FERI

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Jaka Zavratnik

ANALYSIS OF WEB3 SOLUTION DEVELOPMENT PRINCIPLES

Master's Degree Thesis

Barcelona, October 2022

FERI

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Jaka Zavratnik

ANALYSIS OF WEB3 SOLUTION DEVELOPMENT PRINCIPLES

Master's Degree Thesis

Barcelona, October 2022

ANALYSIS OF WEB3 SOLUTION DEVELOPMENT PRINCIPLES

Master's Degree Thesis

Student:	Jaka Zavratnik
Study Programm:	Master's Degree
	Informatics and communication technologies
Mentor:	doc.dr. Muhamed Turkanović, UM FERI
Co-mentor:	dr. Jaime Delgado, UPC FIB

ACKNOWLEDGEMENTS

I would like to say a special Thank You to my family for their continuous support, especially to my parents, Andrej and Mojca, for investing in their son's education and much needed motivational words.

Secondly, I would like to thank my mentor, Muhamed Turkanović, for all the help, expertise, and guidance, and for suggesting this interesting topic related to blockchain, which made me really enjoy this journey.

I would also like to thank my co-mentor, Jaime Delgado, for his willingness and support provided in writing this thesis.

ANALYSIS OF WEB3 SOLUTION DEVELOPMENT PRINCIPLES

Keywords: decentralized app, blockchain, Ethereum, Web3

Abstract

In the master's thesis, we researched the principles of Web3 solution development. We studied the blockchain and blockchain-related technology, development of the Web including all versions of the Web and the differences between them. We presented the popular technologies for Web3 development and the most common Web3 solutions with examples. With help of systematic literature review we explored the state-of-art technologies for Web3 solution development and proposed a full-stack for Web3. In the final part we implemented a proof-of-concept Ethereum decentralized application and compared it with equivalent concept of Web2 application. We proposed future work of researching other popular blockchain protocols like Solana or Polygon.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

dapp - Decentralized Application

NFT - Non-fungible token

URL - Uniform Resource Locator

P2P - Peer-to-peer

DAO - Decentralized Autonomous Organization

ETH - Ether

EVM - Ethereum Virtual Machine

GOR - Goerli Ether

IPFS - InterPlanetary File System

HTML - Hypertext Markup Language

FTP - File Transfer Protocol

CERN - The European Organization for Nuclear Research

CA - Central Authority

NLP - Natural Language Processing

IoT - Internet of Things

AR - Artificial Reality

VR - Virtual Reality

DeFi - Decentralized Finance

DeX - Decentralized Exchange

SLR - Systematic Literature Review

CSS - Cascading Style Sheets

API - Application Programming Interface

ABI - Application Binary Interface

1. INTRODUCTION

This thesis aims to study the principles of Web3 solution development, with its focus on the technologies, platforms, frameworks, and tools for developing solutions. With the help of a systematic literature analysis, we compare development approaches between different studies and answer research questions that relate to Web3 and Web3 solution development. We demonstrate the collected knowledge on a proof-of-concept application.

Within the theoretical background we introduce the blockchain and blockchain-related technology, which is the foundation for this work. Then, we write about development of the Web, where we describe each version of the Web and point out the differences between them. We present the popular technologies for Web3 development. In the last part of theoretical background, we present the most common Web3 solutions with examples.

In the second part of the thesis, we perform a systematic literature review, which is a way of synthesizing scientific evidence to answer the defined research questions in a transparent and reproducible way, while seeking to include all published evidence on the topic and appraising the quality of this evidence. The review is focused on identifying the technologies for Web3 solution development and to find out which technologies represent a full-stack for Web3. The data for performing the review is gathered from the official websites of digital libraries from scientific articles and conference papers.

In the last part, we present our implementation of a Web3 proof-of-concept application based on an identified stack of technologies from the previous chapters. We point out which technologies were used for the implementation and why we chose them. Then we present the concept for Web2 solution implementation of the same proof-of-concept application, and we discuss the differences between the two. Finally, we express our findings and propose future work in a conclusion.

1.1. Motivation

Bitcoin was released in public in year 2009 and since then blockchain technologies continued to show many advantages and applicability in different areas of industries. The biggest advancement in blockchain technologies are smart contracts, which are self-executable contracts containing the terms of agreement between two parts, without the need for an intermediary. These contracts allow developers to build autonomous, self-efficient and decentralized applications.

Blockchain technologies in the last years became incredibly popular, with much of this credit going to cryptocurrency and NFT trading success stories, where certain individuals made a significant amount of profit. This attracted the interest of big companies and investors, who began to invest a lot of money in blockchain projects. As a result, the intensive development of technologies, tools, frameworks, and platforms for the development of Web3 solutions began. Even though the technologies seem mature enough, the technology is still relatively new and a unique approach for the development of Web3 solutions has not yet been established.

Gartner's hype cycle for blockchain and Web3 for the year 2022 indicates that decentralized applications are currently on the slope of enlightenment and will reach the plateau of productivity in between 2 and 5 years. On the other hand, the Web3 is evaluated to be at the peak of inflated expectations and will require at least 5 to 10 years before reaching the plateau of productivity. Cryptocurrency and token values crashed in the first half of 2022, but the value of coins should not be conflated with the value of blockchain technologies. The NFT games and commerce are driving the innovation as companies begin to realize the business value. Gartner mentions there is still no killer use case to be seen, but gradual improvements using blockchain technology have been detected. [1]

According to Gartner's report, we can conclude that blockchain technology is still relatively new, but it is gradually improving, and more and more companies are

incorporating it in their businesses. With combination between Web3 and other technologies like AI and IoT, we could easily see a revolution in technology, and it is just a matter of time before we reach it.

1.2.Identification and problem definition

Nowadays, most online content and user data are under the control of a few large technology companies (e.g., Apple, Google, Amazon, Facebook). These companies collect users' data and then sell it, mostly in the form of user-tailored advertisements. While there is nothing wrong with the idea of suggesting user only the products or services, that might interest him, it is a major privacy breach when the company collects, stores, and takes advantage of e.g., search history of that user, usually without their consent. In current state of the Internet, the information is stored on single servers, and users have little or no control over their data ownership or use. In 2013, Edward Snowden exposed how the government in the United States was using these platforms to spy on their citizens. Another case of corruption happened in 2016 when during elections Facebook was deliberating suppressing conservative news stories. Lately, there is a big issue of platform owners delegating censorship on their platforms, i.e., they ban people off their platforms if their expressed opinion does not match the owner's ideals. That's why there is an increasing opinion that Web2 era is coming to an end and that we need a new system solving the current issues.

In recent years, there has been a growing movement to relieve the current control over data evenly across the Internet, with the help of blockchain technologies, thus transitioning to a new era of the Internet, called Web3. For the successful implementation of the movement, technologies and protocols must be developed that allow users to use the Internet quickly, easily, safely, and without the need to trust a central authority. Various technologies are already available for the development of Web3 solutions, but the full-stack standard, as established for Web2, has not yet been established, which can have a negative impact on new developers who are not yet familiar with the good practices of developing Web3 solutions. Another problem is that

technologies are still in development stage and frequently get updates, which means the developers must constantly follow the news and keep upgrading their knowledge. There is a need for mentioned full-stack standard to provide the new developers a stepping point into the blockchain technologies and Web3 development.

Based on the identified problems, we formulated the following research questions:

- RQ1: What is Web3 and how what makes it different from Web2 and Web1?
- RQ2: What are the technologies, platforms, frameworks, and tools for developing Web3 solutions?
- RQ3: How connected and dependent is Web3 to blockchain and smart contracts?
- RQ4: What is the full-stack for Web3?
- RQ5: Are non-fungible tokens (NFTs) necessary for Web3 development?

1.3. Objectives

The thesis focuses on identifying the technologies for the development of Web3 solutions and investigating which technologies would together form the full-stack for Web3 development. To achieve that, we defined the following objectives:

- Objective 1: To present Web3 and technologies for the development of Web3 solutions.
- Objective 2: Investigate which technologies represent the full-stack for Web3.
- Objective 3: On a practical example, test and present the Web3 development principles, described in this work.

1.4. Hypotheses

Based on the theoretical model, we propose the following hypotheses:

- H1: There is currently no unified full-stack technology for Web3.
- H2: Web3 depends on blockchain and smart contracts.

1.5. Assumptions and limitations

Next, we present the assumptions and limitations of the master's thesis.

Assumptions:

- Technologies for Web3 development are already sufficiently developed.

Limitations:

- We will test the Web3 solution on the local test network.
- A practical solution will be developed with simple functionalities.

2. BLOCKCHAIN

Blockchain popularization began after the year 2008, when an unknown person or a group of people using the name Satoshi Nakamoto invented the cryptocurrency Bitcoin. In 2009 its implementation was released in public as an open-source software. Bitcoin is a decentralized cryptocurrency that works on a peer-to-peer network, where a consensus is in place among users. [2] Consensus merely stands for the rules by which a blockchain network operates and confirms the validity of information written in blocks. Network consists of nodes, which's primary job is:

- to determine whether a block of transactions is legitimate and accept or reject it,
- to save and store transaction blocks,
- to broadcast transaction history to other nodes that may need to synchronize with the blockchain.

The transactions on the network are verified by nodes through cryptography and recorded in a public distributed ledger - called a blockchain. This makes possible for transactions to work between two participants without any intermediary or central authority. [3]

The blockchain is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger. Each block points to the immediately previous block via a reference that is essentially a hash value of the previous block called parent block. The first block of a blockchain is called genesis block and has no parent block. Figure 1 illustrates the example of a blockchain. [2]

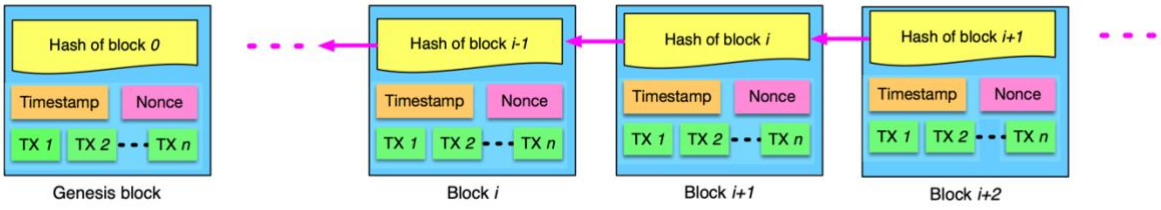


Figure 1. Structure of blocks in a blockchain.

A block in a blockchain consists of the block header and block body. The block header contains:

- Block version: indicates which set of block validation rules to follow.
- Parent block hash: a 256-bit hash value that points to the previous block.
- Merkle tree root hash: the hash value of all the transactions in the block.
- Timestamp: current timestamp as seconds since the January 1st, 1970.
- nBits: a compressed representation of the target value, below which the block's hash must be, to be valid.
- Nonce: a random whole number, which is a 4-byte field and usually starts with 0 and increases for every hash calculation. Nonce is a number that can be used only once.

The block body is composed of a transaction counter and transactions. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. To validate the authentication of transaction in an untrustworthy environment, blockchain uses a digital signature based on an asymmetric cryptography mechanism. [2]

In digital signature, each user has a public and private key. Digital signature has two phases, signing phase and verification phase, as can be seen on Figure 2. In signing phase user generates a hash value derived from the transaction, encrypts it with private key and sends the encrypted hash with original data to another user. That user verifies the received transaction by comparison between decrypted hash (using sending user's public key) and the hash value derived from the received data using the same hash function as used by sending user. [2]

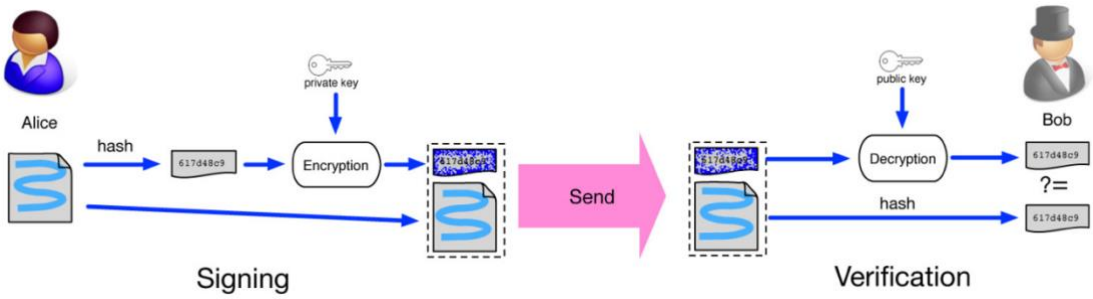


Figure 2. Visual example of digital signature.

To understand blockchain, it is necessary to understand its key characteristics: [4]

- Decentralization: Transaction in blockchain network is conducted between any two participants without the authentication by the central authority. This can significantly reduce the server costs and mitigate the performance bottlenecks that could appear at the central server.
- Persistency: It is nearly impossible to tamper a transaction, since each transaction across the network needs to be confirmed and stored in blocks distributed in the whole network. Apart from that, each block must be validated by other nodes, so any forgery could be easily detected.
- Anonymity: In typical Web2 application a user is asked to create an account and provide email address and some personal information. In blockchain users interact with the blockchain network with a generated address and each user can even generate multiple addresses to avoid identity exposure. This preserves a certain amount of privacy, even though the perfect privacy preservation cannot be guaranteed.
- Auditability: Each transaction on the blockchain is validated and recorded with a timestamp, therefore users can verify and trace the previous records by accessing any node in the network.

2.1. Decentralized applications

Decentralized applications or dapps are applications that exist and run on a blockchain or peer-to-peer network of computers instead of a single computer. In the context of

cryptocurrencies, dapps operate on the blockchain network in a public, open source, decentralized environment and are free from the control and interference of any single authority. [5] For example, a developer could create a Twitter clone dapp and put it on a blockchain where any user can post messages. Once messages are posted, no one, including the creators of the app, can delete them. An ideal dapp would be completely hosted in P2P network and would need no maintenance and governance from original developers. A blockchain application that is operable without any human intervention forms a Decentralized Autonomous Organization or DAO. A DAO is an organization that is governed by rules encoded as smart contracts that run on the blockchain. It is simply a dapp with AI controlled decisions and humans on the edges. DAOs also have their own internal capital. Due to its autonomous and automatic nature, the costs and profits of the DAO are shared by all participants by simply recording all activities in blocks. Decentralized applications are characterized by following properties:

- Open source: It is important to provide open access to the code, so that audits from third parties become possible and consequently earn the trust of the users.
- Internal cryptocurrency support: Internal currency runs the ecosystem of a dapp. It is also the source of profit for the developers of dapp.
- Decentralized consensus: It is the foundation for transparency.
- No central point of failure: All components are hosted and executed in the blockchain on multiple nodes. [6]

2.2. Ethereum

Ethereum, launched in 2015, is the first platform that supported smart contracts (we describe smart contracts later). It was built on Bitcoin's innovation, but with some big differences. Where Bitcoin is only a payment network, Ethereum is programmable, so you can build and deploy decentralized applications on its network. It incorporates Turing complete language, allowing it to support all types of computations, including loops. [7] It provides an abstract layer that allows anyone to create their own rules for ownership, transaction formats, and state transition functions. This is done by incorporating smart contracts, a set of cryptographic rules that are only executed when

certain conditions are met. [8] Every action on the Ethereum network requires a certain amount of computational power. This fee is paid in the form of ether (ETH), which is Ethereum's native cryptocurrency. Originally, Ethereum blockchain was using proof-of-work consensus mechanism, however on 15 September 2022 in an upgrade process "The Merge", it transitioned to proof-of-stake consensus and reduced the energy consumption by approximate 99.95%. In the process the original execution layer of Ethereum was joined with its new proof-of-stake consensus layer, the Beacon Chain. This eliminated the need for energy-intensive mining and instead enabled the network to be secured using staked ETH. [9]

2.3. EVM

Ethereum Virtual Machine is a runtime environment for transaction execution in Ethereum (deployment and execution of smart contracts). It is used to predict the general state of Ethereum for each block on the blockchain as it is added to the chain. EVM uses its own assembly-like, stack based and Turing complete language and consists of approximately 150 unique opcodes, which are machine-level instructions that computer can execute. [10] As opcodes are not developer friendly, Solidity programming language is used to write code and is then compiled in EVM bytecode before deployment on the blockchain. EVM is not completely decentralized since overwhelming amount of Ethereum nodes are hosted on virtual machines like Amazon Web Services. [11]

2.4. Smart contract

The term "Smart contract" was coined by Nick Szabo in mid 1990s, by suggesting translating the clauses of a contract into a code and embed them into software or hardware by making them self-executable, to minimize contracting cost between the involved parties and to avoid accidental exception or malicious actions during contract performance. [12] Smart contract in different disciplines has a different meaning, in our case it stands for a low-level code script running on blockchain. Smart contracts are stored on blockchain and can be automatically executed when certain pre-conditions

are met. They are implemented in programming language Solidity. Main characteristics of smart contracts are autonomy, self-sufficiency, and decentralization. Autonomy means, that after they are deployed, they require no additional monitoring. [13] They are self-sufficient because of the ability to raise funds by providing services and spending them when needed. They are decentralized as they are distributed and self-executed across network nodes. Smart contract system is what makes it possible to build DAOs. [14]

2.5. Crypto wallet

Crypto wallets are software application used to view cryptocurrency balances and make transactions on the blockchain. [15] They store users' public and private keys while providing a straightforward interface to manage crypto balances. Public key is like a bank account number and can be shared publicly, while private key is like a bank account password and should be kept secret. Cryptocurrency is not physically held on the wallet, instead the wallets read the public ledger and show the users the balances in their addresses and hold the private keys that enable making transactions. The wallets store one or more cryptocurrency-unique public addresses. A public address is a hexadecimal string with a combination of numbers and letters, lower and upper case. It must be shared publicly to receive cryptocurrency. A private key is also a hexadecimal string. Since it is difficult to remember, it is stored in wallet software. Instead to access wallet, user must know a pin associated with each private key. For additional security users can also use multi-signature wallets that require two or more private key signatures to authorize transactions. [16] There are different criteria to differentiate crypto wallets, we will focus on the one that is based on device used to store keys:

- Desktop wallet: A software that can be downloaded and installed on a computer. They offer one of the maximum tiers of security.
- Online wallet: A software that runs on the cloud and can be used from anywhere. They are easy to access, but private and public keys are stored by a third-party.
- Mobile wallet: A software that runs as an application on a mobile phone.

- Hardware wallet: A physical storage of keys on a device that works similar to USB devices. They make transactions online, but public and private keys are stored offline.

Another type of wallets worth mentioning is NFT wallet, which is specially designed to store non-fungible tokens. All mentioned wallets offer some or all features like authorization of user, key generation, key management, anonymity, multicurrency support, coin or token conversion rates, crypto-exchange, QR code scan to transact cryptocurrency, push notifications and backup and restoration facility of keys. [17]

2.6. NFT

Non-fungible token (NFT) is defined as cryptographically unique, indivisible, irreplaceable and verifiable token that represents a given asset, be it digital, or physical, on a blockchain. An owner of an NFT can easily prove the existence and ownership of an asset, furthermore all the previous owners can be tracked as well. NFTs are created and managed by smart contracts. [18] There is a lot of confusion among people, that NFT is an asset, e.g., that a digital artwork is an NFT, which is incorrect. NFT is merely a data structure on blockchain that holds the information of ownership of that artwork, with some additional information including a location address of where the asset is stored. Since digital artwork is usually a large file, too large to store it on a blockchain, an alternative way is used for storage. Most common way to store the assets is by using an Interplanetary File System (IPFS), which is a distributed peer-to-peer file-sharing network. A huge burst of popularity of tokens happened in 2021, when certain individuals made a huge amount of profit by trading NFTs. NFTs are mostly associated with digital art, however they are useful for any case where an ownership of some unique asset is required. There is a lot of potential in video game industry, where every item obtained by a player in the game could be represented as an NFT. NFT is created by minting an asset and then uploading it on a blockchain. Minting is a process of creating and producing an NFT and often costs a fee called gas, of which price depends on a platform and a blockchain, where it is deployed. [18] Before Ethereum's upgrade to proof-of-stake, an average gas cost for minting an artwork was around 100$. On other

hand, NFTs have a feature to add in a royalty fee that pays creator a percentage of the transaction each time that NFT is sold, and it is an additional incentive for artists to contribute to the NFT community.

