SELF-SOVEREIGN IDENTITY (SSI) MANAGEMENT AND ACCESS CONTROL

by

NIKOLAOS – TSAMPIKOS PAPATHEODOROU

Bachelor of Science in Informatics Engineering (T.E.), Hellenic Mediterranean, 2019

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2021

Approved by:

Major Professor
Nikolaos Papadakis

Abstract

In the analog world, anyone who wants to prove or confirm their identity usually pulls out their ID card. When going to the authorities, visiting the bank, or checking into a hotel, we can confirm our identity this way. After presentation, the ID card usually goes back into the wallet, which means we retain control over our data. A fundamental change in the current digital world's trust structures is in progress, moving from the centralized trust models, such as the one used in the TLS X.509 PKI, to more distributed and decentralized ones. At the same time, it establishes the way for moving from the present world, where we all are forced to rely on Apple, Microsoft, Google, and the other digital giants for all our digital life, to a world where we communally manage digital identities and trust in the tools and devices, we use for representing people, communities, organizations, and their assets. Furthermore, from the European point of view, we enable the verification of identities, when needed, to the European electronic identities (eID) and the eIDAS regulation, tools, and mechanisms, as well as the International Civil Aviation Organization (ICAO) endorsed "ePassport" or electronically machine-readable travel documents (eMRTD) infrastructure. This thesis investigates existing identification solutions, blockchain innovation, and decentralized frameworks. At that point, it provides a solution based on Binance Chain smart contracts that incorporate cryptographic security, operational simplicity, and information independence. Finally, it is evaluated using a set of plan objectives and extensive security considerations. We will represent the identity of persons and objects with standardized W3C DIDs, allowing us to simultaneously keep the identities private and prove that the acting parties have been granted sufficient privileges or provably possess other attributes requested so the operation can justifiably be conducted. Furthermore, our goal is to make technologies compatible with European eIDAS regulations. However, a trustworthy and reliable system is needed so that individuals, companies, and machines can interact and network digitally and securely.

Table of Contents

Abstract 2
Table of Contents 3
List of Figures 6
List of Tables 8
Acknowledgments 9
Chapter 1 10
Introduction 10
Chapter 2 13
Background 13
Web Applications 13
Blockchain 13
Smart Contract 14
Decentralized Application 16
Related Work 17
Chapter 3 21
Design 21
Saving data into Blockchain 21
Use Case Model 22
Use Case Essential 23
Use Case Elaborated 24
Admin Flow Chart: 25
User Flow Chart: 26
Sequence Diagram 27
Activity Diagram 28
Admin Activity Diagram 29
Component Diagram 30
Deployment Diagram 31
Supplementary Specification 32

Software Quality Attributes 33
Chapter 4 34
Software & Experimental Framework 34
Chapter 5 43
Troubleshooting Guide 43
Roles 43
Why Role Assigned in Constructor? 44
Client Website Landing Page 50
Government Website Landing Page 51
Client Website Dashboard 52
Login Page For Client Website 53
Sign-Up Page 53
Client-Side Dashboard 55
Upload Data to IPFS 55
Sent Requests Page 56
Govt Dashboard To Verify Data 57
Grant Access Page 58
Experimental Evaluation 59
Black Box Testing 59
White Box Testing 63
State of the art comparison 65
Chapter 6 67
Conclusion & Future Work 67
Appendix A 68
WBS (Work Breakdown Structure) 68
Appendix B 69
PERT Chart 69
Appendix C 69
Critical Path 69
Appendix D 70
Blockchains & Their Approach 70

Appendix E 71
Roles & Responsibility 71
Responsibilities 72
Milestones & Deliverables 72
Potential Risk & Issues 73
References 74

List of Figures

Figure 1: Common Web Application....13
Figure 2: Blockchain Network....14
Figure 3: Decentralized Web Application....17
Figure 4: Blockchain Storage Explanation....22
Figure 5: Use Case Diagram with respect to User....22
Figure 6: Admin Flow Chart....25
Figure 7: User Flow Chart....26
Figure 8: Sequence Diagram....27
Figure 9: Activity Diagram....28
Figure 10: Admin Activity Diagram....29
Figure 11: UML Component Diagram....30
Figure 12: Deployment Diagram....31
Figure 13: Solidity....37
Figure 14: MetaMask....38
Figure 15: Ganache....39
Figure 16: IPFS Structure....41
Figure 17: Landing Page....50
Figure 18: Government Website Landing Page....51
Figure 19: Client Dashboard....52
Figure 20: Login Page....53
Figure 21: Sign Up....54
Figure 22: Client Dashboard....55
Figure 23: Upload IPFS....56
Figure 24: Sent Request....56
Figure 25: Govt. Verify Data....57
Figure 26: Govt Verified....57
Figure 27: Govt Req....58
Figure 28: Grant Access....58

Figure 29: Truffle Flow.....63
Figure 30: WBS 68
Figure 31: Pert 69
Figure 32: Blockchains Approach 70

List of Tables

Table 1: Correlation Between uPort & Sovrin.....18
Table 2: Principles of SSI.....19
Table 3: Essential Use Cases.....23
Table 4: Elaborated Use Cases.....24
Table 5: Supplementary Specification.....32
Table 6: Test 1.....60
Table 7: Test 2.....61
Table 8: Test 3.....62
Table 9: Comparison of BSC & Ethereum Network.....66

Acknowledgments

I want to express my appreciation and gratitude to Dr. Nikolaos Papadakis for his assistance and encouragement to complete my thesis. My gratitude goes to professor Hatzivasilis Georgios, my colleague Koumarelis Anastasios, and my friends and family to support my work. I want to thank Committee Chair George Hatzivasilis and Committee member Haridimos Kondylakis.

Chapter 1

Introduction

The global identity verification market is growing fast; it was valued at about six billion dollars in 2019. Before COVID-19, the size was expected to more than double to twelve point eight billion by 2024 [1]. The factors driving this are the industry (including the cases in which it is relevant), the advancement of technology and customer demand, governance, and government.

A Self Sovereign Identity is an Identity that is not dependent on any central authority and cannot easily be removed by one administration but is an entirely portable digital identity [2]. It typically has six variables, an identity model in which its data is owned or controlled by any individual organization or thing; the centralized authority does not control it. The owner of the identity cannot remove it. Identity owners retain all the personal information stored in their accounts. The system does not require an intrinsic authority to trust the nature of the integrity, protection, privacy, and identity of the owners; identity owners may use their data, select who may access and manage their identity online. Modifications are easy to make to the question and/or answer. The transparency is preserved by rapidly providing six sections of the anchor, credibility, entity, self-limit, ZKP, and ZKP itself.

The implementation of blockchain-based identity management models has extended the concept of a self-sovereign identity paradigm to include more than just cyberspace. With SSI, people can take back control of their electronically distributed data whenever they choose. SSI implies that full user control will take decentralized identity a step further by giving users control of their identifiers and data [3]. Organizational identity management such as Facebook can now make profits of eighteen point five billion, and Google can make profits of tens of billions of dollars from this. Based on this organizational identity management, companies valued at seven hundred billion dollars can also be worth up to a trillion dollars. While someone is in control of Self Sovereign Identity is the individual, e.g., Bob, Alice can give some things to the press, his employer, his health care, his security company, and all that kind of stuff. Some can give them whatever data they want, and they do not give them what they do not want. Identity management methods have often changed over time, based on how much privacy they have maintained. Over time, there have been many identity management methods in which their confidentiality has increased over their evolution; for example, centralized, federated, self-sovereign, and then blockchain-based self-sovereign. The applications of algorithms such as public-key encryption

and hash functions can be used for digital identity control. It can be used to assure identity-based records' authenticity, attest to their records' integrity, and share records with smart contracts.

