	DeFi, Marketplaces, Exchanges	Quixotic, Across
IoTeX	IoT	IoT Management, IoT data sharing	StarCrazy, VITA
Harmony	Fintec, general dApps	DeFi, NFTs, Decentralized Exchanges	Sushi, Timeless
Waves	general dApps	NFTs, Games, DeFi	Viers.finance, Waves Ducks
Algorand	FinTech, general dApps	Decentralized Exchange, NFTs, Games	Algodex, Algofi
Qtum	FinTech, Bitcoin dApps	Creating Tokens, Travel	QRC20 Tokens, Travala
Exonum	GovTech, dApps	Property Management, eAuctions	Land Registry in Gerorgia, Ukrain eAuction platform
Zilliqa	FinTech, dApps	ePayments, Advertisement	Xfers, Aqilliz

could easily find resources to help them learn the platform. We considered 65 platforms, and our popularity restraint narrowed our analysis to just 23 platforms. A complete list of the 65 considered platforms can be found on our GitHub repository.

We then further analyzed our platforms to determine the most popular platform. We assigned each platform a popularity score, Eq. (1), based on its GitHub engagement, StackExchange engagement, and its academic popularity. The popularity of each project is shown in Table 2. The popularity and the community support shown in Table 8 give developers an idea of how much support is available. New blockchain developers often need guided support to get started. A popular project with a large community gives new developers community resources to help them learn a platform. We suggest developers who learn best through community resources such as Youtube Tutorials or example applications learn popular platforms with large communities.

Next, we looked at the properties of the selected blockchain development platforms. Table 3 shows these fundamental properties of the blockchain development platforms. New developers should note which platforms require learning a new language to write smart contract code. We provide in-depth development language details in Table 6. Likewise, developers must be aware of their development platforms' permission structure. Developers wishing to create public projects, such as a blockchain-based game, must ensure they use a permissionless blockchain.

To give developers an idea of which platform might best suit their projects, we provide examples of existing applications in Table 4. Links

Table 5
This table highlights development features of our selected blockchain development platforms. Links to each projects documentation, test networks, and development tools are on our GitHub Repository.

Platform	Privacy score	Documentation score	Test network	Development tools	Sybil modularity
Ethereum	1	31	Public	Yes	None
Hyperledger Fabric	7	7	Private	Yes	High
EOS	5	31	Public	Yes	Low
Solana	0	7	Public	Yes	None
Tendermint	5	7	Private	Yes	None
Quorum	7	7	No	Yes	None
Corda	7	15	Public	No	None
Neo	4	3	Public/Private	Yes	None
Tron	5	3	Public	Yes	None
Stellar	0	15	Public	Yes	None
Stacks	0	3	Public	Yes	None
BNB Chain	2	7	Public	Yes	None
Hyperledger Sawtooth	5	3	Public	Yes	High
NEAR	7	7	Public/Private	No	None
Tezos	3	19	No	Yes	None
Optimism	4	3	Public/Private	Yes	None
IoTeX	0	23	Private	No	None
Harmony	3	23	Public	Yes	None
Waves	0	3	No	No	None
Algorand	0	23	Private	Yes	None
Qtum	0	7	Public	Yes	None
Exonum	5	3	Private	No	None
Zilliqa	0	3	Public	Yes	None

Table 6
This table shows which programming languages can be used to develop smart contracts for each platform. Languages marked with a * are blockchain specific languages.

Platform	Smart contract development language
Ethereum	Solidity*, Viper*
Hyperledger Fabric	Go, Java, Javascript, Typescript
EOS	C++
Solana	Rust, C++, C
Tendermint	Go, Python, Cosmos CLI*
Quorum	Solidity*
Corda	Kotlin, Java
Neo	Python, C#, Go, Java, Javascript, Typescript
Tron	Solidity*
Stellar	Javascript, Java, Go, Python, C#.NET, Ruby, IOS, Scala, Qt/C++, Flutter
Stacks	Clarity*
BNB Chain	Solidity*
NEAR	Javascript, Rust, NEAR API
Hyperledger Sawtooth	Python, Go, Javascript, Rust, Java, C++, Swift
Optimism	Solidity*
Tezos	Python, OCaml, Javascript, Pascal, Reason, Indigo, Archtype, Michleson*
IoTeX	Solidity* , Javascript, Java, Go, Swift, C
Harmony	Solidity*
Waves	Ride*
Qtum	Javascript, Solidity*
Algorand	Python, Reach
Exonum	Rust, Java
Zilliqa	Scilla*

to the example applications can be found on our GitHub repository. Developers should use this table to determine if a platform synergizes with their project. For example, developers wishing to create a fintech application should consider platforms designed to support fintech.