2.7.Gas or transaction cost

Nodes running the EVM cannot foresee the amount of resources required for validating a transaction, which enables denial-of-service attacks. To counter that, a pricing mechanism is incorporated. Every computational step in EVM is priced in units of gas. For each transaction the sender must specify the maximum amount of gas that is expected to be consumed by the computation and the price that the user wishes to pay per unit of gas. The price of a gas unit in ether is defined by the market. The transaction fee equals to the gas limit multiplied by the gas price. [2]

3. DEVELOPMENT OF WEB

Our first research question was: *What is Web3 and how what makes it different from Web2 and Web1?* This chapter is dedicated answering that. The Internet is probably one of the most important technological revolutions in the history of mankind, where the Web, as one of the representations of the Internet, is still in development. People often mistakenly use the terms internet and Web as synonyms, even though they have different meanings. The Internet is a global network of interconnected servers, computers, and other devices where each device can connect to another, provided that both devices are connected to the Internet with a valid IP address. The Internet, on the other hand, is only one of the methods of spreading information over the Internet, others include email, File Transfer Protocol (FTP) and instant messaging service). The Internet consists of a huge number of digital documents that we access using Web browsers. In its relatively short history, the Internet has already experienced some major milestones, the biggest being Web1 and Web2. At the moment there is a lot of talk about Web3, which is supposed to be the next revolution of the Web, and which is also the central topic of this work. To better understand what Web3 represents, it is first necessary to understand how the Web has developed throughout history. [19]

3.1.Static web or Web 1.0

Web1 or the World Wide Web or simply put the Web, represents the first evolution of the Web, known as the read-only Web, where there were fewer creators of websites and a greater number of consumers accessing these pages. Tim Berners-Lee innovated the Web in 1989 while working for CERN. He developed the first Web server, the first Web browser and the document formatting protocol Hypertext Markup Language known as HTML. The beginning of the Web is the year 1991, when Berners-Lee released HTML to the public. Web1 was mostly used to represent static content with no or minimal interaction capabilities. Websites were used for displaying information and user could easily access it by visiting the publisher's website. [20] During Web1, Web server performance and bandwidth had to be considered, since multiple pages and enormous content would slow down the entire site. Nevertheless, Web1 included some

capabilities of Web2, but they were implemented differently. Namely, comment section in Web1 was present in the form of a guestbook page, where visitors were able to put down comments.

3.2.Semantic Web or Web 2.0

The transition from Web 1.0 to 2.0 took place over time as servers were upgraded, average connection speeds increased, and developers learned new skills and techniques. The term Web2 was first introduced by Darcy DiNucci in her article "Fragmented Future" in the year 1999. In her article she described how in future the basic information structure and hyper-linking mechanism would be used across a variety of devices and platforms. However, her representation of Web2 does not directly relate to term's current use. The term Web2 became popular in year 2004, when O'Reilly Media and MediaLive hosted the first Web2 conference. At the conference, John Batelle and Tim O'Reilly defined "Web as Platform", where software applications are built upon the Web instead of upon the computer. They proposed that user activity on the website could be harnessed to create value. [19] If Web1 was called read-only Web, we could define Web2 as a read-write Web or participative social Web, where users are invited to interact with dynamic content and contribute to it.

Some important features of Web2 are:

- users as a first-class entity in the system, with correlating profile pages,
- the ability to form connections between users, via links to other users tagged as friends or membership in groups of various kinds or subscription to RSS feeds of updates of other users,
- the ability to post content in multiple forms, as photos, videos, blogs, comments, and ratings,
- other more technical features like public API to allow third-party enhancement or communication with other users using internal email or instant messaging systems. [19]

One of the biggest contributors to Web2 is the mobile internet access and the rise of social networks as we can see from Figure 3 in June of 2022 60% of total Web traffic comes from mobile internet traffic. [21]

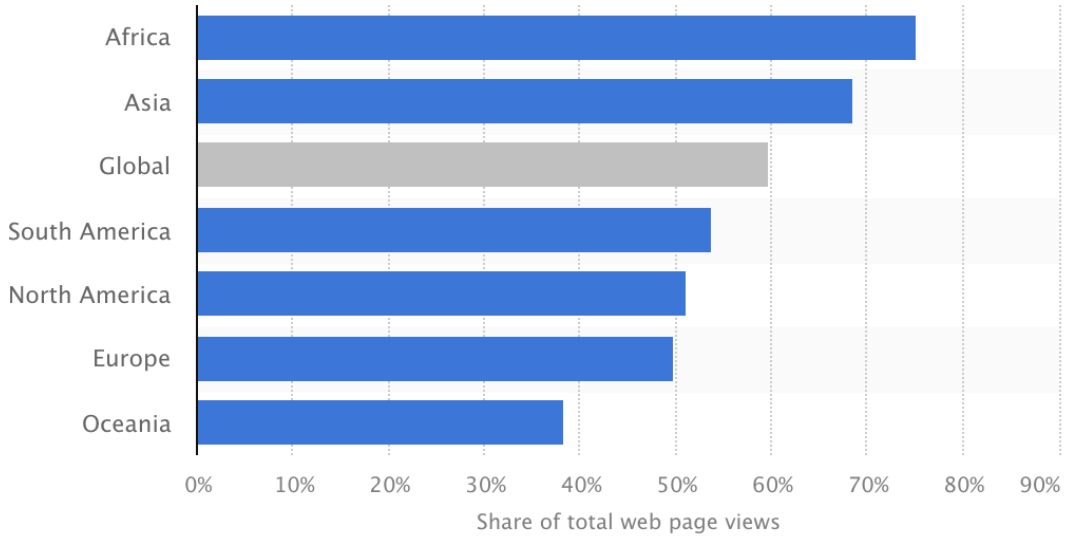


Figure 3. Mobile internet traffic as percentage of total Web traffic in June 2020, by region.

The Web as we know it today, is indeed social and collaborative, but it comes at the cost. The cost that most of the users ignore or are even unaware of. The user data is centralized and exploited by big corporations, it is then used without user's consent for marketing purposes and users have no control over it. Collecting user data, processing it, and using it to make profit is what Web2 stands for.

3.3. Decentralized Web or Web 3.0

The core idea of Web3 is that big corporations (CA) should not have control over user data, but each user should be able to control his own data. As such, Web3 focuses on decentralized data structure, AI driven services and edge computing infrastructure. The term Web3 was coined by Polkadot founder and Ethereum co-founder Gavin Wood in 2014 referring to decentralized online ecosystem based on blockchain. [22, p. 3] The idea of Web3 gained a lot of popularity in 2021, largely due to interest from cryptocurrency enthusiasts and investments from high-profile technologists and companies. Typical characteristics of Web3 are:

- It's a semantic Web, as it lets users create, share, and connect content via search and analysis.
- It is decentralized. Instead of being controlled and owned by centralized entities, ownership gets distributed amongst its builders and users.
- It involves Artificial Intelligence and Machine Learning. If these are combined with Natural Language Processing (NLP), the result is a computer that becomes smarter and more responsive to user needs.
- It offers the connectivity of multiple devices and applications through the Internet of Things (IoT). Semantic metadata enables this process and enables efficient use of all available information. In addition, people can connect to the Internet anytime, anywhere, without the need for a computer or smart device.
- It offers users the freedom to interact publicly or privately without exposing them to risk through an intermediary, thus providing people with "trustless" data.
- Enables participation without requiring permission from an administrative authority. It’s permissionless. [23]

Even though Web3 is not completely developed yet, there are already many elements of Web3 available and used on daily basis, such as NFTs, Blockchain, Distributed ledgers, and the AR cloud. We will discuss these later.

3.4. Comparison between Web1, Web2 and Web3

For better comprehension of the differences between Web1, Web2 and Web3 can be seen on Table 1. The first difference between versions of the Web was their purpose. As Web1 was focused on information sharing the content type consisted of static web pages and was meant to be read-only with offering no interaction. Web2 was based on dynamic web content where the objective was user interaction with the content, that's why we name this version a participative social web. The websites often ask users to interact with the content in terms of likes and comments. While Web2 promised interactivity and engagement, it was always according to the rules and monetization strategies of the existing platforms. Web3 offers far more immersive and engaging

experience, where users have control of their data and content. While in Web1 and Web2 the data is owned by centralized organizations, in Web3 data is owned by users. In Web1 centralized infrastructures i.e., Web servers were used to host websites, which was migrated to Cloud computing infrastructures later with Web2. [24, p. 3] Web3 aims to be decentralized, where everyone can be a participant of the network by hosting their own node. In terms of accessibility, Web1 and Web2 were convenient and easily accessible using Web browsers, with Web Browsers sometimes not supporting the newest features. In Web1 there was no need to registration of its users, while in Web2 almost every website tries to register new users or at least using Single Sign On (SSO) that allows users to use one set of identity-verifying user credentials for authentication on multiple websites. With accessibility in Web3 there is a lack of integration with modern Web Browsers, since user authentication is implemented by connecting user's wallet there is a need of using browser extensions or browser that natively support wallets. In Web1 the advertising was made using simple banners on the website, that's why page views were crucial to generate income. Web2 improved advertising by collecting user data and processing it with recommender systems to create user-tailored ads. This created a system of profit per click, where website owner would earn money based on clicks on ads displayed on their websites. In Web3 we could see a new type of advertising where individuals would opt-in to share their data with companies and for that be compensated. The technologies used for implementing Web1 websites were HTML with basic CSS without considering responsiveness. Web2 was aiming to create websites responsive, so the website would adjust accordingly to devices, i.e., different dimensions and adapted components when displaying website on computer browser or mobile device. HTML5 and CSS3 were used in combination with AJAX and Javascript to enable on-page load, which loads dynamic content without the need to refresh the page. Web3 is based on blockchain, artificial intelligence technologies, and decentralized protocols, and is yet to show what the user interaction will look like. One of the predictions is the users will connect to Web3 using AR or VR technologies. [25]

Table 1. Comparison between Web1, Web2 and Web3.

	Web1	**Web2**	**Web3**
Purpose	Read-only	Read-write	Read, write, interact
Content type	Static web content	Dynamic web content	Semantic content
Content	Home pages	Blogs, wikis	Live-streams, waves
Data owner	Centralized organization	Centralized organization	User
Objective	Information sharing	Interaction	Immersion
Authentication	None	Creating new account or SSO	Connect with crypto wallet
Infrastructure	Centralized infrastructure	Cloud computing infrastructure which is mainly centralized	Decentralized infrastructure
Accessibility	Convenient and accessible	Convenient and accessible	Lack of integration with modern browsers
Advertising	Banner advertising	Interactive and behavioral advertising	Opt-in value exchange advertising
Technologies	HTML, HTTP, URL	AJAX, Javascript, CSS3, HTML5	Blockchain, artificial intelligence and decentralized protocols
Income / Profit	Page Views	Profit per click	Creating value

3.5.Technologies for Web3 development

A distributed ledger works on pre-defined rules which are agreed upon by all the participating nodes in the network. These rules are referred to as a protocol. The most known blockchain protocol for creation of decentralized application is Ethereum, which was the first protocol incorporating smart contracts. Ethereum is a public blockchain and permissionless network which means that it can be accessible to anyone for both read and write operations. The other public networks are Polygon and Solana. On the other hand, Hyperledger Fabric is a private and permissioned network. This means only privileged entities and nodes can participate. To gain access a permission from a trusted Membership Service Provider must be granted. Additionally, in Hyperledger Fabric there is no need for gas, since every participant knows all the other participants in the network and malicious users can easily get detected and removed from the network. [26]

During development developers use development environments to test the contracts in local or public test networks. Popular tools for establishing Ethereum development environment are Hardhat, Truffle, Geth and Remix IDE, where the latter is running in the browser and is also a code editor. Hyperledger Fabric development environment is established using Docker.

User interface or frontend serves as a bridge between users and blockchain. The technologies for developing Web3 solution's frontend are the same as the ones used for Web2 solutions. For developing browser applications most popular are React.js, Vue.js and Django. Example for mobile applications is Android platform.

Frontend communication with blockchain network is implemented using different libraries. The most common one is Web3.js, which is the most popular Javascript based library for interaction with Ethereum network. Others are ethers.js, web3.py, Infura API, Hyperledger Fabric Node SDK etc.

Sometimes we want to explore transactions on the blockchain. For that purpose, the block explorers like Etherscan or Polygonscan can be utilized. These tools allow anyone to browse through blocks, view wallet addresses, network hash rate, transaction data and other key information on the blockchain.

As we mentioned, Web3 aims to be decentralized, and for that decentralized storage must be used. Interplanetary File System (IPFS) is a distributed and decentralized storage network for storing and accessing files, websites, data, and applications, using P2P network to connect a serious of nodes across the world. Content stored on IPFS can be accessed from any IPFS gateway. Swarm is another similar distributed and decentralized network.

For user to interact with blockchain he must authenticate with some sort of identity. In Web3 user authentication is accomplished by connecting the blockchain wallet. The easiest and most common way is by using MetaMask blockchain wallet application which can be installed as a browser extension and a mobile app. It takes care of wallet keys, secure login, token wallet, and token exchange. In addition, a specialized browser like Brave can be used. Brave is a browser that natively supports blockchain wallets.

4. WEB3 SOLUTION EXAMPLES

4.1. DeFi

DeFi stands for decentralized finance, and it was a first popular example of web3 solution. It represents Web3's version of a more transparent financial system with main goal to not be influenced by regulators or the human factor. Most of DeFi solutions allow users to manage their funds in a non-custodial manner using a crypto wallet. Typical DeFi solution is a decentralized exchange (DEX), which is a peer-to-peer marketplace that lets cryptocurrency buyers and sellers interact. One of the most reputed ones is MakerDAO, launched in 2017. It is a P2P lending and borrowing platform for cryptocurrency with all transactions being controlled by smart contracts. Another example is Uniswap, an open-source protocol for providing liquidity and trading ERC20 tokens on Ethereum. In contrast to other popular exchanges like Coinbase or Binance, Uniswap is entirely decentralized. [27]

4.2. Web3 gaming

A Web3 game is a decentralized version of traditional video game where player have complete ownership over their assets and experiences earned in the decentralized Web3 game ecosystem. This enables players an innovative benefit of play-to-earn, since in-game assets can be traded using cryptocurrency. Web3 games also don't have a single point of failure due to their distributed nature, which ensures high availability. Apart from that, they use voting consensus for changes in the gaming process and make players real contributors to the game ecosystem. Another hot topic in Web3 gaming is an idea of a Metaverse, the virtual reality world humans would connect to using virtual reality (VR) or augmented reality (AR) devices and would mimic the real world in every way possible. An example of Web3 game is Axie Infinity, a pokemon-like game, where player collects creatures called Axies and owns them as NFTs. Axies can be bred, traded, or go to battles. It is a free browser-based game, but to play you need to purchase a team of three Axies. The lowest tier Axies used to cost around $350 each, but after a $615 million hack of their network on March 23, 2022, their price fell to a few dollars.

However, this "accident" didn't cause an end of Web3 games era, there are many other Web3 games available and even more in development. [27]

4.3. Web3 social networks

Big corporations like Facebook, Twitter and Instagram currently dominate in social network market and make profits by collecting users' data, selling it, and running targeted advertising. In the past there were numerous complaints and lawsuits against such companies for invading privacy of their users and having too much control over users' personal information. Web3 contributors aim to create different social network, where the platforms are operated by communities and users have control over their personal information, content, and identities. Lens Protocol is an example of advanced Web3 social media solutions. It allows multiple social media and messaging services be built on separate clients but use same open-source smart contract protocol. In Lens, social identities are stored as NFTs in user wallets and can be ported across all dapps that integrate its protocol. [27]

4.4. Web3 marketplaces

Web3 marketplace can be described as a system where a collection of smart contracts coordinates service providers and clients as well as facilitates their interaction. Providers could offer many different levels of customized services or products. Both clients and service providers would earn the same governance token based on their contribution to the system. Braintrust is a good example of such marketplace. It is branded as user-owned talent network and it connects users, who want to work as freelancers with enterprises, which want to quickly find the right talent for their project. It is still early to determine if such business model will be successful in the future and be able to compete with Web2 business that offer the same services, it depends on its community. [27]

It is important to point out a type of Web3 marketplace, that grew so much in the past years it deserves separate mentioning. It is called the NFT marketplace, and it is a

gateway to trade NFTs. There are various such marketplaces, some of biggest include: [28]

- OpenSea: The largest NFT marketplace at the time of this writing. It is very user-friendly and can let users get set up with account within minutes. It offers trading of art, music, photography, trading cards and virtual worlds.
- Rarible: Users can trade art, collectibles, and video game assets. They created its own native token RARI. Apart from that, they are partnered with Adobe to easier verify and protect metadata of digital content.
- NBA Top Shot: It is a marketplace of highlights in basketball history managed by the NBA. It is an example of major companies participating in NFT trend.
- Binance: This is originally a cryptocurrency exchange, but it added NFT marketplace to its features and it offers trading of art, gaming assets and collectibles.
- Nifty Gateway: The platform is known to host expensive and exclusive NFT sales, including digital artist Pak's "The Merge", which sold for $91.8 million. They focus on selling only artwork, especially from celebrities and top artists.

5. SYSTEMATIC LITERATURE REVIEW

We identified and analyzed the existing principles of the development of Web3 technologies through a systematic literature review (SLR). It is a research method by which we evaluate and interpret all available research that relates to a specific research question, topic area, or interest. Based on the guidelines, we conducted a systematic review of the literature in three phases: SLR strategy, results, and discussion.

5.1.Strategy

The systematic literature review was carried out with the aim of obtaining a broader overview of the research area. We wanted to answer RQ2, RQ3, RQ4 and RQ5:

- RQ2: What are the technologies, platforms, frameworks, and tools for developing Web3 solutions?
- RQ3: How connected and dependent is Web3 to blockchain and smart contracts?
- RQ4: What is full stack for Web3?
- RQ5: Are non-fungible tokens (NFTs) necessary for Web3 development?

In the systematic literature review, we focused on the following digital databases, IEEE Explore, SpringerLink and ScienceDirect.

5.1.1. Preliminary search and identification of the need to conduct a systematic literature review

Before carrying out a systematic review of the literature, we carried out a so-called preliminary search, with which we wanted to discover the research area broadness and determine whether there is enough literature available and whether the area is interesting for research. The preliminary search used the same databases as planned for the full systematic literature review. The criteria used is presented in Table 2.

Table 2. Inclusion criteria of the preliminary literature review.

Inclusion criteria	
Criteria	Criteria description
C1	The research includes keyword Web3 or dapp
C2	The research includes keyword platform, tool, framework, technology, or development
Exclusion criteria	
C3	The research was conducted before 2018

The preliminary search was carried out on September 16, 2022. The search was performed by metadata and full text. The search string was created according to the previously presented criteria "(web3 OR dapp) AND (platform OR tool OR framework OR technology OR development)" and only results from 2018 onwards were considered. The number of research found is presented in Table 3.

Table 3. The total number of research found by preliminary search.

Digital library	Search results
IEEEXplore	211
SpringerLink	1646
ScienceDirect	760
Total	15417

In the previous research, we did not find any research that carried out a systematic literature review in the field of Web3 solution development principles. The latter, together with enough literature related to the target concepts, indicates the need to conduct a systematic literature review.