The long-term goals of self-sovereign identity are:

1. Decentralization allows entities to manage themselves and eliminates unnecessary targets or single points of failure. It will enable you to control your digital identifiers without relying on external authorities to access your data and sell it to others.
2. Privacy enhances the security of data.
3. Security enables a necessary degree of trust in DID documents.
4. Proof-of-stake allows DIDs to provide identity and information integrity when communicating.
5. Discoverability increases entities' ability to find their self-identity.
6. Interoperability allows existing tools and software to be used.
7. The portability system is system-agnostic, independent, and able to support all types of DIDs and DID methods.
8. Simplicity means it is easy to understand, implement, deploy, and maintain the technology.

Credentials, including verification information, are securely stored and are readable by computers. This system offers holders of digital identities the opportunity to maintain their own identities. Documentation requires attestation from another party, but it is attested by the self as well. Documents are signed using digital certificates. The signature provides proof that the identity of the cardholder is genuine. The public key is verified by confirming the private key belonging to the credential's owner. Verifiers trust the issuer; therefore, they approve the certificate.

Decentralized identity mechanisms will have a significant impact on business operations and government policymaking in the coming decade [4]. DID uses public-key cryptography to ensure its security (public and an associated private key). The key to doing well on a DID is managing your private keys. For verifiable claims, DID is enabled for digitally signed verification. An entity issues a bond to a buyer. Several elements are involved in a DID, such as

a decentralized identifier, signing key, a collection of authentication methods, a timestamp, and a signature (for integrity).

Many projects in the industry, including early ecosystems and consortiums, are useful for supporting this, so there is the decentralized identity base (DIF), Indy Project, Aries Project, and the W3C [5]. In some cases, self-sovereign identity makes sense as a more secure way to accredit transactions. Critical streamlining workflows lacking standardization of data and communication credentials can be useful in another personal context. Data is a liability and a customer-driven workflow.

Through this project, we will create a decentralized application that runs on the BSC (Binance Smart Chain) network. We are trying to demonstrate how to create and verify immutably self-sovereign identities using smart contracts [6]. We will also let our decentralized application's users upload their user data on the blockchain, and we will visualize all the data as it happens. To ensure that the smart contract is secure and legal for the blockchain, we used tests. We are creating a client-side framework that enables users to communicate with our smart contract by connecting to the network, which provides them with guidance on what to do.

Following that, Chapter 2 goes through prior work and relevant literature. Chapters 3 and 4 go into considerable detail about the design of our method, including use case, flow, and component diagrams. Chapter 4 consists of the programming languages and software that were utilized and as a result, there is the experimental framework that was achieved. Chapter 5 introduces a troubleshooting guide along with the experimental evaluation of our framework which consists of the testing of the smart. Concluding Chapter 7 discusses our conclusions and future work.

Chapter 2

This chapter reviews the history of Web Applications, particularly decentralized ones. Decentralized applications are built on Blockchain, which allows many programs to run when a rule is met. It also points to similar decentralized web applications that will be compared to our research.

Background

Web Applications

A web application is used as an analogy to illustrate what blockchain is. Usually, when someone communicates with a web application, a browser must talk to the network's central server. The app's code will be stored on one server, and this data will be held in a centralized database, as shown in *Figure 1*. Also, if anyone tries to use the application, they need to speak to the central server. It is how a web application works; it is not useful to create an SSI this way. First, since all the data in the database might change at any moment, our information could shift, or it could be entirely deleted. Second, the framework code could change at any moment, which implies that SSI principles could change. It is incorrect to construct a centralized web application because of these problems. Therefore, a decentralized application is developed to solve the issues. We want to ensure our users' integrity, privacy, and anonymity.

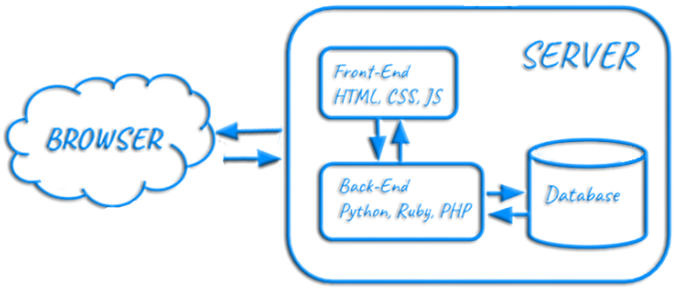


Figure 1: Common Web Application

Blockchain

Blockchain solves these issues, and it is a safer option to create an SSI identity application. All the data is not stored on the central server on the blockchain, but it is decentralized instead. It is distributed through any computer connected to the blockchain. When connected to the blockchain network, the computer acts as a node. The computer is part of the blockchain network, as shown in *Figure 2*. It communicates to all other nodes who share some of

the same responsibilities that a web server can assume. Blockchain uses a public, distributed ledger, which means storing all information within a distributed data storage network (a block). In the distributed public ledger, both the network nodes act simultaneously to ensure the security of all public ledger data. [7]. It is crucial to an SSI application because it ensures that we will always know that an account has submitted the transaction every time a user submits critical information.

It will also provide confidence that the information will be saved to the correct user and be registered forever. Since all the data is exchanged through blockchain devices, the blockchain is a database. Since all the nodes connect on the blockchain to each other, it is also a network. Instead of a conventional web model, someone might think of blockchain as a network and database all in one. Consequently, it is far easier to construct our SSI application on the blockchain in this decentralized way so that someone can use it confidently.

As we previously mentioned, all the data on the BSC (Binance Smart Chain) blockchain is decentralized and secured. Code on the blockchain is considered public and cannot be altered [8], which means that people can write code on the blockchain, and it will be immediately downloaded and executed by other nodes in the network.

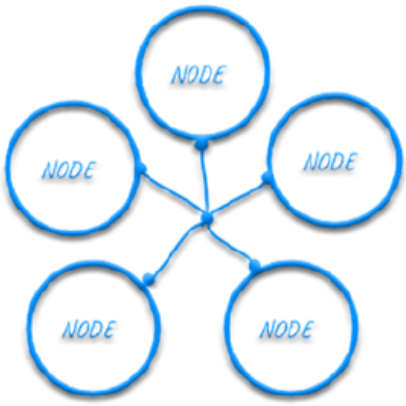


Figure 2: Blockchain Network

Smart Contract

BSC (Binance Smart Chain) allows developers to write code that will be executed on a BSC (Binance Smart Chain) virtual machine with smart contracts [9, 10]. It is where all the business logic and our application will lie. All code is written; it is responsible for the reading and writing of value transfer data and the execution of any business logic that has been

programmed. A smart contract is like a web-based microservice. The Smart Contract is a utility that controls the data stored in the blockchain. It is also a smart contract because it reflects what happened earlier as a part of the insurance application. A contract in which users can upload their files, and once they do, they must permit someone else to access it. Smart contracts which are written in the BSC (Binance Smart Chain), coded with a specific programming language known as Solidity. Solidity will be used to constructing dapps smart contracts.

These are the lines of code that are deposited on an EVM. It executes automatically when prearranged conditions are met. However, at the most rudimentary level, they are computer programs that run as they have been set up to run by the individuals who developed them. The advantage of smart contracts is most apparent in business associations [11]. In this way, they are usually used to enforce some type of agreement so that all contributors can be certain of the outcome without an intercessor's enrollment.

However, the easiest way to clarify what a smart contract ensures is through an example. If we have ever bought a car at a dealership, we know there are numerous steps, and it can be an annoying process. If we cannot pay for the car outright, we will have to get financing. This may require a credit check and we will have to fill out numerous forms with our personal information to verify our identity. Literally the way, we will have to interact with many different people, plus the salesperson, money broker, and lender. To pay for their work, several commissions are added to the actual price of the car.