Next, we review various development considerations in Table 5. We recommend developers review each platform's documentation and official support, which we have provided on our GitHub page (Connors, 2023). Developers should pick a platform that they understand and are confident they can work with. Some platforms have a suite of tools to assist developers in creating applications. Developers who like using these IDEs and Testing Frameworks should aim to use a platform with a diverse toolset. Links to the official tools provided by each platform can be found on our GitHub page.

Lastly, developers should consider the features of their smart contracts. Understanding how smart contracts are executed and deployed is critical to creating secure code. In particular, developers creating fintech applications should pay close attention to how smart contracts are executed on a blockchain platform. A small mistake in their implementation can lead to large financial losses.

In addition to our table, we pose a few questions developers should ask themselves before choosing a blockchain platform. These questions can help new developers pick the best blockchain development platform.

First, developers should ask themselves if they need a permissionless or a permissioned blockchain. If a developer plans on only having a few select users, for example, in a business application, then a permissioned platform may be more appropriate. A permissionless platform will better suit the project if a developer plans to allow anyone to interact with their project. Developers should ask themselves questions such as

Table 7
This table highlights features of smart contracts on our selected blockchain development platforms.

Platform	Directly upgradable	Ricardian contracts	Execution model	OS model	Compiled language
Ethereum	No	No	Order-Execute (OE)	Ethereum Virtual Machine	Ethereum Bytecode
Hyperledger Fabric	Yes	Yes	Execute-Order-Validate (EOV)	Hyperledger OS	–
EOS	Yes	Yes	OE	EOSIO Virtual Machine	WebAssembly
Solana	Yes	No	OE: Parallel Execution	Solana Virtual Machine	Solana Bytecode
Tendermint	No	No	OE	ABCI Protocol	ABCI
Quorum	No	No	OE	Ethereum Virtual Machine	Ethereum Bytecode
Corda	Yes	Yes	If Then Else (ITE)	Corda API	REST API
Neo	Yes	No	OE	Neo Virtual Machine	Neo Bytecode
Tron	No	No	OE	Tron Virtual Machine	Ethereum Bytecode
Stellar	No	No	ITE	Stellar API	REST API
Stacks	No	No	EOV	Clarity Interpreter	–
BNB Chain	No	No	OE	Ethereum Virtual Machine	Ethereum Bytecode
Hyperledger Sawtooth	Yes	No	OE	Sawtooth State Machines	WebAssembly
NEAR	No	No	OE	NEAR Virtual Machine	WebAssembly
Tezos	No	No	OE: Functional Programming	Michelson Functional Programming	Michelson
Optimism	No	No	OE	Optimism Virtual Machine	Ethereum Bytecode
IoTeX	No	No	OE	Ethereum Virtual Machine	Ethereum Bytecode
Harmony	No	No	OE	Ethereum Virtual Machine	Ethereum Bytecode
Waves	No	No	ITE	Waves API	REST API
Algorand	Yes	No	OE	Algorand Virtual Machine	TEAL
Qtum	No	No	OE	x86 Virtual Machine	x86 Assembly
Exonum	Yes	No	OE	Java Virtual Machine	Java Bytecode
Zilliqa	No	No	EOV	Scilla Interpreter	–

"Should anyone be allowed to interact with the smart contract?" or "Dose smart contract code need to be confidential?"

Similarly, developers should consider if they need a public mainnet. Developers should understand the fees and transaction speed using a public mainnet. In contrast, if developers choose to create their own private mainnet, they should understand the cost of creating and maintaining that network. If a business is tracking shipments, it will likely want to keep that data hidden from competitors; thus, the developer must create a private network. In contrast, if a developer is not storing sensitive information on the blockchain, a public network may be more accessible. Questions such as "Do I want to use an existing blockchain infrastructure?", "What is the maximum that any execution of my smart contract can cost?" are essential in making this decision.