5.1.2. Search string identification

To obtain the most relevant research possible, we created 3 groups of keywords for a detailed review of the literature, which can be viewed in Table 4. Group G1 focuses on narrowing the field to decentralized applications. During initial search, we observed both "decentralized" and "decentralised" were used, also just "dapp" was commonly used. Group G2 is trying to verify the study is focusing blockchain. Group G3 is narrowing the field to studies that performed any kind of testing on their solution. The idea is to find studies, that tested their solution on a blockchain test network. Group G4 is verifying if a dapp was implemented or developed during the research. Group G5 is further verifying if a user interface was implemented, since one of the focuses of this study is to find a stack of technologies for Web3 development and not solely blockchain development.

Table 4. Definition of keywords and keyword groups for SLR.

Group	Keyword	Keyword derivatives	Keyword expression	Purpose
G1	dapp	dapp, decentralized application, decentralised application	dapp	Narrowing the field to decentralized applications.
			decentrali*ed application*	
G2	blockchain	blockchain	blockchain	Study has to mention blockchain.
G3	test	test, tested, testing, testnet	test*	Narrowing the field to studies that performed any sort of testing (includes testnet).

G4	implementation	implementation, implemented, development, developed	implement*, develop*	Focusing on implemented or developed solutions.
G5	frontend	frontend, front-end, front end	frontend front?end	We are searching for implementations that include frontend.

Based on the groups of keywords, we made the decision that relevant research must contain at least one derivative from each group of keywords. Based on the selected groups, we defined the following search string:

((dapp OR "decentrali*ed application*")
AND
blockchain
AND
test*
AND
(implement* OR develop*)
AND
(frontend OR "front?end" OR GUI OR interface))

5.1.3. Digital libraries specifics

SLR was conducted on multiple digital libraries, which each has its own database search engine, so an additional search string modifications were necessary. The modifies search strings for each digital library is shown in Table 5.

Table 5. Modified search strings for each digital library.

Digital library	Search string after modifications	Comment
IEEE Explore	((dapp OR "decentrali*ed application*") AND blockchain AND Search_All_Text:test* AND (implement* OR develop*) AND ("Full Text & Metadata":frontend OR "Full Text & Metadata":"front?end" OR "Full Text & Metadata":GUI OR "Full Text & Metadata":interface))	We narrowed the scope by filtering years of publication between 2018 and 2022. Some of keyword were additionally set up to search full text.
SpringerLink	((web3* OR dapp OR "decentrali?ed application?") AND (platform OR tool OR framework OR technology) AND (implementation OR implemented OR development OR developed OR solution))	We further narrowed the scope by filtering years of publication between 2018 and 2022 and choosing English as a preferred language.
ScienceDirect	((dapp OR "decentrali?ed application") AND blockchain AND (implementation OR implemented OR development) AND (frontend OR "front?end" OR "user interface"))	We had to modify the search string to use maximum of 8 boolean connectors. We also filtered years between 2018 and 2022.

5.1.4. Inclusion and exclusion criteria

To select suitable research from the multitude of research that we obtained from selected digital databases based on the search string, we defined inclusion and exclusion search criteria. These are presented in Table 6. The criteria allow a more precise limitation of the contents, which are also consistent with the defined research area. Studies that did not meet the inclusion criteria (C1-C3) or included the exclusion criteria (C4-C7) were eliminated and not discussed further.

Table 6. Definition of inclusion and exclusion search criteria.

Inclusion criteria	
Inclusion criteria	Criteria description
C1	The research is related to the development or technologies for the development of Web3 solutions.
C2	During the research, a Web3 solution was developed and/or analyzed.
C3	The technologies used to implement the solution are clearly and fully specified.
Exclusion criteria	
C4	Access to the research is not available.
C5	The research was conducted before 2018.
C6	The research does not fit any of the types: scientific or professional article, PhD, thesis, conference paper.
C7	The survey is not available in English.

5.2.Search results

The review of the collected research was carried out in several stages, which are shown in Table 7, for greater clarity.

- In the first phase of the research, based on the search string, we determined a set of researches that correspond to the selected keywords.
- In the second phase of the research, we excluded from the set research that met the exclusion criteria C4-C7.
- In the third phase of the research, we focused on meeting the conditions of criteria C1-C2 and reviewed the title, abstract and keywords of the research. If, based on these, we determined that the research was not suitable, we removed it from the set.
- In the fourth phase, we carried out a detailed review of the entire content of the research. If, after reviewing the entire content of the research, we found that the research is not suitable or does not meet the C3 criteria, we removed it from

the set, otherwise we classified it in the group of primary research. After obtaining a mass of primary research, we removed from it duplicates that appeared due to the use of different databases. In doing so, we attributed the research to the database in which we first found it.

Table 7. Phases of the research with description.

Phase	Description
P1	Selection based on search string.
P2	Narrowing the results based on exclusion criteria K3-K6.
P3	Selection based on appropriateness of title, abstract and keywords.
P4	Selection based on the entire content of the research. Elimination of duplicates.

Table 8 shows the numerical results of the found primary research by individual phases. After the last stage of the review, 55 studies remained on which analysis was conducted.

Table 8. Search results after each phase.

Digital library	Search date	Number of studies after P1	Number of studies after P2	Number of studies after P3	Number of studies after P4
IEEE Explore	20.9.2022	133	128	54	23
Springerlink	24.9.2022	1152	254	66	19
Science Direct	22.9.2022	148	111	46	13
Total number					55

In Table 9 are represented articles ordered by year of publishing. We can observe there were more articles meeting the conditions of criteria in the years 2020, 2021 and 2022, than previous years. There was only 1 suitable article found for the year 2018 and only 4 articles for the year 2019. Most suitable articles were found in years 2021 and 2022.

This could be due to technology still being new and studies were focusing more on methodologies and mechanisms, than implementing complete solutions.

Table 9. Summary of studies by years.

Year	Articles	Articles (%)
2018	1	1,82%
2019	4	7,27%
2020	9	14,55%
2021	21	38,18%
2022	20	34,55%

On Table 10, we can observe the leading blockchain technology used in the studies was Ethereum blockchain. There were 4 studies found, that implemented their solution on Hyperledger Fabric and there was only 1 study that implemented their solution on EOS blockchain. Surprisingly, there were no studies found that would incorporate any other popular blockchain, like Solana, Polygon or BNB chain.

Table 10. Summary of studies by technology.

Blockchain technologies	
Ethereum	50
Hyperledger Fabric	4
EOS blockchain	1

Following the blockchain technologies used in studies, it was expected Solidity would be the most popular choice for implementing smart contracts. Solidity was used in all studies, that were built on Ethereum blockchain. The studies using Hyperledger Fabric were using either Golang or Javascript to implement their contracts. One of the studies using Hyperledger Fabric did not define, which language was used. The study using EOS blockchain implemented their smart contracts in C++. The overview of languages can be seen in Table 11.

Table 11. Summary of studies by smart contract languages.

Smart contract languages	
Solidity	50
Golang	2
Javascript	1
C++	1
Not given	1

For implementation of the user interface, various technologies were used across the studies. The results reflect on the popularity of frontend technologies of Web2 development. Most of the user interfaces were implemented by Javascript, or Javascript based framework or library. As seen on Table 12, 16 of the studies chose Javascript combined with HTML and CSS for implementing the frontend. React.js, a Javascript based library, was also a popular choice with 11 studies using it. Other, non-Javascript frontend technologies were Django, Python and ASP.net. There were 2 studies, that implemented a mobile dapp using Android and one study using React Native. Lastly, one of the studies was using Ganache GUI as a user interface. In 6 of the studies, it was not mentioned, which frontend technologies were used.

Table 12. Summary of studies by frontend technologies.

Frontend technologies	
Javascript, HTML, CSS	16
React.js	11
Vue.js	6
Angular	3
Django	2
Python	2
Android	2
Next.js	2

React Native	1
ASP.net	1
Ganache GUI	1
Not given	6

Decentralized storage technologies used in the studies include IPFS and Swarm. The majority of studies did not include any decentralized data storage apart from the one on blockchain. On Table 13, we can observe the most used technology for decentralized storage in the studies was IPFS. We found 18 studies implementing IPFS for various solution types, e.g., sharing credentials, marketplaces, ticketing system, data repositories, or live streaming. What they have in common is storing files that are too large to be stored on blockchain - it would be inefficient to store them, or it would cost too much gas. Another technology used for storage was Swarm and it was used in 3 studies.

Table 13. Summary of studies by storage technologies.

Storage Technologies	
IPFS	18
Swarm	3
None	34

The following **Error! Reference source not found.** focuses on exploring, which technologies were used to communicate between user interface and contracts on blockchain. Here it is important to note, that multiple of these technologies could be used in the same study. The most used was web3.js, being used in 31 studies. From this, we can conclude, it is the most popular and easy to use API for connecting with smart contracts. On the second place was Infura API, which apart from connecting to Ethereum, offers support to connect with IPFS. It was often observed Infura was used in combination with web3.js. Web3.py and web3j function similar to web3.js, only for different platforms, Python and Java/Android respectively. The found technologies for

connecting with Hyperledger Fabric were Hyperledger Fabric Node SDK, Fablo REST API and Hyperledger Composer API. It is interesting that each study using Hyperledger Fabric chose a different API for implementation of their solution. This could indicate that there is no standard choice for Hyperledger Fabric API, or that developers are more open to use different APIs for implementation. However, there is not enough studies researched to confirm any of these. Finally, some additional APIs were found in the studies. Firstly, OpenZeppelin was used in the study that implemented NFTs. Secondly, Whisper API was used to communicate with the Whisper protocol, which was used to implement a messaging dapp.

Table 14. Summary of API technologies identified.

API technologies	
web3.js	31
Infura	8
web3.py	3
web3j	2
Hyperledger Fabric Node SDK	1
Fablo REST API	1
Hyperledger Composer API	1
OpenZeppelin	1
Whisper	1

5.3.Studies review

For each study that passed the quality assessment, we examined the following further aspects: the motivations for which it was written, which contributions it gave to research, which results it has achieved, and which challenges it posed for the future. Finally, a spreadsheet was filled with the following information for each paper:

- title of the study,
- developed solution,

- blockchain technology,
- smart contract language,
- environment tool,
- testnet,
- frontend technology,
- API,
- storage,
- use of smart contracts,
- use of NFTs.

This information was especially useful to structure the relevant aspects of each study necessary for our systematic literature review. Next, we will continue with analysis of the collected studies. For better comprehension, we have grouped some of the studies based on their characteristics or research focus.

Altamimi et al. in the article [29] propose a framework for deploying mobile applications using Ethereum blockchain. The proposed system demonstrates the promise of decentralized systems in enhancing the time and reducing the cost to deploy a mobile application, compared to current platforms e.g., Apple and Google Play stores. The system was tested on Ropsten and Rinkeby testnets and it was observed the gas costs were highest when adding new app information to the blockchain and the lowest when deleting the app. IPFS decentralized storage was used to store and download applications, which offer better optimization of network utilization and shorten the download times.

In the next group of studies, all authors implemented their version of decentralized voting application. In 2019, the authors in the article [30] developed a sample voting app with ASP.net. Canessane et al. [31] proposed to decentralize voting system to increase privacy and to remove the constraints of time and location by allowing users to vote from their own blockchain nodes. Rosa-Bilbao and Boubeta-Puig [32] in contrast to other voting systems, proposed a system, where each voter has a different weight of vote,

depending on university staff type. Additionally, they offer ability to obtain partial and total election results in real time. In 2022, Alvi et al. [33] proposed a mechanism for security in digital voting systems. The implemented system provided voter anonymity by keeping the voter information as a hash in the blockchain. It also provided fairness by keeping the casted vote encrypted till the ending time of the election. After ending time, the voter could verify their casted vote, ensuring verifiability. All 4 studies implemented the systems on Ethereum blockchain and tested the prototype on with Truffle.

In 2022, Sasikala et al. [34] did a survey on latest technologies on decentralized applications, where they implemented a full and a partial decentralized app and carried out performance comparison. The tests were performed utilizing Locust, a Python program for Web application performance testing, and Mocha, a Javascript test framework for creating test scenarios. Results showed fully decentralized app was delivering frontend 41.46% faster, than the partial one. However, when it came to customer solicitations, partial dapp had a faster response time.

The next group of articles focused on developing a decentralized healthcare system on Ethereum blockchain. Santhanakirshnan et al. [35] proposed a proof-of-concept healthcare system for secure healthcare information transfer. They found the number of benefits to be equal to the number of drawbacks, however, they believed with research advancements the benefits could outweigh the drawbacks. They concluded the assurance of immutability, traceability, transparency, and decentralization would provide a huge boon to the healthcare industry. The authors in article [36] implemented a system for secure storage of healthcare data using Ethereum blockchain with additional analytical capabilities utilizing Machine Learning. They proposed 2 possible improvements to build a more promising system, introducing new consensus algorithms to enhance speed and efficiency, and better architecture of storing medical records, with more up-to-date and accurate data available for training Machine Learning models. Satamraju and Malarkodi [37] in their work propose a decentralized framework for IoT

devices using physically unclonable features (PUFs) and blockchain. They showcased their model on a smart healthcare management system and successfully achieved the overall security goals of the IoT applications. The PUFs used in the design have 48.46% uniqueness (50% ideal) and 2.38% reliability. The system was tested on 2 Raspberry Pi based medical devices.

Abdulaziz et al. [38] proposed a secure and anonymous decentralized messaging application built on Ethereum platform using Whisper protocol. The application could send end-to-end encrypted messages while ensuring anonymity of the sender and receiver. Peer-to-peer communication was achieved with Whisper protocol serves to encrypt a message and, in theory, send it to every Whisper node. The messages had to be encrypted asymmetrically using Elliptic Curve Integrated Encryption Scheme or symmetrically encrypted using Advanced Encryption Standard Galios/Counter Mode. However, there were some unexplored issues found, where messages intended for a specific user could expire while user was offline or during an unexpected network failure.

TogEther, was an idea of application for crowd funding by Nagadeep et al. [39]. They implemented a user-friendly app for funding a start-up campaign in a decentralized way on Ethereum blockchain. They compare blockchain transactions with centralized bank transactions, where in case of failed transaction, blockchain would immediately refund ethers into sender's account, opposite of centralized banks where it would take time to process transactions and it could take few days to return the money. The work proposed a technology stack for implementing dapps consisting of React.js, web3.js and Solidity.

Muth and Tschorsch [40] had a vision of a dapp named SmartDHX, supporting Diffie-Hellman key exchange (DHKE) scheme fully implemented as a smart contract. The cryptographic logic was implemented in Solidity, and client-side logic in web3.js. They measured performance based on blockchain specific metric instead of network metrics. Two-party and multi-party SmartDHX were compared, and it was observed, the costs to

perform DHKE on-chain was high. However, other costs e.g., exchanging a shared key, could be negligible.

Both of next articles were focusing on sharing students' credentials. Mishra et al. in 2020 [41] proposed architecture for sharing student's credentials comprising of five major stakeholders, government body, students, companies, schools, and professors. Each of them was given different set of functionalities offered using different dashboards on dapp. They used IPFS to store credentials and the generated hash value together with metadata was stored on Ethereum blockchain. The test of dapp were performed on Rinkeby testnet and it was observed the average upload of credentials was around 16s. Mishra et al. in 2021 [42] focused on security analysis of their solution. They compared architectures with and without privacy protection. The proposed dapp was then evaluated based on top 10 Open Web Application Security Project (OWASP) vulnerabilities. For each vulnerability there was a countermeasure proposed. The findings indicated that in most cases already the use of blockchain technology ruled out the possibility of these security risks. Additionally, the performance experiments showed that the performance of dapp remains almost the same without or with privacy integration. For future, they intend to deploy architecture on permissioned blockchain and integrating the ability of revoking credentials.

One of the most popular uses of blockchain technologies are marketplaces. We will continue with analysis of 6 studies related to marketplaces. In the article, Ivankovic et al. [43] proposed an auction system framework based on Hyperledger Fabric. They subdivided the smart contracts in 3 subtypes, create operations, read operations, and update operations. The KPIs measured were throughput, meaning the rate at which valid transactions are committed, and transaction latency, meaning the amount of time for the effect of a transaction to be acknowledged by other network nodes. Results shown the average latency increases in accordance with the send rate. Comparing the read operations to that of the create operations, there was a significantly higher throughput and significantly lower latency. Shakila and Sultana [44] in their work,

describe the process of migrating existing centralized marketplace to decentralized marketplace. Centralized marketplace was consisting of a frontend application in Javascript and a backend central system with database. The new proposed architecture was frontend in Javascript, Ethereum for storing transaction data, and IPFS for uploading product detail files. Interaction with IPFS was programmed with Infura API and it was tested on local Kovan testnet. After analyzing gas consumption, the authors concluded it was an acceptable amount. Profit margin was higher, compared to traditional centralized systems, Amazon, and Ebay. They concluded that decentralized marketplaces have huge potential, but further tests should be implemented on public blockchain. Although cyber threat intelligence exchange is a theoretically useful technique for improving security of a society, the potential participants are often reluctant to share their knowledge. Riesco et al. [45] proposed a decentralized cyber threat intelligence marketplace on Ethereum blockchain. They implemented CTI token based on ERC20 token standard, which could help attract investors. Additionally, they implemented safe math libraries written in Solidity to implement math operations with safety checks that revert on error. In conclusion, they write the decentralized marketplace provides new economic incentives to all roles involved and its value is depending on the quality of data. Third article by Menges et al. [46] involved implementation of threat intelligence sharing platform, and a prototype was developed based on the EOS blockchain and IPFS. During implementation there were however a few obstacles, EOS developer studio frequently crashed during tests, some features did not work as advertised or did not work at all, and there was no debugging available within the environment. They suggest, future work should ensure privacy and compliance with legal requirements (i.e., GDPR) in practice. Sober et al. [47] propose a decentralized marketplace for IoT data built on Ethereum. The marketplace also includes a proxy, a broker, and user interface to enable data trading. When measuring advantages and disadvantages, they point out the implemented solution achieved transparency, integrity, and verifiability of the data. Discovered disadvantage was it was more difficult to query the data compared to traditional databases. To query the data stored through the various smart contracts, it was necessary to iterate over the entire

storage of a smart contract and filter out the data that is of interest. It would be possible to implement certain filter methods in the smart contract, but this would cause additional costs. Nardini et al. [48] in 2020 proposed a decentralized electronic marketplace for computing resources. The idea was that anyone with spare capacities can offer them on this marketplace. Here the blockchain would act as an escrow service, making sure that the payment was there to begin with and that it gets released once the computational task has been completed and accepted. One of discovered issues was the financial overhead imposed by the blockchain. The system had some non-negligible costs for the users, which would probably make the fees higher than those of large cloud providers. They finalize by mentioning that improving response times would be necessary to have a positive impact on users adopting their framework.

In 2021, Sreedevi et al. [49] proposed a decentralized application for managing the disaster with blockchain, cloud and IoT. In the background the app would run on Hyperledger Fabric blockchain, and smart contracts would be implemented in Golang. Google Maps API was used to plot the disaster affected areas and GPS for navigation. IoT was used to track the movement of the assets in the network by showing, who is the owner of the assets and history of the transactions of the assets. In conclusion, they write that combination of blockchain, cloud and IoT is the best solution for managing disaster, by providing easy management and better communication.

Authors in the article [50] proposed a document uploading and verification dapp on Ethereum blockchain. They presented the architecture of the solution and implementation process. Decentralized app was tested on Ropsten testnet. The communication between user and the network was implemented with web3.js and Infura API.