It is a network of nodes or computers that executes the actions when programmed conditions have been met and confirmed, like releasing funds to the parties; registering a car; sending or receiving notifications. The blockchain is then effective once the transaction is confirmed.

For a second example, let us say Alex and Ben bet on whether it will rain or not. Alex says there will be rain tomorrow and Ben says there will not rain tomorrow. They strike a bargain that if there is rain tomorrow then Ben must pay Alex $10 and if there is no rain tomorrow then Alex must pay $10 to Ben. However, if Ben loses and he refused to pay the $10 as they both agreed, then what can they do without fighting? An alternative way can be to go to a lawyer to make their agreement legal. If one refuses to pay, he will have to go to jail or pay a fine such as an extra $10. Remember, they must also pay that lawyer and for any legal papers. This is costly.

A smart contract would be better a solution; a smart contract on BSC (Binance Smart Chain) blockchain. When certain conditions are met the smart contract will pay the money to the winner. There will be no fighting and no need to pay any third-party expenses. Everything will be reliable and accurate with less expense.

Let us take a look at the third example of rented an apartment. If the person living in an apartment does not pay before the fifth of the coming month, then he will lose access to that apartment. The smart contract will automatically revoke his access to that apartment.

Account in BSC (Binance Smart Chain)

There are two types of accounts in BSC (Binance Smart Chain)

1) Externally Owned Accounts

It has a public key and a private key. It is identified by a wallet address (what a wallet is will be discussed later) and accessed by a private key. The individuals who hold this private key can transfer sign transactions and Ether from this account. EOAs are considered user-type accounts (for users) and are linked to a cryptographic key pair created upon account creation.

However, a public key is used to reference the account and calls externally owned account addresses whereas the private key on the other side is used to sign a transaction before executing any kind of transaction on the network to prove validity. Externally owned accounts have balances that hold Ether cryptocurrency.

2) Contract Accounts

A smart contract is a kind of account that is controlled and operates by EOA. When a smart contract is deployed on BSC (Binance Smart Chain) it is executed by EVM on nodes. A smart contract is considered as an independent agent executed by the EVM BSC (Binance Smart Chain) Virtual Machine and is the central or core foundation and the main building block of any Decentralized Application. Once our code is deployed on the Blockchain the BSC (Binance Smart Chain) virtual machine will maintain this and run it when invoked by EOA.

Decentralized Application

Dapp is a distributed application; it is distributed in every way. Data sharing in the P2P network is done across devices in the network and is decentralized. A user-side application will

be developed containing a web page developed in the markup language HTML (HyperText Markup Language) and in the scripting languages CSS and JavaScript. In this case, instead of having a server operating the nodes locally, they will be connected to blockchain nodes built into a target point, [12, 13] as shown in *Figure 3*. The code to the decentralized application will be written with a smart contract. A local blockchain will be used to support a smart contract. We will allow accounts in the network to use our application and update their personal information.

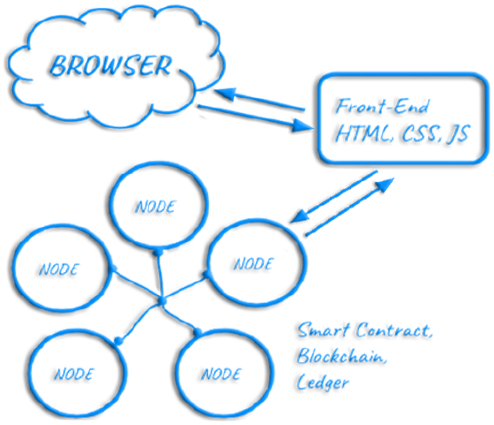


Figure 3: Decentralized Web Application

Related Work

Digital identity is an essential component of any digital platform. However, this field was one of the hardest to master to adequately establish the identity of any individual on the web, as well as in the real world. Throughout the years, many Identities Management models have been used. No model has been able to establish the exact function of personal data storage and identity management until recently. The potential effects of globalization on identity have affected security, privacy, and safeguarding [14]. Many enhancements have been brought forth to SSI, including usability, security, and privacy for everyone who uses it. Besides, it stores all confidential information in a user-owned and managed Digital Wallet. Digital wallets are like physical wallets that contain all digital credentials. However, digital credentials are signed and verified digitally more easily on the digital wallet than their physical counterparts. The framework does not include a mediator, only a person-to-person arrangement [15]. SSI is IDM 3.0. Therefore, it needs to be carefully evaluated for its various aspects. For Federated IDM 2.0, two specifications have been proposed [16, 17]. Two emerging self-sovereign identity solutions, uPort, and Sovrin have been compared and then analyzed, as shown in *Table 1*. The self-

sovereign solution named uPort has been built on the BSC (Binance Smart Chain) public blockchain without authorization [18]. Another solution called Sovrin has been built on the Hyperledger Indy permitted by the public [19].

Specifications	uPort	Sovrin
1. Sovereignty	It is a self-sovereign identity.	It is a self-sovereign identity.
2. Storage-Control	Identity and its associated personal and confidential data is stored in the Digital Wallet at the device owned and controlled by the identity owner.	Identity and its associated personal and confidential data is stored in the Edge Wallet at the device of Edge Agent controlled by the identity owner; It may be stored in the Cloud Wallet at the device of Cloud Agent (protected from unauthorized access).
3. Longevity	It utilises Decentralized Identifiers (DIDs).	It utilises Decentralized Identifiers (DIDs).
4. Verifiability	It utilises Verifiable Credentials (VCs).	It utilises Verifiable Credentials (VCs).
5. Recovery	Social Recovery Method: Recovery Delegates (e.g. selected family members, friends or institutions) nominated by an identity owner, who can assist the user to regain its uPort identity.	Social Recovery Method: Recovery Key Trustees trusted by the identity owner store recovery data on their own agents on the behalf of an identity owner and help them to recover their identity.
6. Cost-Free	Presently identity is free for users, however, all transactions have an inherent cost.	Presently identity is free for users, and no financial cost to identity transactions.
7. Security	It requires a PIN/Password and Biometry for controlling identity through blockchain. Users can securely publish their identity including transfer their credentials, sign transactions and control their keys and data.	It requires a PIN/Password and Biometry for controlling identity through blockchain. Users can securely publish their identity including transfer their credentials, sign transactions and control their keys and data using powerful cryptography.
8. Privacy	It is a Privacy Preserving. Users do not need to disclose personal data in order to create uPort identifiers for low value accounts. It uses various methods to minimize the correlation of a user's on-chain smart contract interactions between different dapps.	It is a Privacy by Design and Privacy by Default. It uses anonymous credentials based on Zero-Knowledge Proofs (ZKPs), which allows users to share the information that maintain the anonymity of users.
9. Safeguard	Users' right to privacy should be protected.	Users' right to privacy should be protected.
10. Accessibility	Simple design architecture and easy to use. At present it has no provision of a Guardian/Agent.	Complex design architecture and some users might require a Guardian to manage the identity on their behalf.
11. Availability	Users should require their smart-phone to manage their identity.	Users should require smart-phone but not necessarily its ownership.
12. Transparency	It is based on open standards and open source projects.	It is based on open standards and open source projects.
13. Portability	It is limited, however, uPort is using several open standards to make it portable, e.g., Verifiable Credential (VC) and Decentralized IDentifier (DID).	It is limited, however, Sovrin is using several open standards to make it portable, e.g., Verifiable Credential (VC) and Decentralized IDentifier (DID).
14. Interoperability	Presently it is evolving, therefore, it requires a further alignment with other SSI solutions.	Presently it is evolving, therefore, it requires a further alignment with other SSI solutions.
15. Scalability	It is limited. The public Ethereum blockchain can process nearly 15 transactions per second. It is resolving this by avoiding creation of multiple smart contracts on the blockchain and letting users to create Ethereum key pair.	It is limited. It is resolving this by using two rings of nodes: a ring of validator nodes to accept write transactions, and a much bigger ring of observer nodes to run read-only copies of the blockchain to process read requests.