Next, developers must ensure the platform allows them to create their projects. For example, a developer wishing to create a banking application should use a platform others have used to create similar applications. The developers should consider a few platforms for any project since one platform might be better optimized for the developer's project. For example, a developer may wish to use Ethereum for a banking application. However, they should also consider platforms like Stellar since it is more optimized for this type of project. "Have similar applications been created using this blockchain development platform?" and "Which blockchain development platform specializes for my use case?" can help developers choose the best platform.

Developers should ensure they are comfortable with the language smart contracts are developed. Knowing the nuances of the language are particularly important when creating financial applications, as a mistake in the code can cause financial harm. To start quickly, developers may wish to write in languages they already know. However, if developers spend time learning a framework-specific language designed for smart contract development, it may make more complicated blockchain tasks easier to program. Developers should consider, "Do I know the language for smart contract development already?" "Am I willing to learn a new blockchain-specific language?"

Similarly, developers need to understand what types of development tools are available. The development tools can help developers create, test, and maintain smart contracts. Developers should understand "What tools exist to help developers?" and "What type of support is there for new developers?" Lastly, developers should understand the size of the project and the amount of community engagement surrounding a project. A large active community can help developers with technical issues when creating their dapps. Developers should

Table 8
This table highlights the communities surrounding our selected blockchain development platforms. The last commit data was collected in September 2022.

Platform	Supporters	Last commit	Total commits	Road map	Official Telegram/Discord	Youtube videos	Subreddit
Ethereum	Vitalik Buterin, Etherium Foundation	Sep 2022	13637	Yes	Yes	1094	1444724
Hyperledger Fabric	Linux Foundation, IBM, Intel, SAP Arabia	Sep 2022	14130	No	Yes	1169	3550
EOS	Block.one, EOSIO Foundation	July 2021	20554	Yes	Yes	103	97768
Solana	Anatoly Yakovenko, Solana Foundation	Sep 2022	20203	Yes	Yes	314	152831
Tendermint	Jae Kwon, Cosmos/Ignite	Sep 2022	8956	Yes	Yes	359	274
Quorum	ConsenSys, JP Morgan	Sep 2022	14311	No	No	83	2131
Corda	R3	Sep 2022	9732	Yes	No	257	469
Neo	Da Hongfei, Erik Zhang, Onchain	Sep 2022	1388	Yes	No	148	117601
Tron	Justin Sun, TRON DAO	Sep 2022	17087	No	Yes	559	124952
Stellar	Jed McCaleb, Joyce Kim, Stellar Development Foundation	Sep 2022	7990	Yes	Yes	143	212769
Stacks	Princeton, Hiro, Stacks Foundation	Sep 2022	16993	No	Yes	494	6652
BNB Chain	Binance	Sep 2022	13506	Yes	Yes	135	17192
Hyperledger Sawtooth	Linux Foundation, IBM, Intel, SAP Arabia	Jul 2022	4590	No	Yes	1169	3550
NEAR	Illia Polosukhin, Alexander Skidanov, NEAR Foundation	Sep 2022	8251	Yes	Yes	378	12254
Tezos	Dynamic Ledger Solutions, Tezos Foundation	Sep 2022	3779	No	Yes	171	71141
Optimism	Optimism Foundation	Sep 2022	19477	Yes	Yes	17	5
IoTeX	IoTeX	Sep 2022	2831	Yes	Yes	360	18044
Harmony	Stephen Tse, Harmony Foundation	Sep 2022	7590	Yes	Yes	406	53477
Waves	Waves Technology	Sep 2022	12971	No	Yes	86	1484
Algorand	Algorand	Sep 2022	34919	No	Yes	183	75645
Qtum	Qtum	Jun 2022	3270	Yes	Yes	98	17679
Exonum	Bitfury	Sep 2022	5652	Yes	Yes	7	46
Zilliqa	National University of Singapore, Zilliqa Foundation	Sep 2022	9383	Yes	Yes	132	44459

consider questions such as "How popular is the blockchain development platform?" or "Is the community large enough to assist new developers?"

7. Conclusion

In this work, we first outline the basics of blockchain so that developers can understand what differentiates various blockchain development platforms. Next, we discuss selecting 23 blockchain development platforms from our extensive list. Then, we analyzed seven tables highlighting the key metrics developers should know for 23 selected platforms. We finished with a brief discussion on how developers can effectively use our metrics when creating new projects. We hope developers use our metrics to select the best project platform.