The next 2 articles both utilized blockchain to solve a problem of data sharing. Putz et al. [51] in their study implement EtherTwin dapp that meets the complex digital twin sharing requirements of the Industry 4.0 landscape. Digital twins are complex digital

representations of assets that are used by a variety of organizations across the industry 4.0 value chain. For the implementation, Ethereum blockchain and Swarm have been used. Smart contracts were analyzed for vulnerabilities using the SmartCheck vulnerability scanner. The twin and document creation rates estimated by the experts did not present a challenge, as even the maximum values were within the performance limits of Ethereum and Swarm. They suggest the solution may be extended to other areas, including healthcare data sharing, data marketplaces and machine certifications. The second article [52] tried to solve privacy issues of the users in the context of social networks. They point out that in traditional systems, the privacy settings of a user are stored by the social network, which acts as a privileged party and could modify the user's choices to spread his/her data at any time without a user being able to prove this violation. They proposed a blockchain technology-based approach combined with a highly adaptable model to define the privacy settings of users in social networks. User can define their own privacy settings and store them on blockchain. Only the social network and user can link these settings to the user, which would make it pseudo-anonymous. The costs for obtaining such features are a few cents per user and makes the implementation cheap and effective.

Another popular use of blockchain was in traceability of various value chains. Alves et al. [53] proposes an architecture for tracing sustainability indicators in textile and clothing industry using Hyperledger Fabric. A traceability platform allows companies and consumers to gain insights into product items or lots by linking previously recorded data. The development environment was created using Fablo tool for generating Hyperledger Fabric blockchain network and it was run on Docker containers. Smart contracts were written in Golang because it is the main supported language in Fabric. For frontend, Vue.js was used together with Fablo API to communicate with the network. Miehle et al. [54] in 2019 proposed a decentralized app named PartChain for multi-tier supply chain in automotive industry. The vision of the PartChain system was to enable participants of supply chain networks to create and transfer a unique digital twin of a physical part using a mobile device. Unlike previous discussed study, they used CouchDB

with Docker and communication with blockchain was implemented using Hyperledger Composer API. The articles [55] [56] [57] all used the same technologies for implementation of their dapps, namely Ethereum blockchain, Javascript or React and web3.js. The main concern of the study [56] was the adoption of proposed solution for prescription drug surveillance, mainly because blockchain and smart contracts are still in their early stages of development. The study [55] points out to how hashed encryption and decentralized storage make data tampering difficult and hence increasing security of the system. Lastly, the study [57] compared their solution of medical supply chain with 3 other medical supply chains and achieved improvements on transparency, decentralization, scalability and user-friendliness.

Authors in next study [58] is a first study that integrated robot operating system with Ethereum blockchain. For that they implemented a specialized smart contract framework called “Swarm Contracts” that rely on blockchain technology in real-world applications for robotic agents with human interaction to perform collaborative tasks while ensuring trust by motivating the agents with incentives using a token economy with a self-governing structure. The user interface was implemented using Python and connected to blockchain web3.py. Swarm was used for storing the data from robots.

Nikhil et al. [59] implemented a use case example lottery dapp, where users can participate in lottery by paying a predefined amount. They observed that execution time is no longer an issue in the Ethereum network because the gas values may be modified. They conclude that blockchain is the most promising technology for security and storage in the future.

Mhamdi et al. [60] tackled the problem of communication between connected vehicles and surrounding objects. As a solution, they integrated the blockchain system on the internet of vehicles (IoV) network. In such network, several entities transfer information to each other. The smart contract adds the automatic and secure aspect of this transfer thanks to the transparency and immutability of the blockchain.

The following articles found the use of blockchain for the Covid19 situation. Goel et al. [61] suggested a contact tracing system for restricting the transmission of any infectious disease using Ethereum blockchain. They used Bluetooth technology to identify contacts, and their data was stored in the blockchain. The study [62] proposed a solution to decrease overproduction and underutilization waste of Covid19 vaccine by using Ethereum blockchain and IPFS. They point out the following challenges of the proposed solution: high power consumption and scalability issues, lack of industry 4.0 experts, privacy of sensitive data on blockchain, and finally interoperability with legacy systems.

Madine et al. [63] proposed a cross-chain interoperability system based on the Electronic Medical Record document sharing across two hospitals. Cross-chain interoperability represents the ability for one blockchain network to interact and share data with another blockchain network. Their algorithm implementation was based on Ethereum network, but it could be generalized for any other network that supports smart contracts. For testing, they used 2 different Ethereum network for representing 2 different hospitals. They outline limitations of their system in the form of open challenges: increasing cost of deployment, limited upgradability, and cross-industry interoperability.

Authors in the study [64] described usability of blockchain technology in a public participatory geographical information systems. They suggested blockchain should be used to have a fully open, transparent, and accountable environment for public participation. They developed a prototype dapp, where users could participate in the site selection of urban facilities. Because blockchain makes their solution tamperproof, the citizens would consider it as an accountable and trustworthy application that merely reflects their collective decision. They mention the constraints of Solidity language and lack of support for geospatial data types.

Carvalho [65] in his article proposed bringing transparency and trustworthiness to loot boxes in video game industry. Ever-increasing reliance on loot boxes in video games has been criticized for its lack of transparency since, before purchasing a loot box, players do not necessarily know the possible items they can win and the associated probabilities. They present their idea in proof-of-concept application, built on Ethereum blockchain with Javascript frontend. After the player opens a loot box, an Etherscan link is generated to assure that transaction was created. They evaluate their solution based on accuracy, security, and costs and conclude that it is accurate, secure, and inexpensive.

Yu et al. [66] and Arulprakash and Jebakumar [67] implemented use of blockchain in crowdsensing. Crowdsensing is a paradigm, where, in exchange for rewards, mobile users collect and share location-specific data values. Because traditional centralized systems are prone to attacks, intrusions, single point of failure, manipulations, and low reliability, they proposed an Ethereum-based system. In [66] they carried out extensive experiments on a real-world crowdsensing system to demonstrate their smart contract protocol's effective-ness and practical performance by developing a full-stack on-chain and off-chain dapp. They used IPFS for off-chain privacy-preserving data storage. In [67] they additionally incorporated a reward system for workers. Moti et al. [68] proposed an Android-based crowdsensing framework for mobile devices. They proposed the proof-of-location protocol to assist their solution on guaranteeing that agents participating in information elicitation mechanisms were at the expected locations when reporting their measurements.

Padghan et al. [69] proposed a business model for cooperative energy sharing using Ethereum blockchain for an appropriate and fair accounting of the energy transferred. A case study with four prosumers using realistic data was presented to demonstrate the usefulness of the proposed framework. The total bill for four prosumers was $527.7 per month, which was reduced to $285.15 per month when using the proposed framework.

They concluded that their framework guaranteed the energy buying or selling of each prosumer in an efficient manner.

Construction industry worldwide suffers from poor payment practices. In [70] the authors proposed a smart contract system for security of payment of construction contracts. Their solution guarantees availability of the funds for a progress payment period by blocking the projected progress payment amount at the beginning of the progress payment period. posed system were revealed through a real construction project. The main contribution of the proposed smart contract payment security system is that it provides a secure, efficient, and trustworthy platform for security of payments of construction contracts, without requiring a trusted intermediary such as lawyers or banks. The system is based on Ethereum blockchain with user interface being developed in Javascript.

The next group of articles was focusing on sharing and storing data. For better comprehension we presented the technology stack in Table 15. All studies built their solution on Ethereum blockchain using Solidity for smart contract development.

Table 15. Technology stack summary of selected studies.

Ref.	Environment tool	Testnet	Frontend	API	Storage
[71]	Geth	Rinkeby	Django	web3.py	/
[72]	/	Ethereum Mainnet	Javascript	web3.js	Swarm
[73]	Truffle, Ganache	Metamask	React	web3.js	IPFS
[74]	Ganache	Metamask	React	web3.js	IPFS
[75]	local test network	/	Android	Infura, web3j	IPFS

From the table we can observe different approaches have been used for implementation and how the technology stack changed based on their choice. We can observe the only

interdependent layers were frontend and API, where API was in based on the same programming language as frontend language. Django framework is based on Python; therefore, API was in python as well. React is based on Javascript, so the API selected was written in Javascript. The same goes for Android, which is Java/Kotlin based, and web3j is written in Java. We can observe that other technologies like storage or testnets could be used with different platforms. In [71] they observe that with an increasing demand for data re-use their solution scaled similarly to any Ethereum based network in private and in public configurations of the network. In general, mining a transaction took between 10-20 seconds and concluded their model introduced low computational cost. In [72] the authors proposed an architecture involving the encryption of data with strong encryption ciphers and decryption keys that are specific to each user. A smart contract was used for requesting and providing access to data uploaded to the Swarm decentralized network storage. The challenge they mentioned was access revocation, if a user that has received access to a file is now considered untrustworthy, or the file-key has been otherwise compromised, access to the original file should be revoked. In their proof-of-concept implementation they included functionality of re-encrypting the file, to partially solve this issue. Wong and Heng [73] in their study suggest to use React frontend library for more secure solution instead of plain HTML, CSS and Javascript. Furthermore, React library was nominated as the top Javascript trends in 2019. Cheng and Heng [74] demonstrated how Ethereum blockchain and IPFS can be used as a decentralized alternative to centralized storage systems. They performed a comparison between centralized and decentralized storage, where identified strong points of decentralized system were anonymity and inalterability, and weak points were low scalability and vulnerability of malware spreading by disguising it as valid content. Tang et al. [75] proposed a data storage based on Ethereum blockchain and IPFS for Android mobile devices. Additionally, they implemented Diffie-Hellman key exchange as encryption scheme to protect user data. In the conclusion, they added that their model can guarantee the high throughput of data storage and has the characteristics of traceability and tamper-proof design.

Most of the existing blockchain frameworks are unsuitable for miniature Internet of medical things (IoMT) devices due to high computational and storage requirements. Authors in [76] aimed to overcome this challenge by proposing a private blockchain framework in which different stakeholders of a medical system such as patients, doctors, IoMT devices, etc. act as nodes, creating a decentralized network in which physiological data output by IoMT devices can be stored securely and tamper-free forever. They proposed an implementation in a Raspberry Pi network, using Proof of Authority (PoA) consensus mechanism which has minimal computational requirements. They ensured data confidentiality with double-encryption mechanism using Elliptic Curve Integrated Encryption Scheme. They mentioned that for a standard Ethereum network, the block gas limit is 8000000, and transaction gas limit is 21000. They have used the same for the proposed system to achieve a minimum transaction speed of 25 transactions per second. The solution consists of a Python module to fetch sensor data, a Javascript frontend module and web3.js API to invoke smart contract function.

Praitheeshan et al. [77] proposed private and trustworthy distributed lending model using Hyperledger Besu, which is an Ethereum client that enables private smart contract transactions and permissioning in a private Ethereum network. They chose Besu because it is more suitable to develop financial applications that require security or high performance in private transaction processing since it is scalable, reliable, and offers secured off-chain privacy. Secondly, they used Orion, a private contract manager, to maintain transactions private and distribute transaction data among the participants only.

Author in [78] suggested the use of Ethereum blockchain and IoT in a fund management of financial poverty alleviation system. Their solution records personnel information, poverty alleviation data and funds in the blockchain to ensure the data credibility and security. IPFS was used to query and store large text data. They implemented a complete application platform, with a few concerns. They mention that performance improvements are necessary and additional encryption to protect the data privacy.

The raw volume of live streaming video continues to expand rapidly as a critical component of the creator economy. In centralized video streaming platforms, the platform owner controls most of the content uploaded on the centralized video platform, not the content producer. Lopes et al. in [79] implemented an Ethereum-based live streaming service with pay-as-you-watch business model. What was interesting in this study, was the stack of technologies used. Their solution was based on Voodfy, which is a decentralized video hosting platform utilizing Filecoin combined with IPFS and Livepeer. Textile ThreadDB API was used to upload content on IPFS decentralized storage. Libp2p protocol, a protocol for developing peer-to-peer network was used for implementing decentralized chat. Superfluid Finance protocol framework was used to enable real-time finance transfer between users' accounts. Every content a streamer creates is minted as a non-fungible token (ERC-721 NFT) and stored to IPFS. To grant access to this content only when a user is subscribed, the Unlock protocol is used, which allows locking/unlocking the content. For providing real time insights of money streamed a decentralized query protocol Graph is used, which allows indexing, querying, and caching the data stored on decentralized data systems.

Another study based on NFTs was written by Arora et al. [70] and they demonstrated NFTs in an Ethereum-based tile-guessing game implemented in React.js. A player earns an NFT by successfully matching 2 matching tiles. After the match, the player is informed about the gas price required for minting the NFT and after confirmation, the NFT is minted and added to the player's wallet. For implementation of NFT they used OpenZeppelin library.

Sedrati et al. [80] pointed out that existing governance frameworks were not sufficient in the IoT context, so they proposed a solution incorporating blockchain. For proving the feasibility of their concept, they implemented a hospital smart parking system using attribute-based access control (ABAC) deployed on Ethereum blockchain. They

concluded that delegating access control policies to the Blockchain ensures a transparent and non-editable system, where no information can be overwritten.

Rafati Niya et al. [81] highlighted that current systems for selling digital event ticket are subjects to counterfeiting, profiteering, and black markets. To overcome these issues, the authors proposed a decentralized ticketing platform using Ethereum blockchain. To avoid ticket frauds, they set a rule that price of resold ticket cannot be higher than original price of the ticket, and the validity of the ticket can always be verified on the blockchain. They used IPFS for storing metadata of the events and tickets. Frontend was implemented in Vue.js.

Authors in [82] pointed out the issues India has when it comes to maintaining land ownership and land records. For that, they proposed to store the ownership of land in blockchain. They implemented a decentralized framework for land registry, that can authorize the originality of the land registration documents and convert them into data to be stored on Ethereum blockchain. Additionally, the framework allows fiat currency for transaction. Angular framework was used for frontend development. They suggested the system could be improved incorporating tokenization.

5.4. Discussion

Based on the articles read and analyzed, we continue to answer 4 research questions. We answer each question individually in a separate subsection. To help and to facilitate the review of the results of the systematic literature review, we created a table located in the Listings, under Appendix A.

5.4.1. RQ2: What are the technologies, platforms, frameworks, and tools for developing Web3 solutions?

During our systematic literature review, a total of 55 studies were analyzed, which at the same time represent 55 unique Web3 solution examples. We carefully examined which technologies were used for implementation of each solution and collected them

in a spreadsheet, look Appendix A. We will follow with answering RQ2 and holding a discussion in separated paragraphs for each of technologies, platforms, frameworks, and tools.

According to the studies, the most popular technology was Ethereum, where most of the studies implemented their solution on Ethereum blockchain. There were 3 studies found, that built their solution on Hyperledger Fabric blockchain, and only one study that used EOS blockchain. The programming language used for developing smart contracts for Ethereum blockchain was Solidity in all studies, for Hyperledger Fabric we detected use of Golang, which is preferred by the community, and one usage of Javascript. EOS blockchain smart contracts were implemented in C++. We suspect the reason for so many studies to use Ethereum blockchain comes from its API implementations available for the popular frontend development frameworks. In the studies, we detected Ethereum API libraries written in various programming languages e.g., web3.js in Javascript, web3.py in Python and web3j in Java. Additionally, at the time of writing, Ethereum has the second highest market cap in cryptocurrency and there are almost 56,000 repositories on GitHub. In comparison there are only around 8,000 Hyperledger Fabric repositories. We were surprised there were no studies found building their solution on Polygon or Solana, which have around 16,700 and around 12,500 repositories, respectively.

There were 3 platforms found in total, namely Infura, Voodfy and OpenZeppelin. Infura was mostly used for its API to easily connect with IPFS. However, in [68] it was used for hosting Ethereum node cluster, because hosting a node on mobile device is energy demanding and demotivating for user agents. Voodfy is another platform, that was used in [79] for storing the live streamed videos. As of today, it seems this platform is no longer supported as there have not been any update since Summer 2021. Lastly, OpenZeppelin was used in [83] for implementation of NFTs. The platform provides predefined smart contracts to reduce coding complexity.

Frameworks found mostly consist of the frontend frameworks, and it seemed like any frontend framework could be used to develop Web3 applications. The most popular frameworks were Javascript based, with Vue.js leading, following with Angular, Next.js and React Native. We would include React.js here, but it is a library and not a framework. Other frontend frameworks used were ASP.net, Android, and Django. Finally, the important framework that was used by many studies is Truffle, which is used for compiling and migrating smart contracts. It was used by at least 28 studies.

The tools for Web3 development observed in the studies were used for creating a blockchain client i.e., blockchain node. The most used was Ganache, which is used to create Ethereum client node. Other Ethereum-based tools were Geth, Parity Ethereum and Remix IDE. We suspect most researchers chose Ganache, because of its convenient and easy-to-use GUI, where Geth is a command-line tool and might be more complicated to use. Hyperledger Fabric involving studies used Docker tool to establish the blockchain network.

We would like to point out another subject that was not mentioned in the research question i.e., approach to decentralized storage. We found 2 different approaches of implementing decentralized storage. The more common one was storing files on IPFS decentralized storage, and second one was using Swarm decentralized storage.

5.4.2. RQ3: How connected and dependent is Web3 to blockchain and smart contracts?

To answer this question, we must go back to the definition of what Web3 is. We defined that Web3 focuses on decentralized data structure, AI driven services and edge computing infrastructure. Blockchain is a decentralized distributed ledger, and that already covers decentralized data structure and edge computing infrastructure. Then we are missing AI driven services, and here the smart contracts step in. Smart contracts are autonomous, which means once they are deployed, they will operate on their own. We could say blockchain is a foundation for Web3 and smart contracts are the AI

workers creating value. As expected, the results of SLR showed that all analyzed studies used blockchain technology to implement their solutions. Therefore, we can conclude that Web3 depends on blockchain and smart contracts.

5.4.3. RQ4: What is the full-stack for Web3?

By analyzing the current literature on development of Web3 solutions, we got better understanding on how the full-stack of technologies for Web3 development should look like. First, we gathered the technology stack from the examined studies, ordered it into logical layers and added them to a mind map for better presentation. This can be seen on Figure 4. Then we did additional research of gray literature and our own experience and included the findings in the existing mind map. On the figure can be seen our proposed mind map of the full-stack technologies for Web3 development. We defined 8 layers, with 5 of them, highlighted with green border, being the minimal stack for developing a complete Web3 solution. Network layer presents the blockchain and should be the starting point for developers when they are choosing the technologies to build a solution. Developer environment is an important aspect, because most of the smart contract testing will be performed there. The choice of frontend framework is left to developer's preferences and would usually be chosen based on whether the developer already had experience working with it. Communication layer is the bridge between frontend and smart contracts. This is where frontend interacts with smart contract's functions. The last of minimal required stack for development is identity layer, which enables connecting the wallet with the application. The most important additional layer is storage layer, which is required in any scenario where large data must be stored. Block explorers' layer adds the ability to scan the blockchain and see details of any transaction on the network. Finally, there is Web3 supporting platforms' layer which is lately gaining the most attention and the whole communities get built around it. Web3 platforms provide developers the necessary tools to make the process of building Web3 application easier and faster. While the layers presented will likely stay the same, the projects and with them communities have a potential to evolve drastically.