Table 1: Correlation Between uPort & Sovrin

Christopher Allen suggested the Ten Principles for Self-Sovereign Identity [20], which provided an implementation framework for the concept of self-supreme identity. Furthermore, as shown in *Table 2*, the Sovrin Foundation [21] grouped in a white paper three categories: security, monitoring, and portability. However, in this sense, persistence does not interfere with Allen's

"right to be forgotten." It may as well be listed separately since control and consent must be created as well as access and removal of identity.

Security	Controllability	Portability
Protection	Existence	Interoperability
Persistance	Control	Transparency
Minimisation	Consent	Access
		Portability

Table 2: Principles of SSI

Domain Name System–Identity Management, also referred to as DNS–IdM is a scheme of Identity Management based on smart contracts. This helps users preserve their identities related to distinctive characteristics, achieving the self-sovereign concept. With respect to protection and privacy, DNS–IdM has promising results [22]. DNS–IdM can avoid traditional security threats and the drawbacks of existing decentralized identity management systems because of its decentralized existence. For identity management, this method provides a domain name system-like experience. A person can use real-world attributes of his or her identity via DNS–IdM to generate a digitized identification which is able to be used by various companies to authenticate a user and offer their services based on validated criteria. DNS–IdM varies from conventional schemes of identity management as well as the current advancements in the blockchain–based identity systems by utilizing the blockchain as a trustworthy and stable network to have identities and use smart contracts to obtain identity from their respective recipients safely.

Most conventional schemes of Identity Management primarily focus on mediators to validate statements concerning identity characteristics that, once checked, remain true even though the benefactors have revoked or modified them. However, identity systems that are based on blockchain depend entirely on the users' acknowledgment of their I.D. attributes or their approval of the details being monitored. DNS–IdM gives clients the ability to control their identification details. For suppliers of services, that would make it possible for any of them to utilize already presented identity attributes and to request them directly from beneficiaries through smart contracts when the features needed do not exist. The entire identity management process (e.g., the persistence of attributes, requests, and verification) is seamlessly enforced by integrating authorization and unauthorized blockchains with intelligent contracts. The

blockchain’s intrinsic properties, manipulating resistance, longevity, and traceability render the DNS-IdM secure and trustworthy.

Chapter 3

Design

In the Project Design Stage, we will show the details for how the project will be managed and governed. Project Design differs from the Operational Planning Stage in that things at a higher level are about managing the project.

For example:

- How will decisions be made and who has the authority?
- If the Implementation Phase must happen in August, then when does the money have to be secured to start the project?
- What staff training is required and when?

The following chapter indicates the design of our project.

Saving data into Blockchain

A blockchain gathers information in groupings, often known as blocks, that hold sets of data. When a block's storage capacity is reached, it is chained onto the previously filled block, producing a data chain known as the "blockchain" [23]. All new information that follows that newly added block is compiled into a newly formed block, which is then added to the chain once it is complete as shown in *Figure 4*. It links all blocks by storing the previous hash and timestamp, the timestamp should be greater than the previous block. The transaction root contains the hash of the merkle tree and the merkle tree has all hashed transactions in that block. This is called transaction processing, one can just check the validity of the block by just checking the header hash of the block because the header hash contains the hash of all items in that block. Some people say's, "the blockchain is a chain of header hashes, and one header hash is connected to the previous header hash".

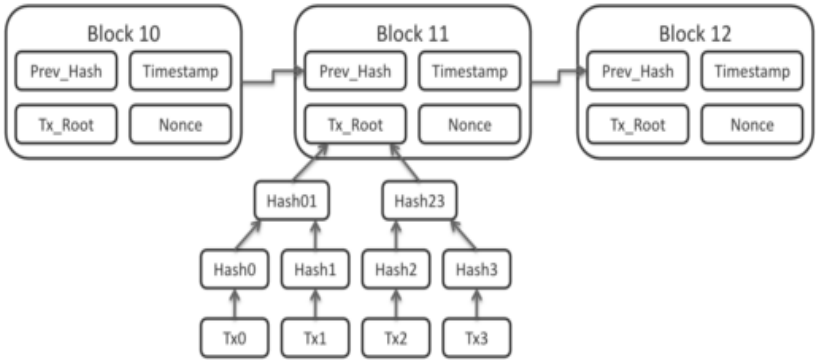


Figure 4: Blockchain Storage Explanation

Use Case Model

Use case models are models of how several types of users interrelate with the system to solve a problem. The use case model shown in Figure 5 defines the journey of the user to this web application. User first sign in or sign up with his metamask account and his address is recorded in the blockchain, he can change his account address also. Then he needs to enter all detailed information about him. There is also an option to upload a file to IPFS in which the hash of the file is internally stored in smart-contract. Now, other users can send a request to see that data by sending a request transaction from a function. Then the owner of the data will see the request and he can accept or reject that request. If the user who requested the data, gets his request approved then he can download the file containing all data for a particular user only.

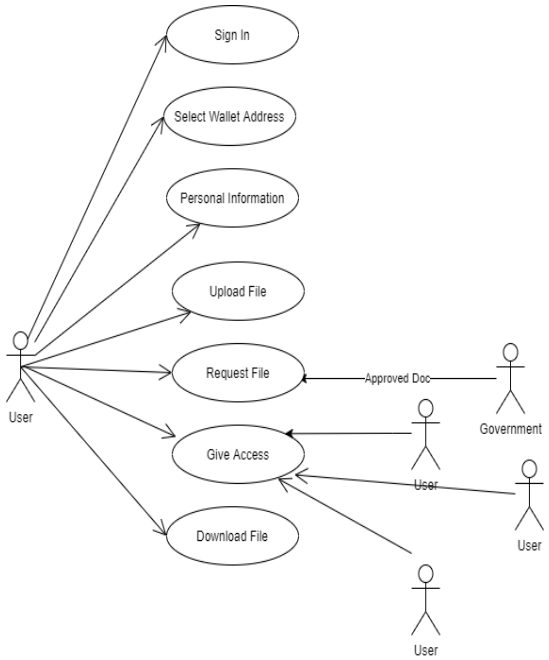


Figure 5: Use Case Diagram with respect to User

Use Case Essential

The essential use case is also known as a business use case. It is a simplified, abstract, generalized use case that captures the intentions of a user in the desired technology and performs independently.

User intention	System responsibility
Sign-up with details	Stores user's data on the blockchain as well as on IPFS.
Govt. verification	Govt. Authority will verify the user's data.
Users can request to see another user data	Will fetch other user's details and show it to a user who requested it.
Download user data	After the user's account address has granted access to see data, he can directly download files or see data in fields from the blockchain smart contract.
Grant access role	Data owners can grant access to other accounts by their addresses.

Table 3: Essential Use Cases

Use Case Elaborated

Use cases are elaborated by adding details to contact between the user and the software system as shown in Table 4. User first sign in or sign up with his metamask account and his address is recorded in the blockchain, he can change his account address also. Then he needs to enter all the detailed information about him. There is also an option to upload a file to IPFS in which the hash of the file is internally stored in smart-contract. Now, other users can send a request to see that data by sending a request transaction from a function. Then the owner of the data will see the request and he can accept or reject that request. If the user who requested the data, gets his request approved then he can download the file containing all data for a particular user only.