In future work, we aim to create the same sample project on all our selected platforms and post those projects on GitHub. This GitHub page will allow developers to look at code to understand each project's nuances better. We hope that the addition of code will assist new developers in selecting a platform and quickly getting started on projects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

Agbo, Corneilus, Mahmoud, Qusay, 2019. Comparison of blockchain frameworks for healthcare applications. Internet Technol. Lett. 2 (5), e122.
Ali, Muneeb, 2020. Stacks 2.0 apps and smart contracts for Bitcoin. Gaia.Blockstack.Org.
Androulaki, Elli, Barger, Artem, Bortnikov, Vita, Cachin, Christian, Christidis, Konstantinos, De Caro, Angelo, Enyeart, David, Ferris, Christopher, Laventman, Gennady, Manevich, Yacov et al., 2018. Hyperledger fabric. In: Proceedings of the Thirteenth EuroSys Conference.
Anon, 0000a. Neo White Paper. Neo Website.
Anon, 0000b. Lisk Website. Lisk White Paper.
Anon, 2017. The ZILLIQA technical whitepaper. Zilliqa.

Ben Sasson, Eli, Chiesa, Alessandro, Garman, Christina, Green, Matthew, Miers, Ian, Tromer, Eran, Virza, Madars, 2014. Zerocash: Decentralized anonymous payments from Bitcoin. In: 2014 IEEE Symposium on Security and Privacy.
Bnb-Chain, 2020. BNB-chain whitepaper. GitHub.
Brown, Richard Gendal, 2018. The Corda Platform: An Introduction.
Buterin, Vitalik, 2014. Ethereum Whitepaper.
Chen, Jing, Micali, Silvio, 2016. Algorand. http://dx.doi.org/10.48550/ARXIV.1607.01341, arXiv.
Connors, Collin, 2023. https://github.com/CollinConnors/Survey-Of-Prominent-Blockchain-Development-Platforms.
Connors, Collin, Sarkar, Dilip, 2022a. Comparative study of blockchain development platforms: Features and applications.
Connors, Collin, Sarkar, Dilip, 2022b. Review of most popular open-source platforms for developing blockchains. In: 2022 Fourth International Conference on Blockchain Computing and Applications. BCCA.
ConsenSys, 2018. Quorum Whitepaper.
David, Bernardo, Gazi, Peter, Kiayias, Aggelos, Russell, Alexander, 2023. Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain.
Dernayka, Iman, Chehab, Ali, 2021. Blockchain development platforms: Performance comparison. In: 2021 11th IFIP International Conference on New Technologies, Mobility and Security. NTMS, pp. 1–6.
Goldreich, Oded, Oren, Yair, 1994. Definitions and properties of zero-knowledge proof systems. J. Cryptol. 7 (1), 1–32.
Goodman, L.M, 2014. Tezos - a self amending crypto-ledger White paper.
Grigg, I., 2004. The Ricardian contract. In: Proceedings. First IEEE International Workshop on Electronic Contracting, 2004. pp. 25–31.
Guo, Huaqun, Yu, Xingjie, 2022. A survey on Blockchain technology and its security. Blockchain: Res. Appl. 3 (2), 100067.
Hearn, Mike, Brown, Richard Gendal, 2019. Corda: A distributed ledger.
Hyperledger Foundation, 2018. Introduction to Hyperledger, Hyperledger.
Iqbal, Mubashar, Matulevičius, Raimundas, 2021. Exploring sybil and double-spending risks in blockchain systems. IEEE Access 9, 76153–76177.
Javaid, Mohd, Haleem, Abid, Pratap Singh, Ravi, Khan, Shahbaz, Suman, Rajiv, 2021. Blockchain technology applications for Industry 4.0: A literature-based review. Blockchain: Res. Appl. 2 (4), 100027.
Kwon, Jae, 2014. Tendermint: Consensus without mining. Tendermint.Com.
Larimer, Daniel, et al., 2018. eos.io technical white paper v2.
Mazieres, David, 2016. The Stellar Consensus Protocol: A Federated Model for Internet-level Consensus.
Mingxiao, Du, Xiaofeng, Ma, Zhe, Zhang, Xiangwei, Wang, Qijun, Chen, 2017. A review on consensus algorithm of blockchain. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics. SMC, pp. 2567–2572.