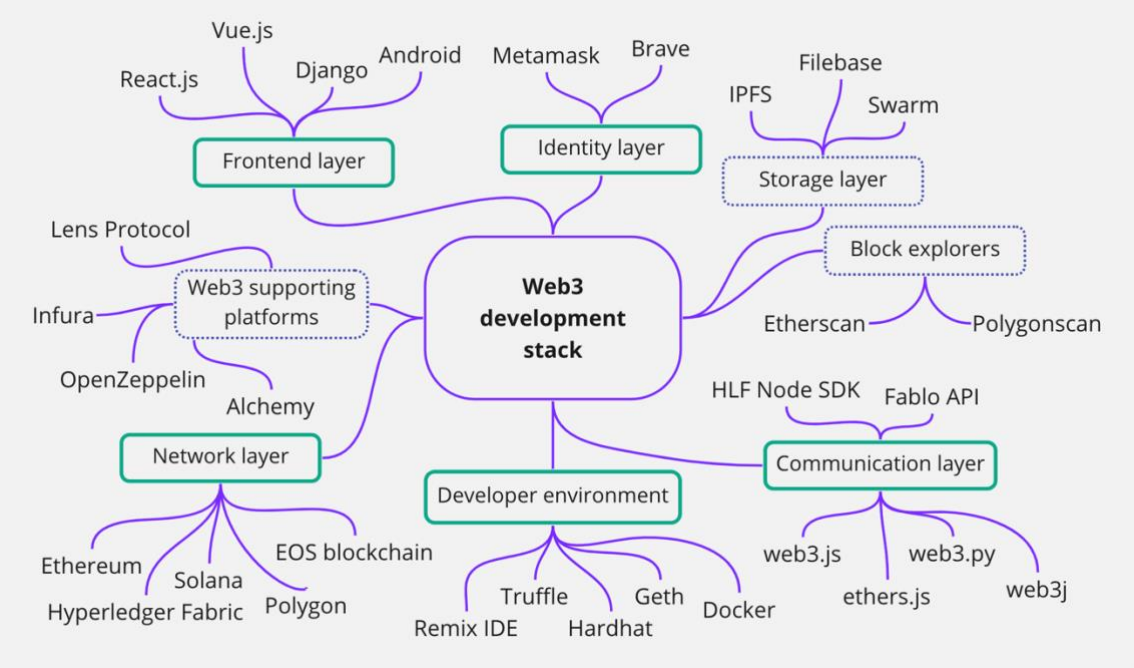


Figure 4. Web3 development technology stack.

5.4.4. RQ5: Are non-fungible tokens (NFTs) necessary for Web3 development?

While NFT cannot exist without a blockchain, a blockchain does not necessarily require an NFT. We have observed during the analysis of literature that most studies did not implement NFTs in their solution, to be exact, only 2 of the studies implemented NFTs. There was a use case for implementing a live video streaming platform, where content is stored as NFT. [79] The other implementation used NFTs as a reward for successfully playing their puzzle game. [83] We concluded that NFTs in Web3 development have a role in specific niche of Web3. A good demonstration of the role NFT plays in Web3 was proposed in a short article by All NFT Space [84] and can be seen on Figure 5. Currently, the most common use of NFTs is in presenting ownership of a digital asset, usually of digital art or an item earned in a video game. However, there is an increasing number of practices using NFT in different areas i.e., ticketing systems or personal identification.

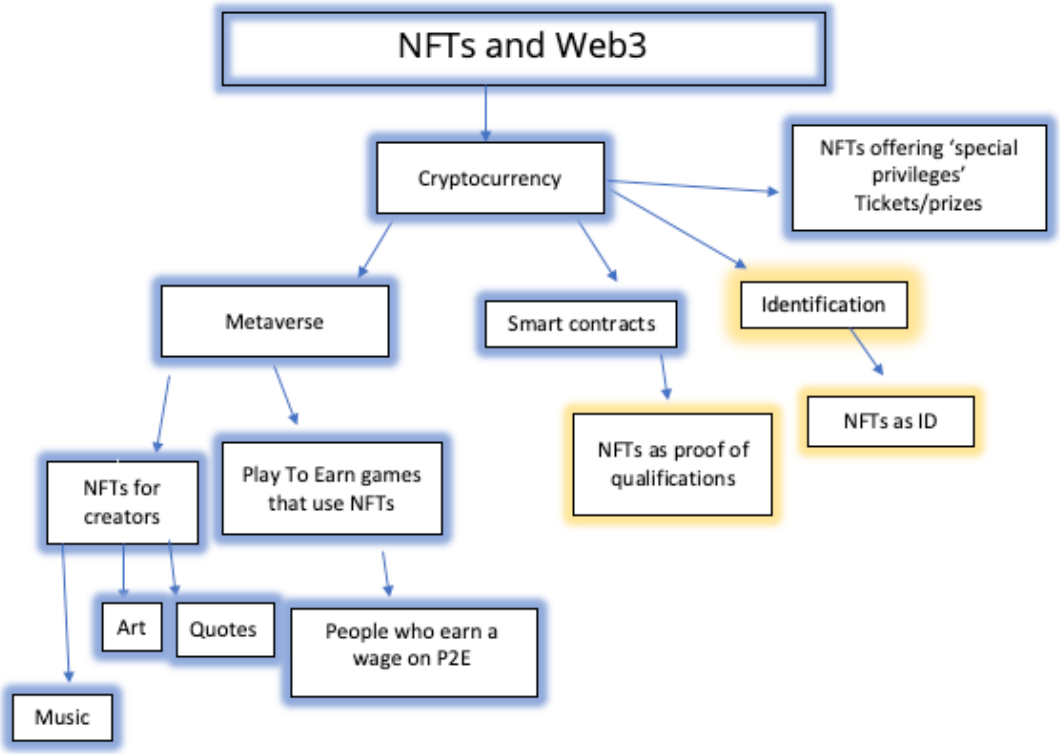


Figure 5. Role of NFTs in Web3.

6. EXPERIMENT

The differences between Web2 and Web3 have already been discussed in chapter 3. However, we wanted to further explore how the differences reflect on development of the applications. For that purpose, we implemented a decentralized application using a full-stack for Web3 development that was previously presented. We aimed to use at least one technology from each stack layer, for the experiment to be wholesome. We then suggest how the same solution would be implemented using Web2 full-stack technologies, how the architecture of the system would look like, and which technologies would be used to implement it. Finally, we discuss the observed differences between Web2 and Web3 solutions.

As a proof-of-concept application, we developed WeddingFund, which is a decentralized application designed for collecting wedding presents in form of cryptocurrencies and wedding card wishes. The idea for a solution comes from the fact that the newlyweds can be young couples without big savings and often prefer to receive money rather than meaningless gifts from the invitees, in addition, the wedding itself is expensive and they would like to afford a pleasant honeymoon. Our suggested solution offers donating any amount of Ether to a created fund for newlyweds, which must be accompanied by a digital wedding card wish in an image format. The donations are transferred to the fund on a smart contract and can be collected anytime by the owner of the contract. Wedding card wishes are stored on IPFS decentralized storage.

6.1. Environment setup

In this chapter we discuss the technologies used for setting up the environment and implementing the solution. Most of the technologies were selected based on our previous experience using them. Tools and frameworks for development were installed and ran using Node Package Manager (npm) version 8.5.5. and Node.js version 16.13.1. The Integrated Developer Environment (IDE) tool we chose for our solution was Visual Studio Code. The important extensions we installed were Solidity and React extensions. For the local testing environment, we used Hardhat, which offers deploying smart

contracts, running tests and debugging Solidity code on a local Ethereum blockchain. It is a great tool for Solidity debugging and it displays Solidity stack traces, console.log and explicit error messages when transactions fail. Smart contracts were implemented in Solidity and compiled with Hardhat built-in compiler. They were deployed to Hardhat network and tested using scripts written in Javascript. For public testing, Alchemy was used to host our Ethereum node on the Goerli test network. Alchemy also provided us with developer dashboard for the in-depth statistics insight. For developing on the Goerli testnet we had to own some of its cryptocurrency Goerli ETH (GOR), which we obtained from Goerli Faucet. It allows to obtain 0.2 GOR each day. To connect our wallet with Goerli testnet we used MetaMask browser extension on Google Chrome Web Browser. For testing our solution, we created multiple MetaMask wallets.

Our application required of storing wedding card wishes in image format and storing them on blockchain would be inefficient and expensive. We decided to use IPFS decentralized storage and Infura platform to host our IPFS node. Infura, like Alchemy, provides detailed usage statistics of the hosted node. Frontend framework we decided to use was Next.js, a Javascript framework built on React. The reason for this decision was that we had previous experience with React.js, and we believed Next.js would meet all the requirement of our project. Since Next.js is Javascript framework, we had to choose a Javascript-based APIs for API layer. We decided to use ethers.js for interacting with the smart contracts and ipfs-http-client to interact with IPFS. To observe the transactions on public Goerli testnet, we used Goerli Etherscan block explorer. Figure 6 demonstrates the designed system architecture of WeddingFund decentralized application.

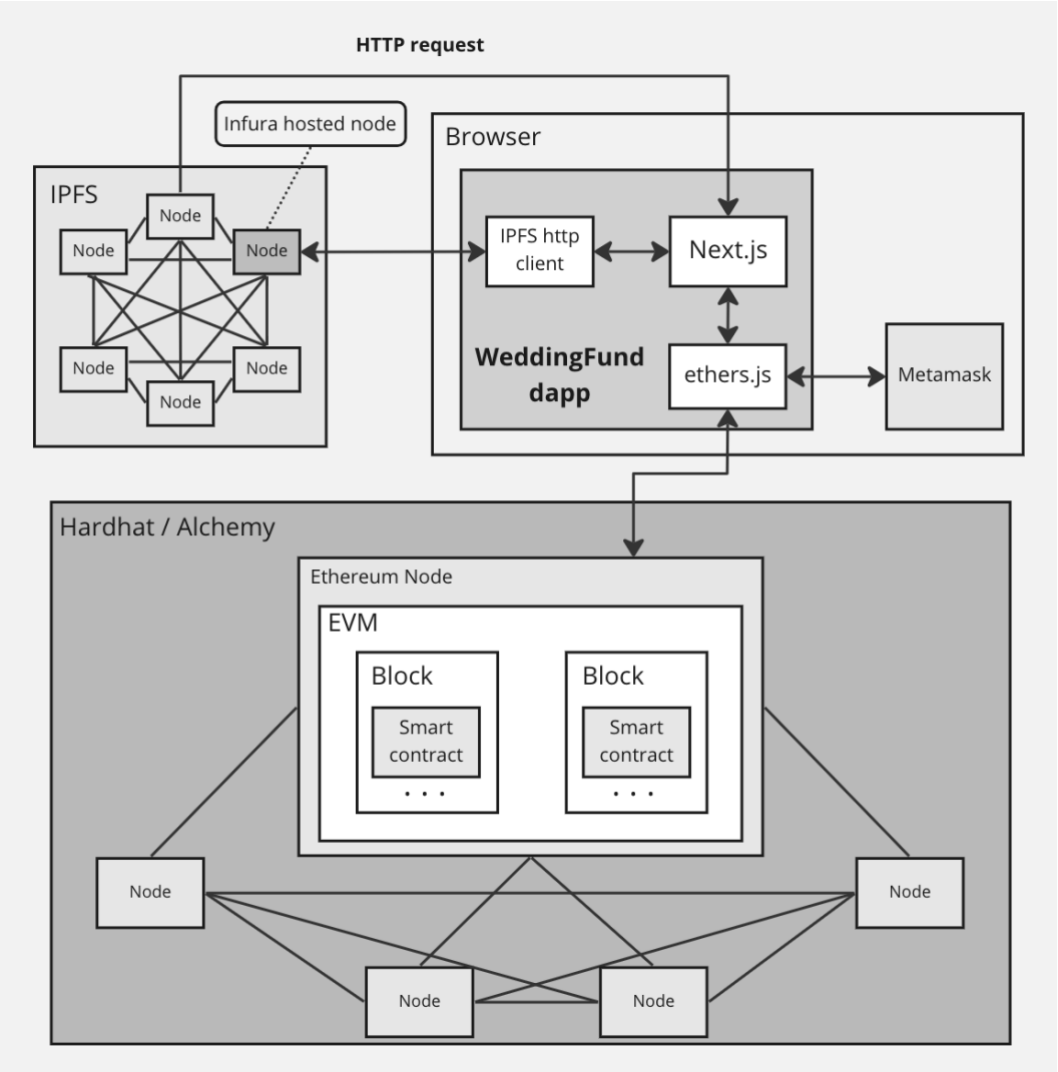


Figure 6. WeddingFund dapp system architecture.

6.2.Implementation

6.2.1. Implementation of Web3 solution

We began by creating a sample project using Hardhat. From the given options we selected the basic sample project. On Figure 7, we can see the project structure that was created. Important folder created are *"contracts"*, this is where we store our contract files, *"scripts",* this is where we store our scripts for interacting with smart contract, and finally *"hardhat.config.js"*, where the configurations for Solidity version and deployment

settings are defined. The folders *"artifacts"* and *"cache"* were generated after deploying the smart contract.

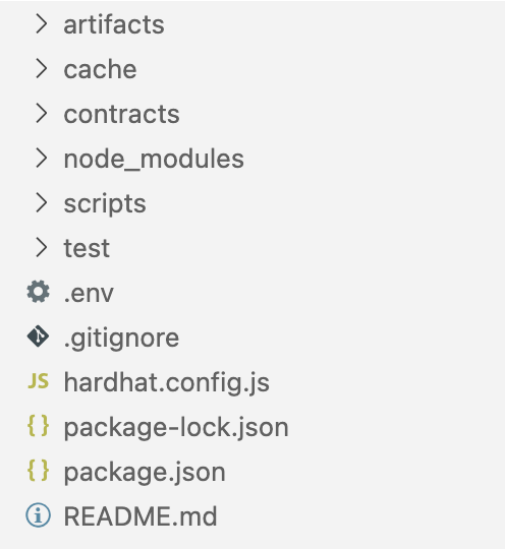


Figure 7. Hardhat generated project structure.

The first thing we implemented was WeddingFund smart contract, which can be found under Appendix B. Here we defined the Memo object, where we will store the address of the sender, timestamp, name, message and IPFS file path. All the operations that smart contract can execute are defined here, namely, paying the wedding donation, withdrawing funds, and retrieving all Memo objects. Then, we implemented the script for deploying the contract on the network, look Appendix C. Here we use methods *getContractFactory()* and *deploy()*, which Hardhat provides us. Next, we implemented the script for testing the smart contract on a local Hardhat network. The script can be found under Appendix D. Here, we used Hardhat to generate a few dummy accounts for us, so we can test the contract with multiple participants. Hardhat by default assigns every account with 10000 ETH. We use one account to deploy the contract and become the contract's owner, and the other 3 accounts to pay wedding donations. After that we withdraw all donations and transfer it to owner's account. Lastly, we print all stored Memos. We print the balances of all addresses before the payment, after the payment, and after the withdrawal. We can see the example of running the test script on the Figure 8. Address 0 represents contract owner, address 1 is one of donor accounts, and address 2 is the address of the deployed smart contract. We can observe that small

amount was paid from owner's account for deploying the contract. After the payments we can see address 1 is missing a bit over 1 ETH, because 1 ETH is paid, and small amount is deducted as gas price for the transaction. The balance of address 2 is 3.0 ETH, because 3 dummy accounts donated 1 ETH each. After withdrawal we can observe the balance of contract is back to 0 ETH and the owner received a bit less than 3 ETH, since small amount was deducted as gas.

```
Wedding fund deployed to  0x5FbDB2315678afecb367f032d93F642f64180aa3
--- start ---
Address 0 balance:  9999.998461624375
Address 1 balance:  10000.0
Address 2 balance:  0.0
--- after payments ---
Address 0 balance:  9999.998461624375
Address 1 balance:  9998.999707521762723376
Address 2 balance:  3.0
--- after withdrawal ---
Address 0 balance:  10002.998415718991159167
Address 1 balance:  9998.999707521762723376
Address 2 balance:  0.0
--- memos ---
At 1665261646, Bob (0x70997970C51812dc3A010C7d01b50e0d17dc79C8) said: "Have fun
on honeymoon!" + hash: hash1
At 1665261647, Samantha (0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC) said: "Exci
ted for the wedding!" + hash: hash2
At 1665261648, George (0x90F79bf6EB2c4f870365E785982E1f101E93b906) said: "Enjoy
this donation. :D" + hash: hash3
```

Figure 8. Example running the test script.

After we had seen the implementation was working, we began with implementation of the user interface i.e., frontend. First, we configured new projects on Alchemy and Infura platforms. The process of creating a project on Alchemy platform is seen on Figure 9. The platform at the time of writing supported 6 blockchains, from which we chose Ethereum. Networks available for Ethereum were Mainnet or Goerli network, other networks were marked as deprecated and were disabled. We selected Goerli network, so we don't have to spend real currency for our solution. After we created new project, we can access the API key to connect to the platform. To be able to connect to Goerli network, we had to configure Hardhat config file to use the connection URL that Alchemy generated for us. This can be seen on Appendix E.

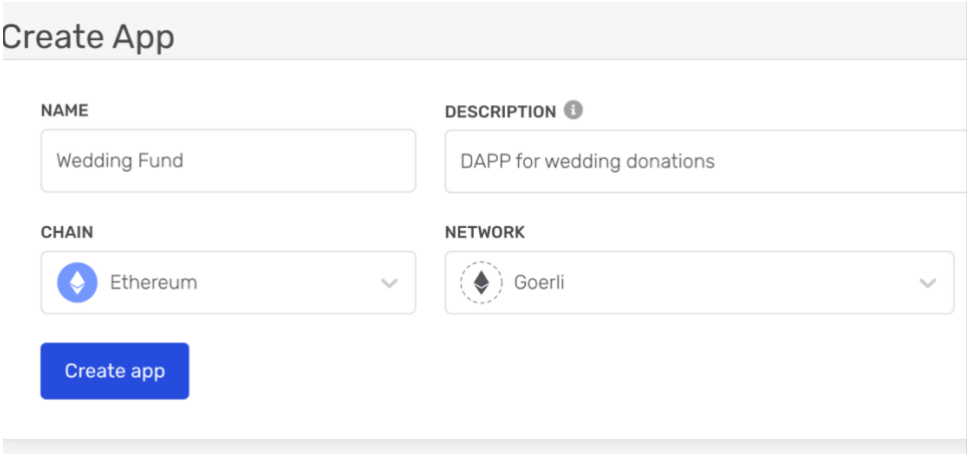


Figure 9. Alchemy create app example.

Next, we had to configure MetaMask to use Goerli test network. The creation of the network is demonstrated on the Figure 10. During creation of a new network, we start by naming the network. Then we must provide a new RPC URL, which was generated by Alchemy for us. The chain ID is used by Goerli network is 5 and the name of the currency is Goerli ETH (GOR). For the block explorer we select Etherscan block explorer. Because at the time we owned 0 GOR cryptocurrency, we had to visit Goerli Faucet website and request it. After the request we almost instantly received 0.2 GOR.

After that we had to create a new project on Infura platform. We used Infura to host our IPFS node and to provide us a developer dashboard to easily gain insight into our files uploaded to IPFS. Infura offers different pricing, depending on your needs, but for us their free plan was sufficient. It allows 100,000 of total requests per day and 5GB of storage on IPFS. We planned to host only images on IPFS, and 5GB is more than enough for a proof-of-concept application.

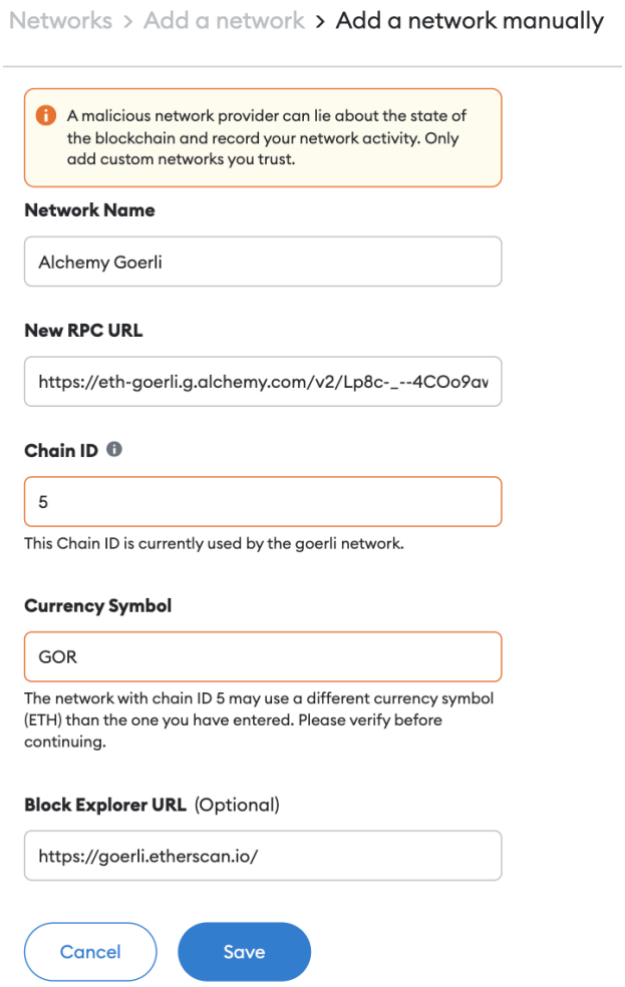


Figure 10. Creation of Goerli testnet using MetaMask.