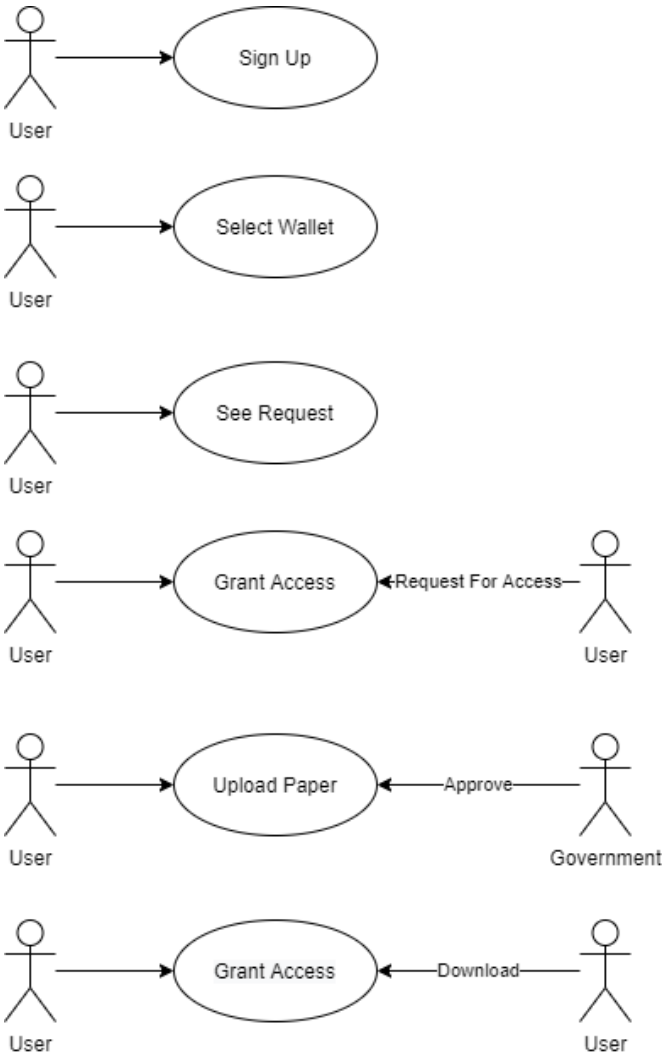


Table 4: Elaborated Use Cases

Admin Flow Chart:

The admin flow chart as pictured in Figure 6, shows that how the admin will go through the process. Admin has a dashboard and the authority to reject and discard a document. All users are also visible to the admin. As we have made two websites, one for admin and one for the user. Admin can log in to the dashboard with his metamask. He can approve or reject the data uploaded by the user. This basically is government.

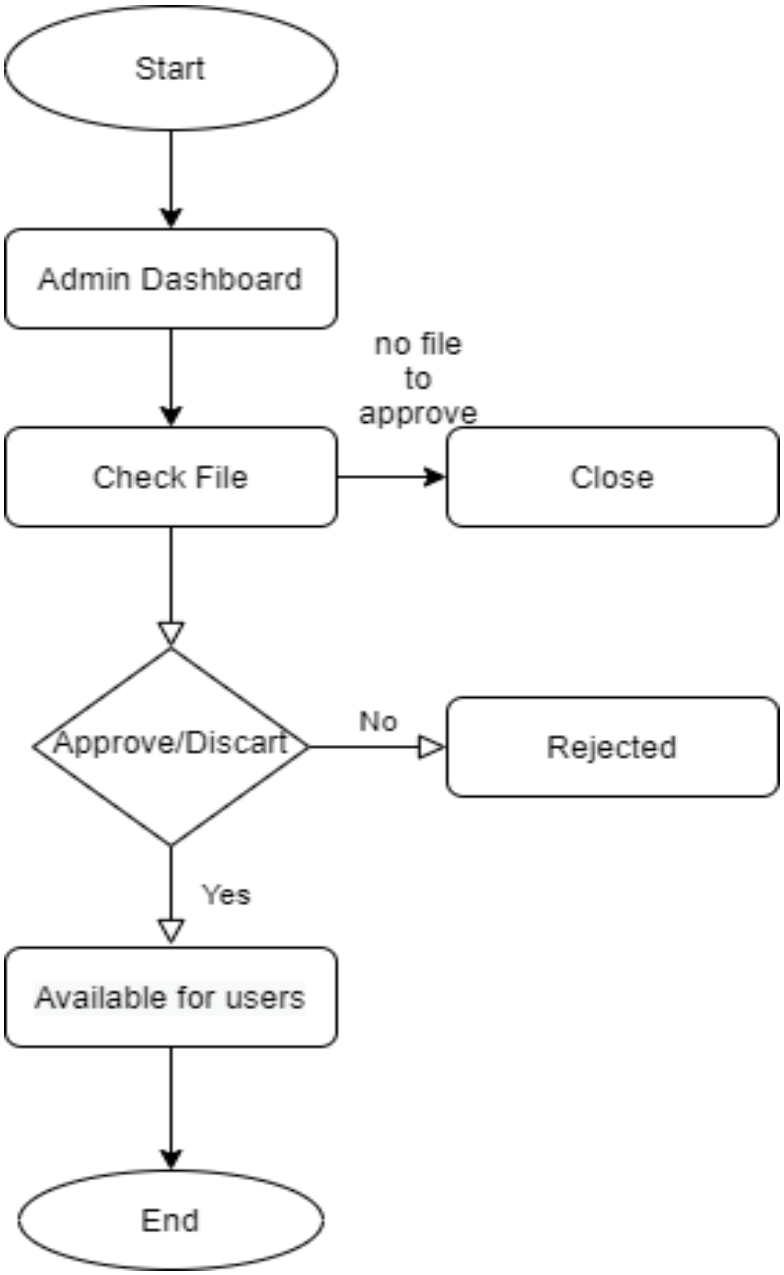


Figure 6: Admin Flow Chart

User Flow Chart:

Figure 7 shows the workflow of the user of SSI. The individual will have to connect their wallet, sign-up, upload documents, look for approval, and also can download other individual's documents. The user first needs to install metamask extension and have to make a new account. Then he can select an account. He can put his information then upload that file.