Nair, P. Rajitha, Dorai, D. Ramya, 2021. Evaluation of performance and security of proof of work and proof of stake using blockchain. In: 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks. ICICV, pp. 279–283.
Nakamoto, Satoshi, 2008. Bitcoin: A peer-to-peer electronic cash system. SSRN Electronic J..
NEAR Foundation, 2022. The NEAR white paper. NEAR Protocol.
Nguyen, Giang-Truong, Kim, Kyungbaek, 2018. A survey about consensus algorithms used in blockchain. Koreascience.Or.Kr.
Nofer, Michael, Gomber, Peter, Hinz, Oliver, Schiereck, Dirk, 2017. Blockchain. Bus. Inform. Syst. Eng. 59 (3), 183–187.
Olson, Kelly, Bowman, Mic, Mitchell, James, Amundson, Shawn, Middleton, Dan, Montgomery, Cian, 2018. Sawtooth: An Introduction, Hyperledge Org.
Optimism Team, 2023. How Optimism Works. Optimism Docs.
Polge, Julien, Robert, Jérémy, Le Traon, Yves, 2021. Permissioned blockchain frameworks in the industry: A comparison. ICT Express 7 (2), 229–233.
Popov, Serguei, 2018. The Tangle.
Qtum Org, 2020. Qtum Blockchain New Whitepaper.
Quasim, Mohammad Tabrez, Khan, Mohammad Ayoub, Algarni, Fahad, Alharthy, Abdullah, Alshmrani, Goram Mufareh M., 2020. Blockchain frameworks. In: Khan, Mohammad Ayoub, Quasim, Mohammad Tabrez, Algarni, Fahad, Alharthi, Abdullah (Eds.), Decentralised Internet of Things: A Blockchain Perspective. Springer International Publishing, Cham, pp. 75–89.
Sekniqi, Kevin, Laine, Daniel, Gun Sirer, Emin, 2020. Avalanche platform whitepaper, Assets.Website-Files.Com.
Sun, Xiaoqiang, Yu, F. Richard, Zhang, Peng, Sun, Zhiwei, Xie, Weixin, Peng, Xiang, 2021. A survey on zero-knowledge proof in blockchain. IEEE Netw. 35 (4), 198–205.
Tavares, Bruno, Correia, Filipe, Restivo, André, 2019. A Survey on Blockchain Technologies and Research. Vol. 14. pp. 118–128.
The Harmony Team, 2019. Harmony Technical Whitepaper. Harmony.One.
The Iotex Team, 2018. IoTeX A decentralized network for inter of things powered by a privacy-centric blockchain. The Whitepaper Database.
TRON, 2018. Tron whitepaper. Whitepaper.Io - Search and Find All Whitepapers on Whitepaper.Io.
Valenta, Martin, Sandner, Philipp, 2017. Comparison of ethereum, hyperledger fabric and corda. Smallake.Kr.
Velliangiri, S., Karthikeyan, P., 2020. Blockchain Technology: Challenges and Security issues in Consensus algorithm. In: 2020 International Conference on Computer Communication and Informatics. ICCCI, pp. 1–8.
Waves Enterprise, 2022. Enterprise-grade Hybrid Blockchain Platform.
Yaga, Dylan, Mell, Peter, Roby, Nik, Scarfone, Karen, 2018. Blockchain technology overview. http://dx.doi.org/10.6028/nist.ir.8202.
Yakovenko, Anatoly, 2018. Solana: A new architecture for a high performance blockchain v0.8.13. Solana.Com.
Yanovich, Yury, Ivaschenko, Ivan, Ostrovsky, Alex, Shevchenko, Aleksandr, Sidorov, Aleksei, 2018. Exonum: Byzantine fault tolerant protocol for blockchains.
Zhang, Rong, Chan, Wai Kin, 2020. Evaluation of energy consumption in block-chains with proof of work and proof of stake. J. Phys. Conf. Ser. 1584 (1), 012023.
Zou, Weiqin, Lo, David, Kochhar, Pavneet Singh, Le, Xuan-Bach Dinh, Xia, Xin, Feng, Yang, Chen, Zhenyu, Xu, Baowen, 2021. Smart contract development: Challenges and opportunities. IEEE Trans. Softw. Eng. 47 (10), 2084–2106.

Collin Connors is a Ph.D. candidate in Computer Science at the University of Miami. His research is focused on the security of emerging technologies, including blockchain, and applications of artificial intelligence in cybersecurity.

Dilip Sarkar is a professor of Computer Science at the University of Miami. His areas of research included Wireless communication, Multimedia transport over the Internet, Wireless Sensor Networks, distributed computing and parallel computing.