For the frontend framework we chose Next.js. We used the *create-next-app* CLI command, to generate our project structure. Then we first installed the necessary dependencies that we would require, and they can be seen under Appendix G. We installed *bootstrap* to provide us with styling components, *ethers.js* for implementing the communication with the blockchain, and *ipfs-http-client* to communicate with IPFS. For enabling ethers.js to call communicate with smart contract, we had to generate Application Binary Interface (ABI) of WeddingFund smart contract and import it in the index.js file. ABI is a JSON file that holds the information about the precise names and types associated with the smart contract's operations. Appendix F shows the generated ABI of WeddingFund contract.

The first thing we implemented on the frontend part was checking if the user's wallet is connected. If it is not connected, the user will be asked to connect it. This is basically an authentication in Web3. The implemented functions for checking if wallet is connected and to connect the wallet are demonstrated in Appendix H. We can see the initial page on Figure 11. When user clicks on the button, he will be asked to connect his wallet using MetaMask. Initial page also displays the current value of the fund because we don't require the user's wallet for querying the network.

Figure 11. Initial page of the solution.

After user connects the wallet, he can send the donation and the wedding card wish using the form shown on Figure 12. First, we had to configure IPFS client, which is shown under Appendix L. Implementation of sending the donation can be seen under Appendix I. The donation amount is in Ethers and the minimum amount that can be sent is 0.001 ETH. This function includes the implementation of uploading the wedding card wish image to the IPFS. After the image is successfully uploaded, we retrieve the IPFS path created for this image, and we store it to the smart contract combined with other data from the form. If the connected wallet is the owner of the smart contract, he can also withdraw from the fund. The code snippet of withdraw function is shown under Appendix J. When the owner withdraws the funds, the balance of the contract goes back to 0 ETH, but the Memos with images stay on the blockchain and displayed on the page.

Connected users can also see all the donations from the other donors, as it is visible on Figure 13. The implementation of retrieving all Memos from the smart contract is included under Appendix K. We first retrieve the list of Memos from the smart contract, and we iterate over each of them and display them on the webpage. For displaying the images, we use the IPFS path of the image to find where each image was stored.

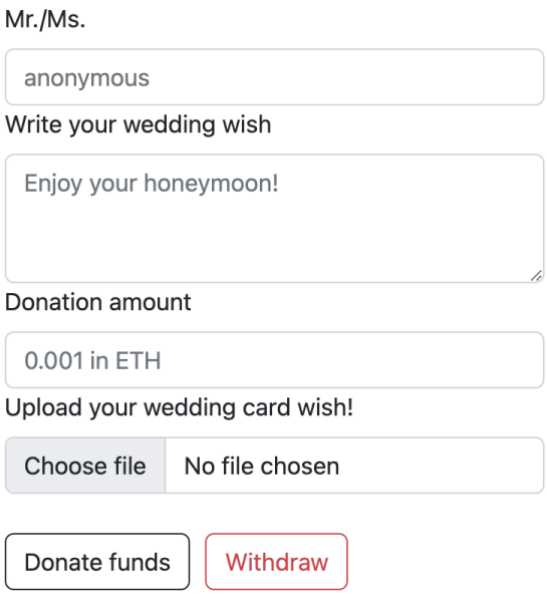
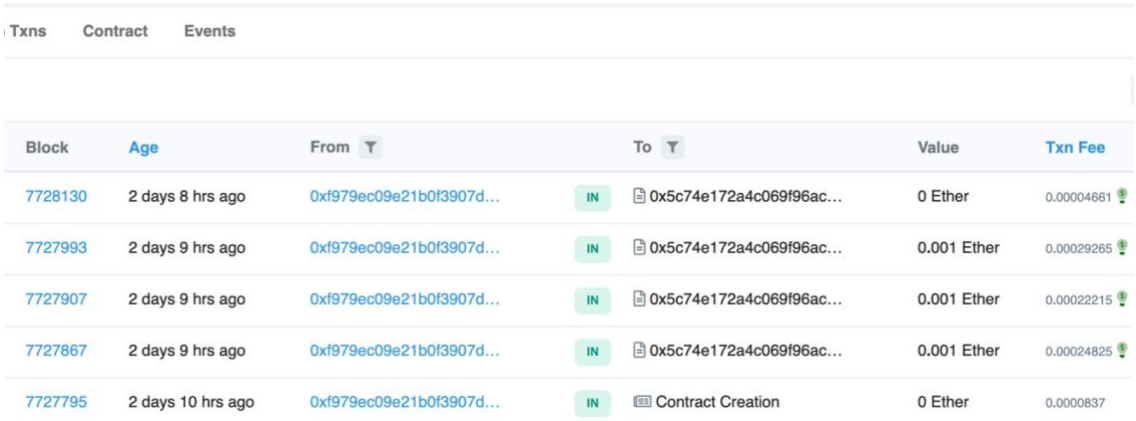


Figure 12. Donation form on WeddingFund dapp.

Figure 13. Displayed donations from all donors.

We used block explorer Etherscan to watch the transactions of our solution on a public test network Goerli. On Figure 14, we can see all the transactions that were executed with corresponding gas prices of the transaction. First transaction starting from the bottom represents the contract deployment to the network, where we can see from which address the transaction was sent and the transaction value. Here we can observe the initial value of the contract was 0. The 3 transactions in the middle represent donations to the fund, where each donation was 0.001 ETH. From these transactions we can already find the address of the contract under column "To". The transaction on the top is the withdrawal of the funds, and we can notice the gas amount (i.e., Txn Fee) is lowest during this operation, which is because we don't require a lot of computational power for withdrawing funds. The highest gas price was during donations, because here we were adding Memos to the block.


Txns Contract Events

Block	Age	From		To	Value	Txn Fee
7728130	2 days 8 hrs ago	0xf979ec09e21b0f3907d...	IN	0x5c74e172a4c069f96ac...	0 Ether	0.00004661
7727993	2 days 9 hrs ago	0xf979ec09e21b0f3907d...	IN	0x5c74e172a4c069f96ac...	0.001 Ether	0.00029265
7727907	2 days 9 hrs ago	0xf979ec09e21b0f3907d...	IN	0x5c74e172a4c069f96ac...	0.001 Ether	0.00022215
7727867	2 days 9 hrs ago	0xf979ec09e21b0f3907d...	IN	0x5c74e172a4c069f96ac...	0.001 Ether	0.00024825
7727795	2 days 10 hrs ago	0xf979ec09e21b0f3907d...	IN	Contract Creation	0 Ether	0.0000837

Figure 14. Transactions on Block Explorer Etherscan.

6.2.2. Concept of Web2 solution implementation

To be able to compare the implemented Web3 solution with the equivalent Web2 solution, we prepared a concept of system architecture of Web2 application. For the frontend implementation we selected the same framework Next.js. In developing Web2 solution we would additionally have to implement backend. For that we would use Node.js, more specifically Express.js, which is a de facto standard server framework for

Node.js. While designing a Web2 solution concept we wanted to cover the following functionalities:

- authentication: We would only allow authenticated users to donate wedding present. For this we proposed using OAuth with Firebase API. OAuth is an authentication protocol that allows users to approve one application interacting with another on their behalf without giving away their password. Google Firebase is a platform that provides API for the implementation of OAuth and a developer dashboard that shows users' statistics.
- donating funds: For donating funds we would use PayPal API for securely sending the funds to a designated PayPal account. PayPal is a company that operates as a payment processor for online payments, for which it charges a fee. After a user would confirm the donation by pressing the button, a PayPal window would open with further instructions to login to PayPal account and confirming the donation.
- storing wedding card wishes: Here we had two options. We could store the images on our backend server, in which case we, as the owner of the server would have total control over the stored data. However, it would provide bad user experience for the users that try to connect to our application, but live far away from the server, because of the big latency. The second was using Amazon Cloud storage, namely, Amazon S3, for hosting our data and taking care of selecting the best server, when a user tries to connect to the application. We selected the latter for our concept.
- storing other data: The information from the users' authentication and details from donations would be stored on our server's storage in a MySQL relational database. This data would be hosted and controlled by us.

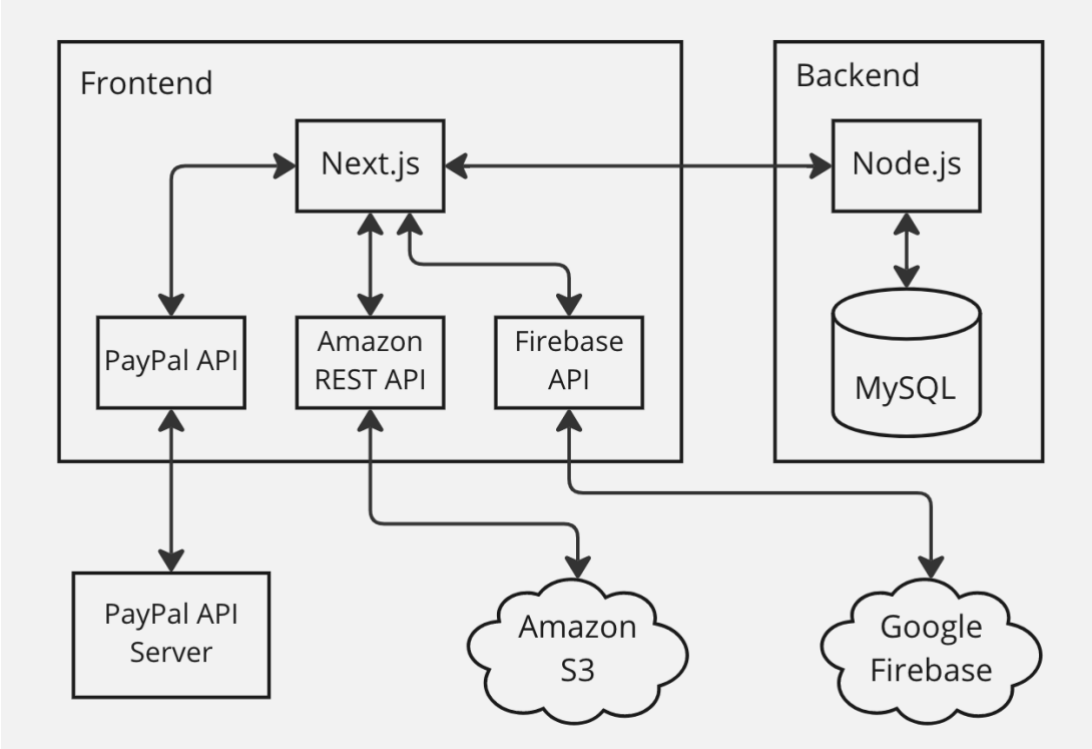


Figure 15. System architecture concept of Web2 application.

6.3. Discussion

The implemented WeddingFund solution is a fully decentralized application built on Ethereum blockchain, with its main purpose of donating funds and wedding card wishes for the newlyweds.

The main benefits of our dapp are:

- It is decentralized: There is no central authority that could control the data of the users. There is no single point of failure, which means the details of donation and wedding card wishes should be always accessible with no downtime. There is no service fee to be paid for processing the donation, apart from the small fee of transaction gas.
- It is anonymous: The interaction with the application is created using MetaMask wallet, where only user's public address is shared. If that address has not been compromised, the anonymity is guaranteed.

- It is transparent: All donations are publicly visible.
- It is tamperproof: The data is safely stored on Ethereum blockchain and cannot be tampered. Additionally, it is stored permanently.

Our dapp was tested on Goerli test network. In the Table 16 we can see the gas prices of each operation and their value in USD at the time of writing. The lowest transaction fee was during withdrawing funds, and the highest during sending the donation, which makes sense, because we were writing data to the block.

Table 16. Gas prices in ETH and USD.

Operation	Gas price in ETH	Gas price in USD
Deploying smart contract	0.000083	0.11
Donation 1	0.000248	0.32
Donation 2	0.000222	0.29
Donation 3	0.000292	0.38
Withdrawing funds	0.000046	0.06

When we compare the Web3 and Web2 system architectures, we immediately see that in Web2 there is no decentralization. Every component is centralized, where we use centralized backend server, and 3 centralized services, namely PayPal, Amazon S3 and Google Firebase. This means that data from the application is shared with other 3 companies, and we are not in control of that data. Apart from data, we usually must pay these companies to use their services or cloud, even though they might offer free limited plans. In the Web2 there is also no anonymity, because we require some sort of authentication and we are using Google Firebase, which already provides us at least email address of the user. Another difference is there is no transparency, as PayPal is handling the payments, only the donor and the owner of the fund can know the amount of the donation. The storage of the wedding card wishes on both solutions is distributed, where IPFS uses peer-to-peer node network for storing files, and Amazon is using Content Distribution Networks (CDNs). The difference between the approaches is

Amazon S3 is centralized and controlled by Amazon, while IPFS is decentralized and controlled by users. We cannot precisely determine the complexity of the development of Web2 solution to compare it to the complexity of Web3 solution development but based on the concept of system architecture and our own experience with developing Web2 applications, we could even argue that Web2 is more complex, since it additionally incorporates MySQL backend storage.

One of the observed issues of our dapp was occasional slow loading speed of the images from IPFS. This would commonly happen right after sending the donation and if the image was a large file. Another disadvantage is that on average transaction time is 15s and this must be considered when implementing frontend, to achieve a fluid user experience.

There are some propositions for future work. The application could be upgraded into a platform that would allow everyone to create multiple funds, with few ideas including charity, crowdfunding for project or weekly/monthly allowance. Another proposition is to implement a function that would allow to close the fund from future donations, for example when the wedding is over, there is no more reason for the fund. Additionally, we could implement timed transaction, that would withdraw funds at the exact predefined date and time, for example as a birthday present or at the end of a concert or charity event. In terms of improving the application, more detailed performance analysis could be conducted.

7. CONCLUSION

This final chapter will depict the main highlights of the work together with the initial objectives that were achieved and propose improvements for future work.

The main purpose of this thesis was to study the technologies for the development of Web3 solutions and investigating which technologies would together form the full-stack for Web3 development. To support this study, the concepts of blockchain technologies and Web3 were explored. We introduced concepts such as Ethereum, smart contracts, crypto wallets, non-fungible tokens, and gas or transaction cost. We continued with the description of the development of the Web, where we presented in detail each version of the Web separately, and then compared them with each other for an easier understanding of the differences. In this chapter, we also presented the technologies used for the development of Web3 solutions. Then we presented the most common types of Web3 solutions and highlighted a few examples of each.

In the core of the master's thesis, we conducted a systematic literature review, which was divided into search strategy, search results, studies review, and discussion. During the search strategy, we determined the inclusion and exclusion criteria and defined the search string and adjusted it according to the specifications of each digital database. In the search results, we presented the gradual search phases to limit the results to those that would answer the selected research questions. For easier transparency, we collected the results in a table, where for each article we presented which technologies were used to develop the solution. Then we presented individual articles, where we grouped some of them together according to the similarity of the implemented solution. In the discussion, we used all the acquired knowledge to answer the research questions.

Finally, we conducted an experiment where we implemented a proof-of-concept solution based on the identified technologies. We named the resulting decentralized application WeddingFund and its purpose is to collect funds for newlyweds. First, we presented the preparation of the experiment, where we presented the development

environment and all the technologies used. Then we presented in detail the process of implementing the solution and the result. Furthermore, we compared the resulting application with the equivalent concept of a Web2 application, for an easier presentation of the differences between Web2 and Web3. At the end, we discussed possible improvements to our solution.

Regarding the hypotheses that were explained in chapter 1:

- H1: (i.e., *There is currently no unified full-stack technology for Web3.)* We can confirm this hypothesis. While the literature analysis has clearly shown the pattern of technologies used for implementing their solutions, the differences between individual approaches were still present.
- H2: (i.e., *Web3 depends on blockchain and smart contracts.)* We can confirm this hypothesis, since all the studies implemented their solutions on top of blockchain and smart contracts, as it can be seen under Appendix A.

Blockchain technology has changed and continues to change the world. It has brought innovation to many different industries, and it is enabling companies to introduce new and exciting services and capabilities. Decentralized applications are one of the by-products of blockchain technology that offer secure open-source software for everyday users and businesses. As with all advancements, it is inevitable that many of the practices currently in use will become obsolete, however, the basic approach should likely stay the same. The results of systematic literature review gave us an overview of state-of-art Web3 development technologies, which we collected and analyzed to propose a mind map of full-stack technologies for Web3 solution development. We strongly believe that the proposed stack is valid, and that the layers' concept can stay valid in future as technologies develop even further. By implementing a proof-of-concept solution we additionally confirmed the validity of the mentioned stack.

The dominant blockchain protocol in analyzed studies was Ethereum, with few mentions of Hyperledger Fabric and EOS blockchain. There was no article found implementing

other popular protocols like Solana or Polygon, that would be matching our search criteria. For the future work it would be interesting to investigate, what is the reason that there are no studies incorporating these protocols. Another idea is to conduct a comparison between the popular blockchain protocols either building the equivalent solution using all of them or comparing the concepts of the system architecture, similar to what we did in our thesis.

8. BIBLIOGRAPHY

[1] "Gartner Hype Cycle for Blockchain and Web3, 2022," *Avivah Litan*, Jul. 22, 2022. https://blogs.gartner.com/avivah-litan/2022/07/22/gartner-hype-cycle-for-blockchain-and-web3-2022/ (accessed Oct. 09, 2022).

[2] Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities: a survey," *Int. J. Web Grid Serv.*, vol. 14, no. 4, p. 352, 2018, doi: 10.1504/IJWGS.2018.095647.

[3] "Blockchain Nodes: How They Work (All Types Explained) - Nodes.com." https://nodes.com/ (accessed Oct. 10, 2022).

[4] "Developer Glossary - Bitcoin." https://btcinformation.org/en/developer-glossary (accessed Oct. 10, 2022).

[5] S. Raval, *Decentralized Applications: Harnessing Bitcoin's Blockchain Technology*. O'Reilly Media, Inc., 2016.

[6] "Decentralized Applications: The Blockchain-Empowered Software System | IEEE Journals & Magazine | IEEE Xplore." https://ieeexplore-ieee-org.recursos.biblioteca.upc.edu/abstract/document/8466786 (accessed Oct. 10, 2022).

[7] A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo, and J. Garcia-Alfaro, Eds., *Foundations and Practice of Security: 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers*, vol. 10723. Cham: Springer International Publishing, 2018. doi: 10.1007/978-3-319-75650-9.

[8] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain technology, bitcoin, and Ethereum: A brief overview," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, Mar. 2018, pp. 1–6. doi: 10.1109/INFOTEH.2018.8345547.

[9] "The Merge," *ethereum.org*. https://ethereum.org (accessed Oct. 10, 2022).

[10] E. Hildenbrandt *et al.*, "KEVM: A Complete Formal Semantics of the Ethereum Virtual Machine," in *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, Jul. 2018, pp. 204–217. doi: 10.1109/CSF.2018.00022.