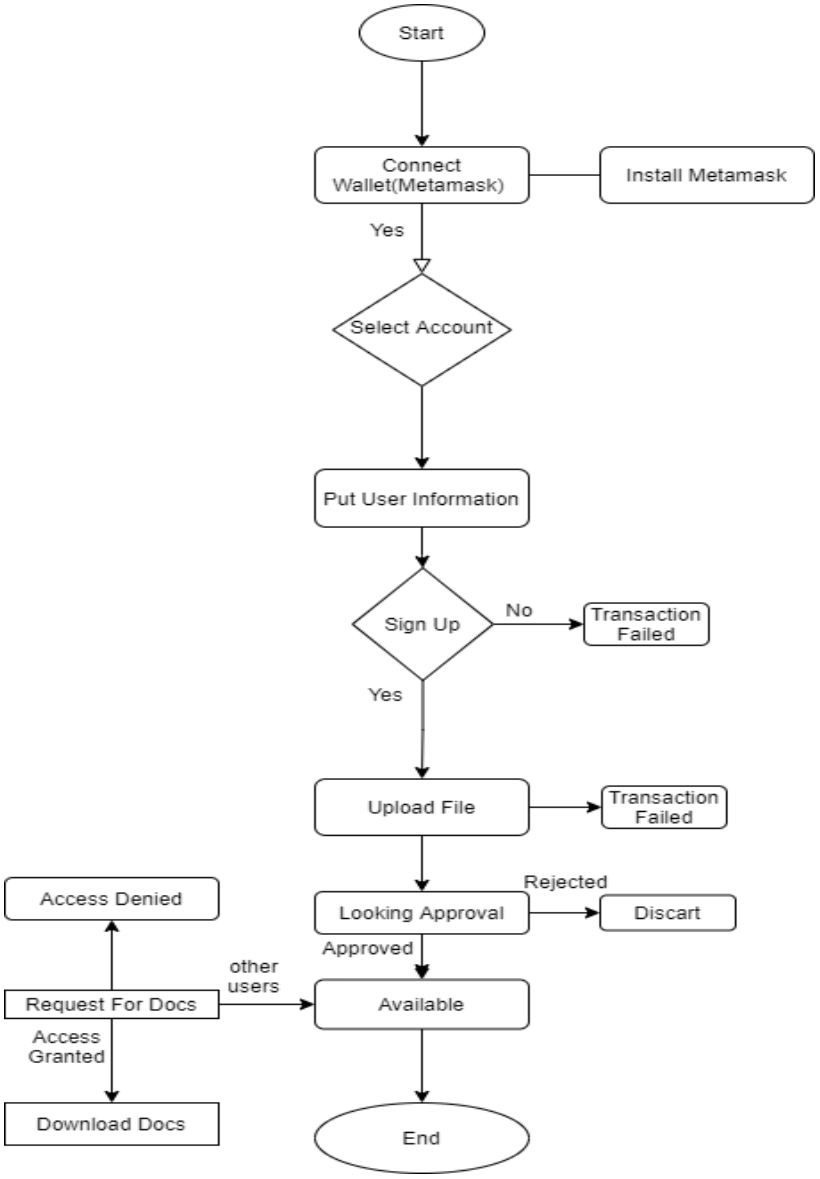


Figure 7: User Flow Chart

Sequence Diagram

A sequence diagram as shown in Figure 8 is a type of communication diagram because it describes how and in what respective order objects work together. The diagram illustrated in Figure 8 is used by software development professionals to understand the requirements for a new system or to document an existing process.

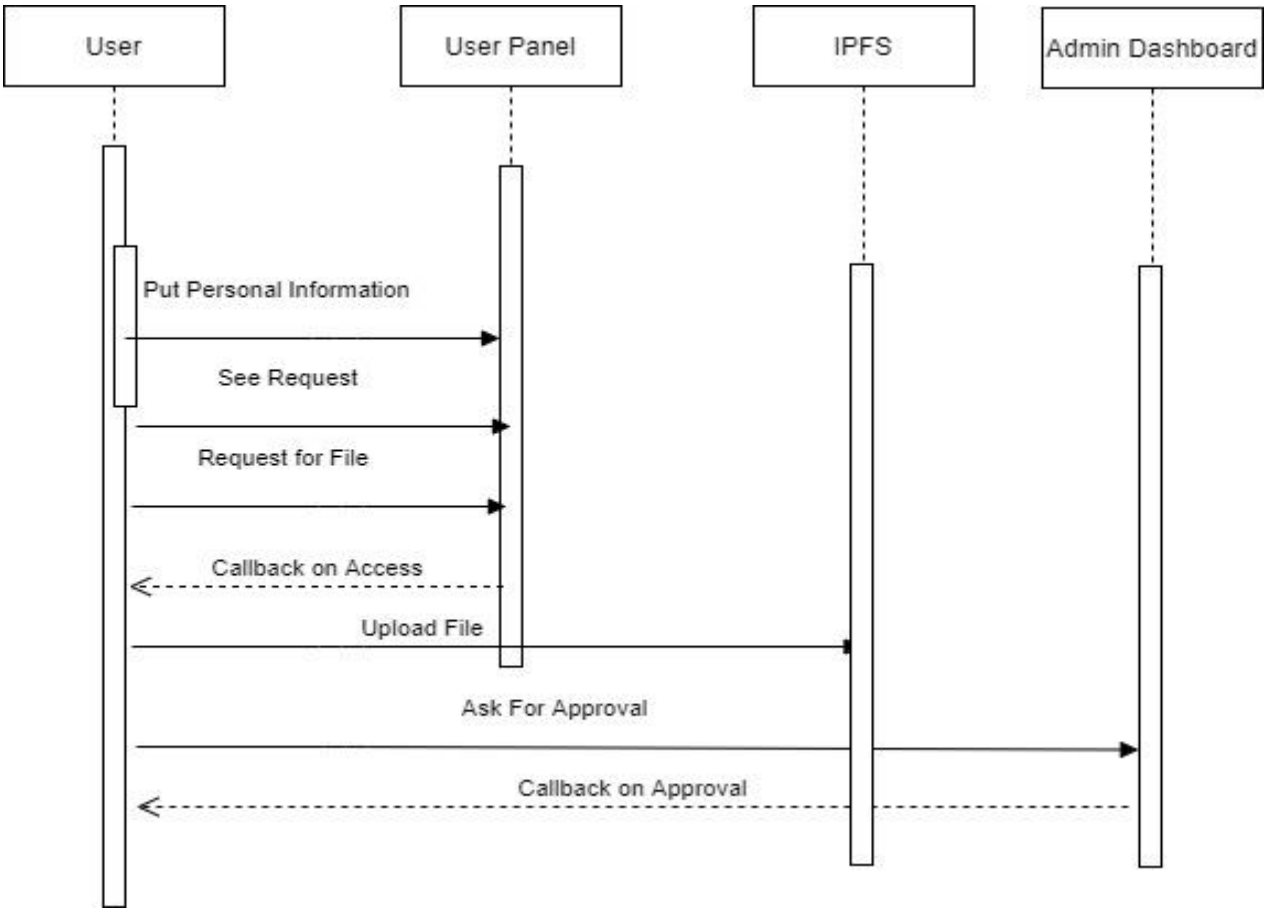


Figure 8: Sequence Diagram

Activity Diagram

An activity diagram as shown in Figure 9 visually grants a series of activities or represents the flow of control in a system like a flowchart or any data flow diagram which have been described above.

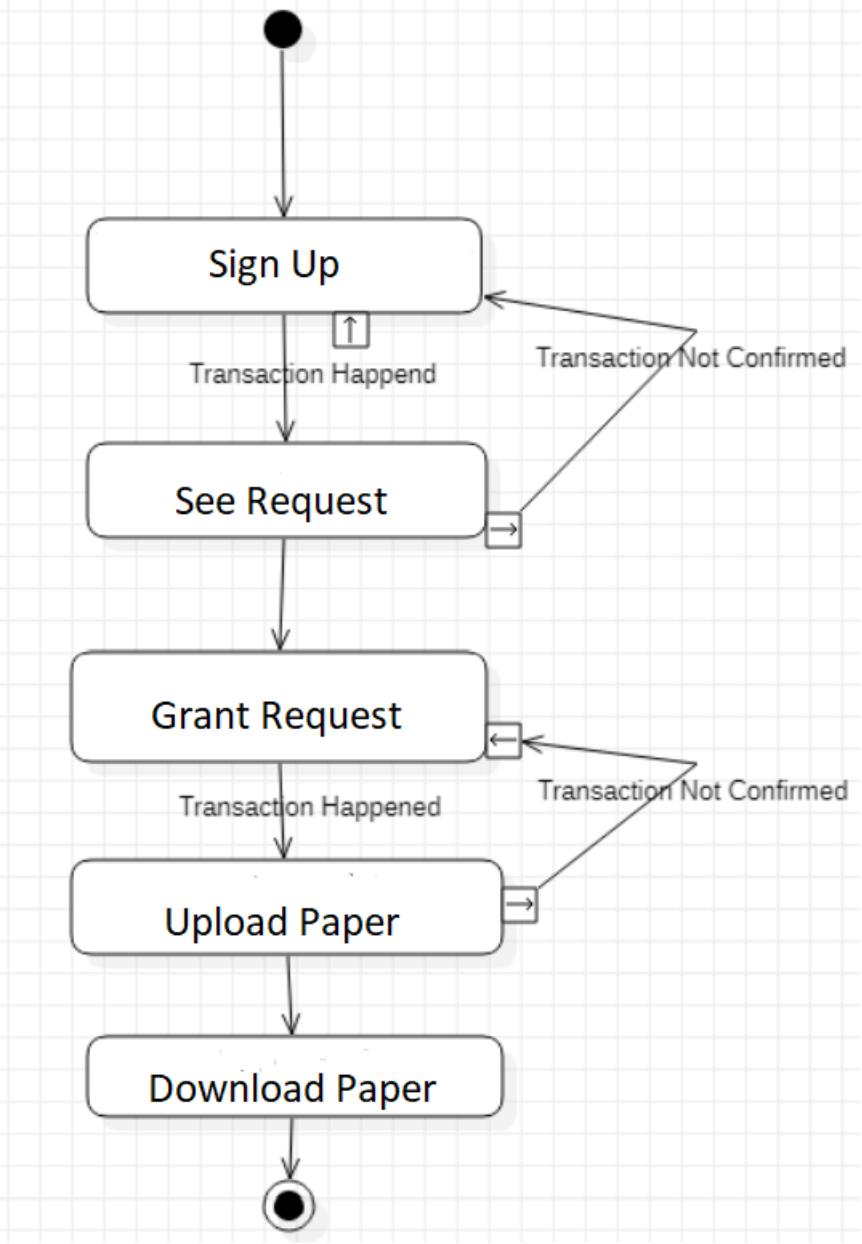


Figure 9: Activity Diagram

Admin Activity Diagram

Figure 10 shows the activity diagram of an admin who can perform different processes by using our platform which has been described above.

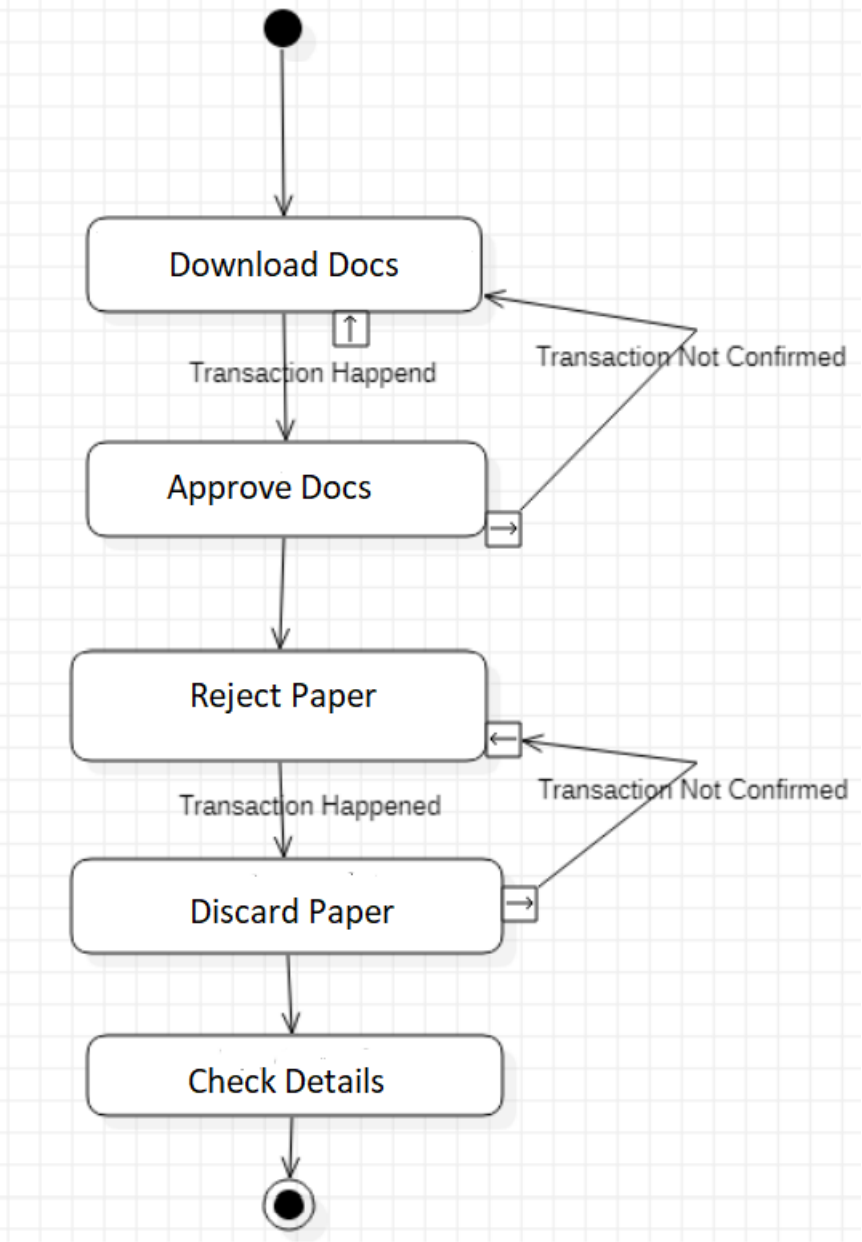


Figure 10: Admin Activity Diagram

Component Diagram

Figure 11 is also known as a UML component diagram. It describes the organization and connection of the physical components in a system. All the required info have been described in Use Case Diagrams.