[11] "What Is an Ethereum Virtual Machine (EVM)? A Beginner's Guide | Bybit Learn." https://learn.bybit.com/deep-dive/what-is-ethereum-virtual-machine-evm/ (accessed Oct. 10, 2022).

[12] W. Zou *et al.*, "Smart Contract Development: Challenges and Opportunities," *IEEE Trans. Softw. Eng.*, vol. 47, no. 10, pp. 2084–2106, Oct. 2021, doi: 10.1109/TSE.2019.2942301.

[13] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin, and F.-Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2018, pp. 108–113. doi: 10.1109/IVS.2018.8500488.

[14] B. K. Mohanta, S. S. Panda, and D. Jena, "An Overview of Smart Contract and Use Cases in Blockchain Technology," in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2018, pp. 1–4. doi: 10.1109/ICCCNT.2018.8494045.

[15] S. Suratkar, M. Shirole, and S. Bhirud, "Cryptocurrency Wallet: A Review," in *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, Sep. 2020, pp. 1–7. doi: 10.1109/ICCCSP49186.2020.9315193.

[16] M. di Angelo and G. Salzer, “Characteristics of Wallet Contracts on Ethereum,” in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*, Sep. 2020, pp. 232–239. doi: 10.1109/BRAINS49436.2020.9223287.
[17] “What is a Crypto Wallet? A Beginner’s Guide.” https://crypto.com/university/crypto-wallets (accessed Oct. 10, 2022).
[18] Q. Wang, R. Li, Q. Wang, and S. Chen, “Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges.” arXiv, Oct. 24, 2021. Accessed: Oct. 10, 2022. [Online]. Available: http://arxiv.org/abs/2105.07447
[19] “View of Key differences between Web 1.0 and Web 2.0 | First Monday.” https://journals.uic.edu/ojs/index.php/fm/article/view/2125/1972 (accessed Oct. 10, 2022).
[20] G. Kuck, “Tim Berners-Lee’s Semantic Web,” *South Afr. J. Inf. Manag.*, vol. 6, Dec. 2004, doi: 10.4102/sajim.v6i1.297.
[21] “Statista - The Statistics Portal,” *Statista*. https://www.statista.com/markets/424/topic/538/mobile-internet-apps// (accessed Oct. 10, 2022).
[22] “What is Web 1.0, Web 2.0, and Web 3.0? Definition, Difference & Similarities,” *Simplilearn.com*, Apr. 25, 2022. https://www.simplilearn.com/what-is-web-1-0-web-2-0-and-web-3-0-with-their-difference-article (accessed Oct. 10, 2022).
[23] S. Alam, “Challenges And Benefits Of WEB3 Technology.” https://www.c-sharpcorner.com/article/challenges-and-benefits-of-web3-technology/ (accessed Oct. 10, 2022).
[24] “What is Web 3? Difference Between Web1 vs Web2 vs Web3.” https://www.becomebetterprogrammer.com/web1-vs-web2-vs-web3/ (accessed Oct. 10, 2022).
[25] “Comparison Between Web 1.0, Web 2.0 and Web 3.0,” *GeeksforGeeks*, Sep. 24, 2018. https://www.geeksforgeeks.org/web-1-0-web-2-0-and-web-3-0-with-their-difference/ (accessed Oct. 10, 2022).
[26] S. Paszun, “Blockchain - Hyperledger vs Ethereum: a deep comparison,” *Espeo Blockchain*, Jul. 24, 2018. https://espeoblockchain.com/blog/hyperledger-vs-ethereum/ (accessed Oct. 10, 2022).
[27] “Decentralization for Web3 Builders: Principles, Models, How,” *Future*, Apr. 07, 2022. https://future.com/web3-decentralization-models-framework-principles-how-to/ (accessed Aug. 02, 2022).
[28] “Top 10 Best NFT Marketplaces for 2022,” *www.top10.com*, Feb. 07, 2022. https://www.top10.com/best-nft-marketplaces (accessed Oct. 10, 2022).
[29] F. Altamimi, W. Asif, and M. Rajarajan, “DADS: Decentralized (Mobile) Applications Deployment System Using Blockchain : Secured Decentralized Applications Store,” in *2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Oct. 2020, pp. 1–8. doi: 10.1109/CITS49457.2020.9232506.
[30] R. Taş and Ö. Ö. Tanrıöver, “Building A Decentralized Application on the Ethereum Blockchain,” in *2019 3rd International Symposium on Multidisciplinary Studies*

and Innovative Technologies (ISMSIT), Oct. 2019, pp. 1–4. doi: 10.1109/ISMSIT.2019.8932806.

[31] R. A. Canessane, N. Srinivasan, A. Beuria, A. Singh, and B. M. Kumar, "Decentralised Applications Using Ethereum Blockchain," in *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, Mar. 2019, vol. 1, pp. 75–79. doi: 10.1109/ICONSTEM.2019.8918887.

[32] J. Rosa-Bilbao and J. Boubeta-Puig, "RectorDApp: Decentralized Application for Managing University Rector Elections," in *2021 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, Aug. 2021, pp. 161–165. doi: 10.1109/SOSE52839.2021.00024.

[33] S. T. Alvi, M. N. Uddin, L. Islam, and S. Ahamed, "DVTChain: A blockchain-based decentralized mechanism to ensure the security of digital voting system voting system," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 6855–6871, Oct. 2022, doi: 10.1016/j.jksuci.2022.06.014.

[34] N. Sasikala, B. M. Sundaram, S. Biswas, A. Sai Nikhil, and V. S. Rohith, "Survey of latest technologies on Decentralized applications using Blockchain," in *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, Feb. 2022, pp. 1432–1436. doi: 10.1109/ICAIS53314.2022.9742768.

[35] B. Santhanakrishnan, A. K. Tyagi, and T. F. Fernandez, "Blockchain Network based Decentralized Applications for Healthcare Sector," in *2022 International Conference on Computer Communication and Informatics (ICCCI)*, Jan. 2022, pp. 1–6. doi: 10.1109/ICCCI54379.2022.9740743.

[36] M. A S, A. S, G. M. Mufeed, A. K R, and Gahana, "Converging Blockchain and Artificial-Intelligence Towards Healthcare: A Decentralized-Private and Intelligence Health Record System," in *2022 2nd International Conference on Intelligent Technologies (CONIT)*, Jun. 2022, pp. 1–8. doi: 10.1109/CONIT55038.2022.9847762.

[37] K. P. Satamraju and B. Malarkodi, "A decentralized framework for device authentication and data security in the next generation internet of medical things," *Comput. Commun.*, vol. 180, pp. 146–160, Dec. 2021, doi: 10.1016/j.comcom.2021.09.012.

[38] M. Abdulaziz, D. Çulha, and A. Yazici, "A Decentralized Application for Secure Messaging in a Trustless Environment," in *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Dec. 2018, pp. 1–5. doi: 10.1109/IBIGDELFT.2018.8625362.

[39] C. Nagadeep, M. V. V. S. Durga, S. M. Reddy, and M. A. Jabbar, "TogEther - A Decentralized Application Connects Ideas and Investors," in *2021 International Conference on Decision Aid Sciences and Application (DASA)*, Dec. 2021, pp. 813–818. doi: 10.1109/DASA53625.2021.9682273.

[40] R. Muth and F. Tschorsch, "SmartDHX: Diffie-Hellman Key Exchange with Smart Contracts," in *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Aug. 2020, pp. 164–168. doi: 10.1109/DAPPS49028.2020.00022.

[41] R. A. Mishra, A. Kalla, N. A. Singh, and M. Liyanage, “Implementation and Analysis of Blockchain Based DApp for Secure Sharing of Students’ Credentials,” in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, Jan. 2020, pp. 1–2. doi: 10.1109/CCNC46108.2020.9045196.

[42] R. A. Mishra, A. Kalla, A. Braeken, and M. Liyanage, “Privacy Protected Blockchain Based Architecture and Implementation for Sharing of Students’ Credentials,” *Inf. Process. Manag.*, vol. 58, no. 3, p. 102512, May 2021, doi: 10.1016/j.ipm.2021.102512.

[43] V. Ivankovic, Z. Shi, and Z. Zhao, “A Customizable dApp Framework for User Interactions in Decentralized Service Marketplaces,” in *2022 IEEE International Conference on Smart Internet of Things (SmartIoT)*, Aug. 2022, pp. 224–231. doi: 10.1109/SmartIoT55134.2022.00043.

[44] U. K. Shakila and S. Sultana, “A Decentralized Marketplace Application based on Ethereum Smart Contract,” in *2021 24th International Conference on Computer and Information Technology (ICCIT)*, Dec. 2021, pp. 1–5. doi: 10.1109/ICCIT54785.2021.9689879.

[45] R. Riesco, X. Larriva-Novo, and V. A. Villagra, “Cybersecurity threat intelligence knowledge exchange based on blockchain,” *Telecommun. Syst.*, vol. 73, no. 2, pp. 259–288, Feb. 2020, doi: 10.1007/s11235-019-00613-4.

[46] F. Menges, B. Putz, and G. Pernul, “DEALER: decentralized incentives for threat intelligence reporting and exchange,” *Int. J. Inf. Secur.*, vol. 20, no. 5, pp. 741–761, Oct. 2021, doi: 10.1007/s10207-020-00528-1.

[47] M. Sober, G. Scaffino, S. Schulte, and S. S. Kanhere, “A blockchain-based IoT data marketplace,” *Clust. Comput.*, Sep. 2022, doi: 10.1007/s10586-022-03745-6.

[48] M. Nardini, S. Helmer, N. El Ioini, and C. Pahl, “A Blockchain-based Decentralized Electronic Marketplace for Computing Resources,” *SN Comput. Sci.*, vol. 1, no. 5, p. 251, Aug. 2020, doi: 10.1007/s42979-020-00243-7.

[49] B. Sreedevi, S. K. Kumar, and S. Samraj E, “Decentralized Application for managing the Disaster with Block chain, Cloud &IOT,” in *2021 International Conference on Computer & Information Sciences (ICCOINS)*, Jul. 2021, pp. 328–332. doi: 10.1109/ICCOINS49721.2021.9497189.

[50] Mrs. L. S. S, Mrs. P. N, and Mrs. A. Shettar, “Block chain Based Framework for Document Verification,” in *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*, Feb. 2022, pp. 1–5. doi: 10.1109/AISP53593.2022.9760651.

[51] B. Putz, M. Dietz, P. Empl, and G. Pernul, “EtherTwin: Blockchain-based Secure Digital Twin Information Management,” *Inf. Process. Manag.*, vol. 58, no. 1, p. 102425, Jan. 2021, doi: 10.1016/j.ipm.2020.102425.

[52] G. Lax, A. Russo, and L. S. Fascì, “A Blockchain-based approach for matching desired and real privacy settings of social network users,” *Inf. Sci.*, vol. 557, pp. 220–235, May 2021, doi: 10.1016/j.ins.2021.01.004.

[53] L. Alves, E. F. Cruz, and A. M. Rosado Da Cruz, “Tracing Sustainability Indicators in the Textile and Clothing Value Chain using Blockchain Technology,” in *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*, Jun. 2022, pp. 1–7. doi: 10.23919/CISTI54924.2022.9820241.

[54] D. Miehle, D. Henze, A. Seitz, A. Luckow, and B. Bruegge, “PartChain: A Decentralized Traceability Application for Multi-Tier Supply Chain Networks in the Automotive Industry,” in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*, Apr. 2019, pp. 140–145. doi: 10.1109/DAPPCON.2019.00027.
[55] S. W. Sheng and S. Wicha, “The Proposed of a Smart Traceability System for Teak Supply Chain Based on Blockchain Technology,” in *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*, Mar. 2021, pp. 59–64. doi: 10.1109/ECTIDAMTNCON51128.2021.9425780.
[56] M. Alnafrani and S. Acharya, “SecureRx: A blockchain-based framework for an electronic prescription system with opioids tracking,” *Health Policy Technol.*, vol. 10, no. 2, p. 100510, Jun. 2021, doi: 10.1016/j.hlpt.2021.100510.
[57] S. K. Panda and S. C. Satapathy, “Drug traceability and transparency in medical supply chain using blockchain for easing the process and creating trust between stakeholders and consumers,” *Pers. Ubiquitous Comput.*, Jul. 2021, doi: 10.1007/s00779-021-01588-3.
[58] S. Mallikarachchi, C. Dai, O. Seneviratne, and I. Godage, “Managing Collaborative Tasks within Heterogeneous Robotic Swarms using Swarm Contracts,” in *2022 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Aug. 2022, pp. 48–55. doi: 10.1109/DAPPS55202.2022.00014.
[59] Nikhil, S. Panday, A. Saini, and N. Gupta, “Instigating Decentralized Apps with Smart Contracts,” in *2022 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, Jan. 2022, pp. 1–5. doi: 10.1109/ACCAI53970.2022.9752568.
[60] H. Mhamdi, A. Zouinkhi, and H. Sakli, “Smart contracts for decentralized vehicle services,” in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, Jun. 2021, pp. 1846–1851. doi: 10.1109/IWCMC51323.2021.9498954.
[61] R. Goel, U. Singh, and D. Sadhya, “Privacy Preserving Contact Tracing using Ethereum Blockchain Network,” in *2022 IEEE Region 10 Symposium (TENSYMP)*, Jul. 2022, pp. 1–6. doi: 10.1109/TENSYMP54529.2022.9864380.
[62] A. Musamih, K. Salah, R. Jayaraman, I. Yaqoob, Y. Al-Hammadi, and J. Antony, “Blockchain-based solution for COVID-19 vaccine waste reduction,” *J. Clean. Prod.*, vol. 372, p. 133619, Oct. 2022, doi: 10.1016/j.jclepro.2022.133619.
[63] M. Madine, K. Salah, R. Jayaraman, Y. Al-Hammadi, J. Arshad, and I. Yaqoob, “appXchain: Application-Level Interoperability for Blockchain Networks,” *IEEE Access*, vol. 9, pp. 87777–87791, 2021, doi: 10.1109/ACCESS.2021.3089603.
[64] M. Farnaghi and A. Mansourian, “Blockchain, an enabling technology for transparent and accountable decentralized public participatory GIS,” *Cities*, vol. 105, p. 102850, Oct. 2020, doi: 10.1016/j.cities.2020.102850.
[65] A. Carvalho, “Bringing transparency and trustworthiness to loot boxes with blockchain and smart contracts,” *Decis. Support Syst.*, vol. 144, p. 113508, May 2021, doi: 10.1016/j.dss.2021.113508.

[66] R. Yu, A. M. Oguti, D. R. Ochora, and S. Li, “Towards a privacy-preserving smart contract-based data aggregation and quality-driven incentive mechanism for mobile crowdsensing,” *J. Netw. Comput. Appl.*, vol. 207, p. 103483, Nov. 2022, doi: 10.1016/j.jnca.2022.103483.
[67] M. Arulprakash and R. Jebakumar, “Enhanced Data Privacy Preservation Model for Mobile Crowdsensing System Using Blockchain Technology,” in *Ambient Communications and Computer Systems*, Singapore, 2022, pp. 563–576. doi: 10.1007/978-981-16-7952-0_53.
[68] M. H. Moti, D. Chatzopoulos, P. Hui, B. Faltings, and S. Gujar, “Orthos: A Trustworthy AI Framework for Data Acquisition,” in *Engineering Multi-Agent Systems*, Cham, 2020, pp. 100–118. doi: 10.1007/978-3-030-66534-0_7.
[69] P. R. Padghan, S. Arul Daniel, and R. Pitchaimuthu, “Grid-tied energy cooperative trading framework between Prosumer to Prosumer based on Ethereum smart contracts,” *Sustain. Energy Grids Netw.*, vol. 32, p. 100860, Dec. 2022, doi: 10.1016/j.segan.2022.100860.
[70] S. Ahmadisheykhsarmast and R. Sonmez, “A smart contract system for security of payment of construction contracts,” *Autom. Constr.*, vol. 120, p. 103401, Dec. 2020, doi: 10.1016/j.autcon.2020.103401.
[71] V. Urovi, V. Jaiman, A. Angerer, and M. Dumontier, “LUCE: A blockchain-based data sharing platform for monitoring data license accoUntability and CompliancE,” *Blockchain Res. Appl.*, p. 100102, Sep. 2022, doi: 10.1016/j.bcra.2022.100102.
[72] M. Siopi, G. Vlahavas, K. Karasavvas, and A. Vakali, “DeCStor: A Framework for Privately and Securely Sharing Files Using a Public Blockchain,” in *Discovery Science*, Cham, 2020, pp. 280–293. doi: 10.1007/978-3-030-61527-7_19.
[73] Z.-K. Wong and S.-H. Heng, “Blockchain-Based Image Sharing Application,” in *Advances in Cyber Security*, Singapore, 2020, pp. 46–59. doi: 10.1007/978-981-15-2693-0_4.
[74] K.-W. Cheng and S.-H. Heng, “Blockchain-Based Content Sharing and Data Repository System,” in *Advances in Cyber Security*, Singapore, 2021, pp. 207–224. doi: 10.1007/978-981-33-6835-4_14.
[75] X. Tang, H. Guo, H. Li, Y. Yuan, J. Wang, and J. Cheng, “A DAPP Business Data Storage Model Based on Blockchain and IPFS,” in *Artificial Intelligence and Security*, Cham, 2021, pp. 219–230. doi: 10.1007/978-3-030-78612-0_18.
[76] D. Mohan, L. Alwin, P. Neeraja, K. D. Lawrence, and V. Pathari, “A private Ethereum blockchain implementation for secure data handling in Internet of Medical Things,” *J. Reliab. Intell. Environ.*, Aug. 2021, doi: 10.1007/s40860-021-00153-2.
[77] P. Praitheeshan, L. Pan, and R. Doss, “Private and Trustworthy Distributed Lending Model Using Hyperledger Besu,” *SN Comput. Sci.*, vol. 2, no. 2, p. 115, Feb. 2021, doi: 10.1007/s42979-021-00500-3.
[78] X. Zhang, “The use of ethereum blockchain using internet of things technology in information and fund management of financial poverty alleviation system,” *Int. J. Syst. Assur. Eng. Manag.*, Apr. 2022, doi: 10.1007/s13198-022-01644-y.

[79] E. J. Lopes *et al.*, “Live video streaming service with pay-as-you-use model on Ethereum Blockchain and InterPlanetary file system,” *Wirel. Netw.*, vol. 28, no. 7, pp. 3111–3125, Oct. 2022, doi: 10.1007/s11276-022-03009-6.
[80] A. Sedrati, A. Ouaddah, A. Mezrioui, and B. Bellaj, “IoT-Gov: an IoT governance framework using the blockchain,” *Computing*, vol. 104, no. 10, pp. 2307–2345, Oct. 2022, doi: 10.1007/s00607-022-01086-1.
[81] S. Rafati Niya, S. Bachmann, C. Brasser, M. Bucher, N. Spielmann, and B. Stiller, “DeTi: A Decentralized Ticketing Management Platform,” *J. Netw. Syst. Manag.*, vol. 30, no. 4, p. 62, Jul. 2022, doi: 10.1007/s10922-022-09675-3.
[82] Md. A. Ahmad, P. Singh, M. Sushmitha, H. A. Sanjay, and N. Madhu, “Profit Driven Blockchain Based Platform for Land Registry,” in *Emerging Research in Computing, Information, Communication and Applications*, Singapore, 2022, pp. 911–922. doi: 10.1007/978-981-16-1342-5_72.
[83] A. Arora, Kanisk, and S. Kumar, “Smart Contracts and NFTs: Non-Fungible Tokens as a Core Component of Blockchain to Be Used as Collectibles,” in *Cyber Security and Digital Forensics*, Singapore, 2022, pp. 401–422. doi: 10.1007/978-981-16-3961-6_34.
[84] E. Team, “NFTs and web3. The complete guide to NFTs & Web3,” *All NFT Space*, Jan. 24, 2022. https://allnftspace.com/2022/01/24/nfts-and-web3-the-complete-guide-to-nfts-web3/ (accessed Oct. 06, 2022).

Appendix A: Observed technologies used in each study.

Ref.	Solution	Technology	Smart Contract	Development Environment	Testnet	Frontend	API	Storage	Smart contract	NFT
IEEE Explore										
[29]	Framework for deploying mobile apps	Ethereum	Solidity	Truffle, Ganache	Metamask (Ropsten, Rinkeby)	Vue.js	web3.js, graphql	IPFS	✓	/
[30]	Voting app	Ethereum	Solidity	Truffle, Ganache	/	ASP.net	/	/	✓	/
[34]	Comparison between full and partial dapp	Ethereum	Solidity	Truffle, Ganache	Metamask (Ropsten)		web3.js	IPFS	✓	/
[35]	Healthcare system	Ethereum	Solidity	Truffle	/	React	web3.js	/	✓	/
[38]	Messaging dapp	Ethereum	Solidity	Geth	/	React	web3.js, Whisper	/	✓	/
[39]	TogEther - Crowd funding dapp	Ethereum	Solidity	Truffle	/	React	web3.js, Infura	IPFS	✓	/

[40]	SmartDHX - Diffie-Hellman key exchange with smart contracts	Ethereum	Solidity	Truffle	/	Javascript	web3.js	/	✓	/
[41]	Student credential sharing dapp	Ethereum	Solidity	/	Metamask (Rinkeby)	Next.js	web3.js	IPFS	✓	/
[36]	AI supported health record system dapp	Ethereum	Solidity	Ganache	Metamask	React	web3.js	IPFS	✓	/
[43]	Auction system framework	Hyperledger Fabric	Javascript	Docker CouchDB	/	Vue.js	Hyperledger Fabric Node SDK	/	✓	/
[31]	Voting dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	/	/	/	✓	/
[49]	Disaster managing dapp	Hyperledger Fabric	Golang	Docker Swarm	/	Javascript	CURL requests	Swarm	✓	/

[32]	RectorDApp - Rector voting dapp	Ethereum	Solidity	Truffle	Metamask	/	/	/	✓	/
[44]	Decentralization of existing centralized marketplace	Ethereum	Solidity	Truffle, Ganache	Metamask (Kovan)	React	web3.js, Infura	IPFS	✓	/
[50]	Dapp for uploading and verifying documents	Ethereum	Solidity	Truffle	Metamask (Ropsten)	React	web3.js, Infura	/	✓	/
[53]	Dapp for tracking sustainability indicators	Hyperledger Fabric	Golang	Docker Fablo	/	Vue.js	Fablo REST API	/	✓	/
[58]	Connecting Robots with blockchain	Ethereum	Solidity	Ganache	/	Python	web3.py	Swarm	✓	/
[59]	Lottery dapp	Ethereum	Solidity	/	Metamask (Binancechain)	Vue.js	web3.js	/	✓	/

[60]	Dapp for Internet of vehicles (IoV)	Ethereum	Solidity	Truffle, Ganache	Metamask	/	/	/	✓	/
[54]	PartChain - traceability for supply chain	Hyperledger Fabric	/	Docker CouchDB		Angular	Hyperledger Composer REST API	/	✓	/
[61]	BlockTracer - Covid contact tracing mobile dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	React Native	web3.js	/	✓	/
[55]	Teak traceability platform	Ethereum	Solidity	Truffle, Ganache	Metamask	Javascript	web3.js	/	✓	/
[63]	AppXchain for cross-chain interoperability	Ethereum	Solidity	Truffle, Ganache	/	/	/	IPFS	✓	/
ScienceDirect										
[64]	Dapp for public participatory geographical information systems	Ethereum	Solidity	Truffle, Ganache	/	Javascript	web3.js	/	✓	/

[51]	EtherTwin - Information managment	Ethereum	Solidity	Parity Ethereum	/	Vue.js	web3.js	/	✓	/
[42]	Dapp for sharing students' credentials	Ethereum	Solidity	/	Metamask (Rinkeby)	Next.js	web3.js	IPFS	✓	/
[52]	Privacy manager dapp	Ethereum	Solidity	Truffle, Ganache	Metamask (Ropsten)	Javascript	Javascript	/	✓	/
[65]	Dapp for buying lootboxes	Ethereum	Solidity	/	Ropsten	Javascript	web3.js	/	✓	/
[66]	Crowdsensing dapp	Ethereum	Solidity	Truffle	Metamask (Ropsten)	Javascript	Infura	IPFS	✓	/
[37]	Smart Healthcare Management system	Ethereum	Solidity	Truffle, Ganache	Metamask	Python	web3.py	/	✓	/
[69]	Cooperative energy sharing dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	Ganache GUI	web3.js	/	✓	/

[62]	Covid19 vaccine waste reduction dapp	Ethereum	Solidity	Remix VM	Metamask (Kovan)		Infura	IPFS	✓	/
[70]	SMTSEC - Payment security system for constructing sector	Ethereum	Solidity	Ganache	/	Javascript	web3.js	/	✓	/
[56]	SecureRx - Electronic prescription tracking	Ethereum	Solidity	/	Metamask (Ropsten)	React	web3.js	/	✓	/
[71]	LUCE - Data sharing platform	Ethereum	Solidity	Geth	Rinkeby	Django	web3.py	/	✓	/
[33]	DVTChain - Digital voting system dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	React	web3.js	/	✓	/
SpringerLink										

[76]	Private blockchain implementation	Ethereum	Solidity	Truffle, Geth	Metamask (Geth local)	Javascript	web3.js	/	✓	/
[45]	Marketplace for cybersecurity threat intelligence	Ethereum	Solidity	Truffle, Ganache	Metamask (Ropsten)	/	/	/	✓	/
[46]	DEALER marketplace	EOS blockchain	C++	/	EOS Kylin testnet	Javascript	/	IPFS	✓	/
[47]	IoT data marketplace	Ethereum	Solidity	Truffle	Metamask	Angular	/	IPFS	✓	/
[57]	Medical supply chain platform	Ethereum	Solidity	Truffle, Ganache	Local, Kovan	Javascript	web3.js	/	✓	/
[77]	Loan transaction processing system	Ethereum	Solidity	Hyperledger Besu	Metamask	Django	web3.js	/	✓	/
[78]	Poverty alleviation system	Ethereum	Solidity	Truffle, Geth	/	Javascript	web3.js, Infura	IPFS	✓	/

[79]	NiftySubs - Live streaming service	Ethereum	Solidity	/	Metamask (Rinkeby)	/	Textile ThreadDB (IPFS), libp2p (chat), Superfluid, Unlock, The Graph, Voodfy	IPFS	✓	✓
[48]	Marketplace for computing resources	Ethereum	Solidity	Truffle, Geth	/	Javascript	JSON-RCP	/	✓	/
[80]	IoT-Gov - IoT governance framework	Ethereum	Solidity	Geth	Metamask	Javascript	web3.js	/	✓	/
[81]	DeTi - ticketing management platform	Ethereum	Solidity	/	/	Vue.js	web3.js	IPFS	✓	/
[82]	Land Registry process dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	Angular	web3.js	/	✓	/

[67]	Crowdsensing dapp	Ethereum	Solidity	/	CrowdBC	Javascript	web3.js	/	✓	/
[72]	DeCStor - Sharing files dapp	Ethereum	Solidity	/	Ethereum Mainnet	Javascript	web3.js	Swarm	✓	/
[73]	Image sharing dapp	Ethereum	Solidity	Truffle, Ganache	Metamask	React	web3.js	IPFS	✓	/
[83]	Prototype of interaction ERC-721 token with dapp	Ethereum	Solidity	Ganache	Metamask	React	OpenZeppelin	/	✓	✓
[74]	Remain - Content sharing and data repository system	Ethereum	Solidity	Ganache	Metamask	React	web3.js	IPFS	✓	/
[68]	Orthos - Data aquisition framework	Ethereum	Solidity	/	Rinkeby	Android	Infura, web3j	/	✓	/
[75]	Data storage dapp	Ethereum	Solidity	Local test network	/	Android	Infura, web3j	IPFS	✓	/

Appendix B: Smart Contract WeddingFund.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.9;

contract WeddingFund {
    // event to emit when a memo is created
    event NewMemo(
        address indexed from,
        uint256 timestamp,
        string name,
        string message,
        string ipfsPath
    );

    //memo struct
    struct Memo {
        address from;
        uint256 timestamp;
        string name;
        string message;
        string ipfsPath;
    }

    //list of all memos received
    Memo[] memos;

    // address of contract deployer
    address payable owner;

    // constructor logic on deploy
    constructor(){
        owner = payable(msg.sender);
    }

    /**
     * @dev pay donation for contract owner
     * @param _name name of the donor
     * @param _message a message from the donor
```

```
     * @param _ipfsPath ipfs file path address
     */
    function payWeddingDonation(string memory _name,
    string memory _message, string memory _ipfsPath) public payable{
        // donation must be positive value
        require(msg.value > 0, "Can't donate with 0 ETH");
```

```
        // add memo to storage
        memos.push(Memo(
            msg.sender,
            block.timestamp,
            _name,
            _message,
            _ipfsPath
        ));

        // emit a log event when a new memo is created
        emit NewMemo(
            msg.sender,
            block.timestamp,
            _name,
            _message,
            _ipfsPath
        );
    }

    /**
     * @dev send entire stored balance in this contract
     *   to the contract owner
     */
    function withdrawFund() public {
        require(owner.send(address(this).balance));
    }

    /**
```

```
     * @dev retrieve all memos stored on the blockchain
     */
    function getMemos() public view returns(Memo[] memory) {
        return memos;
    }
}
```

Appendix C: Script to deploy smart contract.

```
const {ethers} = require("hardhat");

async function main(){
  // get the contract to deploy and deploy it
  const WeddingFund = await ethers.getContractFactory("WeddingFund");
  const weddingFund = await WeddingFund.deploy();
  await weddingFund.deployed();
  console.log("WeddingFund deployed to ", weddingFund.address);

}

main().catch((error) => {
    console.error(error);
    process.exitCode = 1;
});
```

Appendix D: Script for testing smart contract.

```
const {ethers} = require("hardhat");

// return balance of a given address
async function getBalance(address){
  const balanceBigInt = await ethers.provider.getBalance(address);
  return ethers.utils.formatEther(balanceBigInt);
}

// console.log ether balances from a list of addresses
async function printBalances(addresses){
  let idx = 0;
  for(const address of addresses){
    console.log(`Address ${idx} balance: `, await getBalance(address));
    idx++;
  }
}

// console.log memos stored on-chain from donations
async function printMemos(memos){
  for(const memo of memos){
    const timestamp = memo.timestamp;
    const supporter = memo.name;
    const supporterAddress = memo.from;
    const message = memo.message;
    const ipfsPath = memo.ipfsPath;
    console.log(`At ${timestamp}, ${supporter} (${supporterAddress})
    said: "${message}" + hash: ${ipfsPath}`);
  }
}


async function main() {
  // get example accounts
  const [owner, account, account2, account3] = await ethers.getSigners();

  // get the contract to deploy and deploy it
```

```
const WeddingFund = await ethers.getContractFactory("WeddingFund");
const weddingFund = await WeddingFund.deploy();
await weddingFund.deployed();
console.log("WeddingFund deployed to ", weddingFund.address);
```

```
// check balances before the donation payment
const addresses = [owner.address, account.address,
  account2.address,account3.address, weddingFund.address];
console.log("--- start ---")
await printBalances(addresses);

// pay few donations for the owner
const payment = {value: ethers.utils.parseEther("1")};
await weddingFund.connect(account)
.payWeddingDonation("Sarah","Loved your dress!","hash1", payment);
await weddingFund.connect(account2)
.payWeddingDonation("Samantha","Enjoy the honeymoon!","hash2", payment);
await weddingFund.connect(account3)
.payWeddingDonation("George","Have fun. :D","hash3", payment);

// check all balances after paying donations
console.log("--- after payments ---")
await printBalances(addresses);

// withdraw funds
await weddingFund.connect(owner).withdrawDonations();

// check all balances after withdraw
console.log("--- after withdrawal ---")
await printBalances(addresses);

// print all received memos
```

```
  console.log("--- memos ---")
  const memos = await weddingFund.getMemos();
  printMemos(memos);
  }

// Recommended pattern to be able to use async/await everywhere
// and properly handle errors.
main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Appendix E: Hardhat.config.js configuration file.

```
require("@nomicfoundation/hardhat-toolbox");
require("@nomiclabs/hardhat-ethers");
require("dotenv").config()

const GOERLI_URL = process.env.GOERLI_URL;
const PRIVATE_KEY = process.env.PRIVATE_KEY;

/** @type import('hardhat/config').HardhatUserConfig */
module.exports = {
  solidity: "0.8.17",
  networks:{
    goerli:{
      url: GOERLI_URL,
      accounts: [PRIVATE_KEY]
    }
  }
};
```

Appendix F: WeddingFund smart contract ABI.


```
{
  "_format": "hh-sol-artifact-1",
  "contractName": "WeddingFund",
  "sourceName": "contracts/WeddingFund.sol",
  "abi": [
    {
      "inputs": [],
      "stateMutability": "nonpayable",
      "type": "constructor"
    },
    {
      "anonymous": false,
      "inputs": [
        {
          "indexed": true,
          "internalType": "address",
          "name": "from",
          "type": "address"
        },
        {
          "indexed": false,
          "internalType": "uint256",
          "name": "timestamp",
          "type": "uint256"
        },
        {
          "indexed": false,
          "internalType": "string",
          "name": "name",
          "type": "string"
        },
        {
          "indexed": false,
          "internalType": "string",
          "name": "message",
          "type": "string"
```

```
    },
    {
      "indexed": false,
      "internalType": "string",
      "name": "ipfsPath",
      "type": "string"
    }
  ],
```

```
  "name": "NewMemo",
  "type": "event"
},
{
  "inputs": [],
  "name": "getMemos",
  "outputs": [
    {
      "components": [
        {
          "internalType": "address",
          "name": "from",
          "type": "address"
        },
        {
          "internalType": "uint256",
          "name": "timestamp",
          "type": "uint256"
        },
        {
          "internalType": "string",
          "name": "name",
          "type": "string"
        },
        {
          "internalType": "string",
          "name": "message",
```

```
            "type": "string"
          },
          {
            "internalType": "string",
            "name": "ipfsPath",
            "type": "string"
          }
        ],
        "internalType": "struct WeddingFund.Memo[]",
        "name": "",
        "type": "tuple[]"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
```

```
{
    "inputs": [
      {
        "internalType": "string",
        "name": "_name",
        "type": "string"
      },
      {
        "internalType": "string",
        "name": "_message",
        "type": "string"
      },
      {
        "internalType": "string",
        "name": "_ipfsPath",
        "type": "string"
```

```
      }
     ],
     "name": "payWeddingDonation",
     "outputs": [],
     "stateMutability": "payable",
     "type": "function"
    },
    {
     "inputs": [],
     "name": "withdrawDonations",
     "outputs": [],
     "stateMutability": "nonpayable",
     "type": "function"
    }
  ],
  "bytecode": "0x60806.....5234",
  "deployedBytecode": "0x6080.....6040",
  "linkReferences": {},
  "deployedLinkReferences": {}
}
```

Appendix G: Package.json file.

```
{
  "name": "support-this-project",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "bootstrap": "^5.2.2",
    "ethers": "^5.7.1",
    "ipfs-http-client": "^58.0.1",
    "next": "12.3.1",
    "react": "18.2.0",
    "react-bootstrap": "^2.5.0",
    "react-dom": "18.2.0"
  },
  "devDependencies": {
    "eslint": "8.24.0",
    "eslint-config-next": "12.3.1"
  }
}
```

Appendix H: Wallet connection logic.

```
// Wallet connection logic
 const isWalletConnected = async () => {
   try {
     const { ethereum } = window;

     const accounts = await ethereum.request({method: 'eth_accounts'})
     console.log("accounts: ", accounts);

     if (accounts.length > 0) {
       const account = accounts[0];
       console.log("wallet is connected! " + account);
     } else {
       console.log("make sure MetaMask is connected");
     }
   } catch (error) {
     console.log("error: ", error);
   }
 }

 const connectWallet = async () => {
   try {
     const {ethereum} = window;

     if (!ethereum) {
       console.log("please install MetaMask");
     }

     const accounts = await ethereum.request({
       method: 'eth_requestAccounts'
     });

     setCurrentAccount(accounts[0]);
   } catch (error) {
     console.log(error);
   }
 }
```

Appendix I: Frontend payWeddingDonation function implementation.

```
const payWeddingDonation = async () => {
    try {
      const {ethereum} = window;
      let ipfsPathResult = "";
      if(ethereum){
        if(amount>0){
        const provider = new ethers.providers.Web3Provider(ethereum, "any");
        const signer = provider.getSigner();
        const weddingFund = new ethers.Contract(
          contractAddress,
          contractAbi,
          signer
        );

        // logic for ipfs
          try {
            let ipfs = await ipfsClient();

            let options = {
              warpWithDirectory: false,
              progress: (prog) => console.log(`Saved ${prog}`)
            }

            let result = await ipfs.add(selectedFile,options);

            ipfsPathResult=result.path;

          } catch (error) {
            console.log(error);
            return;
          }

        // pay donation loggic
        const donationTxn = await weddingFund.payWeddingDonation(
```

```
        name ? name : "anonymous",
        message ? message : "Enjoy this donation!",
        ipfsPathResult ? ipfsPathResult : "error",
        {value: ethers.utils.parseEther(amount+"")}
      );
```

```
      await donationTxn.wait();

      // clear form
      setName("");
      setMessage("");
      setIpfsHash("");
      setSelectedFile("");
      setIsSelected(false);

    }
  }
  } catch (error) {
    console.log(error);
  }
}
```

Appendix J: Frontend withdrawFunds function implementation.

```
const withdrawFunds = async () => {

  try {
    const {ethereum} = window;
    if(ethereum){

      const provider = new ethers.providers.Web3Provider(ethereum, "any");
      const signer = provider.getSigner();
      const weddingFund = new ethers.Contract(
        contractAddress,
        contractAbi,
        signer
      );

      const contractBalance = await getBalance(provider, weddingFund.address);
       console.log("Current balance of contract: ", await getBalance(provider, weddingFund.address),
"ETH");

      // Withdraw funds if there are funds to withdraw.
      if (contractBalance !== "0.0") {
          console.log("withdrawing funds..")
          const withdrawTxn = await weddingFund.withdrawDonations();
          await withdrawTxn.wait();
      } else {
          console.log("No funds to withdraw!");
      }
      console.log("Funds withdrawn!");


  }
  } catch (error) {
    console.log(error);
  }
 }
```

Appendix K: Frontend getMemos function implementation.

```
// Function to fetch all memos stored on-chain.
 const getMemos = async () => {
   try {
     const { ethereum } = window;
     if (ethereum) {
       const provider = new ethers.providers.Web3Provider(ethereum);
       const signer = provider.getSigner();
       const weddingFund = new ethers.Contract(
         contractAddress,
         contractAbi,
         signer
       );

       console.log("fetching memos from the blockchain..");
       const memos = await weddingFund.getMemos();
       console.log("fetched!");
       setMemos(memos);
     } else {
       console.log("Metamask is not connected");
     }
   } catch (error) {
     console.log(error);
   }
 };
```

Appendix L: Frontend IPFS client configuration.

```
const auth =
    'Basic ' + Buffer.from(projectId + ':' + projectSecret).toString('base64');


  async function ipfsClient (){
    const ipfs = await create({
      host: 'ipfs.infura.io',
      port: 5001,
      protocol: 'https',
      headers: {
          authorization: auth,
      },
    });
    return ipfs;
  }
```