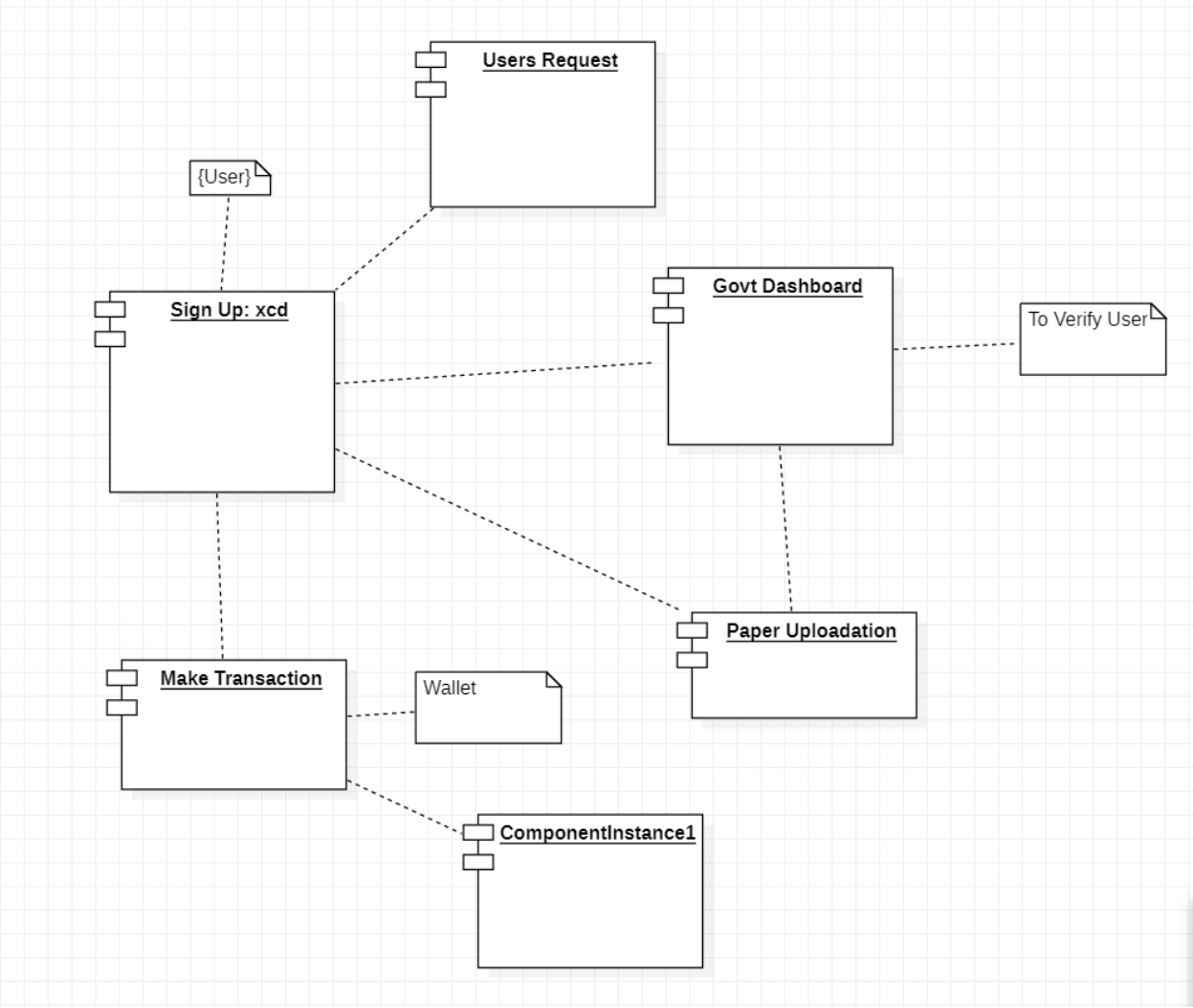


Figure 11: UML Component Diagram

Deployment Diagram

A deployment diagram is also a UML diagram type that demonstrates the execution architecture of a software system, including nodes such as hardware-software execution environments and connecting them as shown in Figure 12. This defined how to deploy the smart contract from remix to mainnet or testnet. There is a tool name Remix.ide for developing and testing smart contracts. The user just needs to connect his metamask and select the network on which he has to deploy. In our case, we selected Binance smart chain as it is stable and fast.

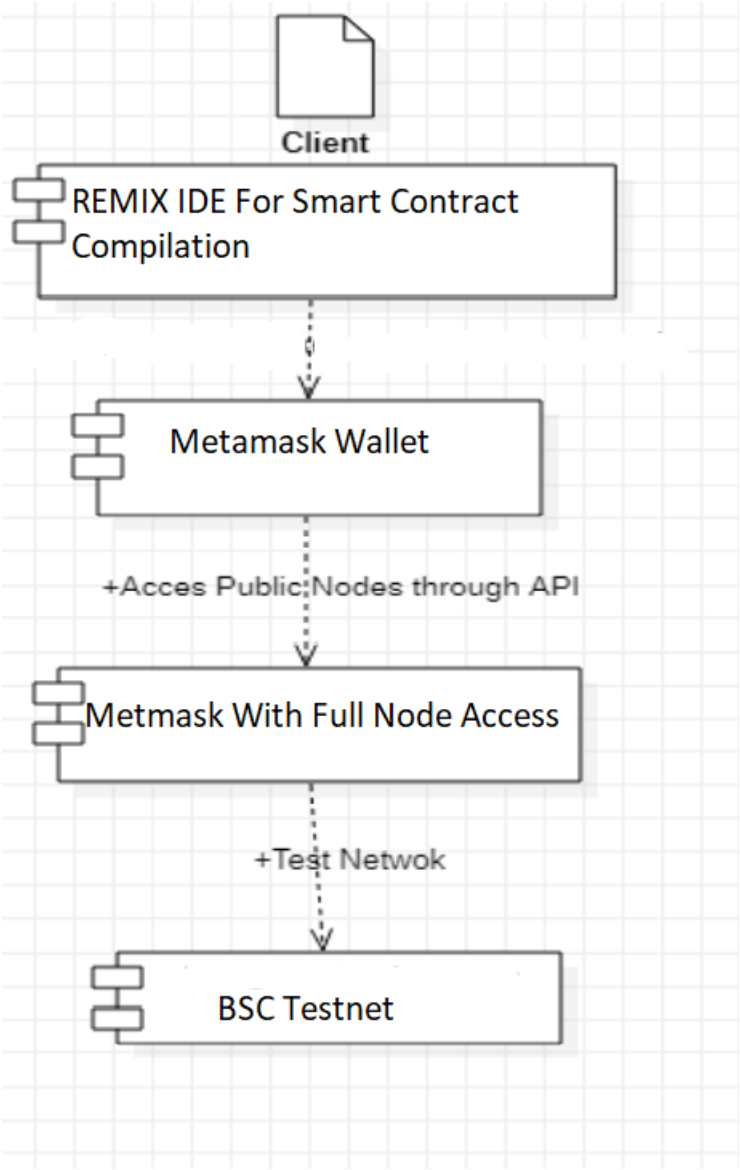


Figure 12: Deployment Diagram

Supplementary Specification

Requirements	Question	Answer	Priority
Security	Is this system secure?	Secured	High
Speed	Is this system working fast?	A little bit	Medium
Integrity	Does this system provide security of data?	Yes	High
Usability	Is this system easy to use?	Yes	Medium
Robustness	Is this system strong enough to support this?	Yes, it is	High
Reliability	Is this system reliable?	Yes	High
Recoverability	Can this system recover someone's data?	No it cannot	Low
Maintainability	Is this system easy to maintain?	Very easy to maintain	High
Scalability	Does this system provide scalability?	Not at all	Medium
Capacity	Does this system have good capacity regarding data storage and integration?	Yes, it has enough space for data storage.	High
Performance	Does this system perform well?	Yes, it is working without any interruption.	High

Table 5: Supplementary Specification

Software Quality Attributes

- ***Reliability***

  The system can be used by different users concurrently.

- ***Availability***

  The system is available 24/7.

- ***Portability***

  User can store/view data any time.

- ***Robustness***

  The system cannot be broken down.

- ***Maintainability***

  The system is quite easy to maintain.

- ***Usability***

  The system is easy to use.

Chapter 4

Software & Experimental Framework

This chapter contains in-depth information about the most valuable development assets that were utilized in the development of our project. It focuses exclusively on how it was constructed and the actual result, which has been used as an experimental source for additional investigation.

HTML

HTML5 is the fifth and newest version which offers new structures that deliver not only rich media provision, but also enhanced support for making web apps that can interact with users, their data, and servers more effortlessly and effectively than was previously possible [24].

CSS

It stands for Cascading Style Sheets which is used to create the layout style and design on web pages [25]. One can use it to change the font, character size, color, and spacing of your content or organize it into numerous columns or enhance or add animations and many other decorative features.

JavaScript

It is an immensely powerful **client-side language**. JavaScript is used for ornamental contact or interaction with a user on a webpage [26]. We can make our webpage livelier and interactive, with its help. JavaScript is used broadly in game development and mobile apps development. In our project JavaScript plays an especially important role.

Web3

Its concept was proposed at the end of the previous century, with the major focus on so-called or known as Semantic Web (the web where humans and computers have some common language anywhere between natural human language and computer code) [27]. Then the idea of liability and distribution of control was already proposed.

However, Web3 is an architecture that cannot exist without decentralization or distribution of control. BSC (Binance Smart Chain) based apps are considered the backbone of

Web3. Decentralized applications ns does not support Web2.0 (which is the existed web technology).

Node.js

It is an open-source, cross-platform environment that allows web developers to create all kinds of server-side applications in JavaScript. Likewise, the setting neglects browser-specific JavaScript APIs (Application Programming Interfaces) [28]. It also adds support for more customary OS APIs with HTTP (HyperText Transfer Protocol) and file system libraries.

To build our Dapp, few dependencies are needed first. The project's first dependency is a package manager named npm, which is included in Node.js. It can be downloaded and installed for Windows directly from the node.js' website [29]. Alternatively, on Linux, to install node, a command-line tool like homebrew is needed to deploy these commands,

- "`sudo apt update`"
- "`sudo apt install nodejs`"
- "`sudo apt install npm`"

By deploying the command "`node -v`" in both operating systems' command-line tool, someone can confirm that the installation was successful.

Webpack

Webpack is a module builder, which implies it is a tool that we utilize during the creation of our code. It is not anything that is utilized throughout the operation of our assets. As a result, it is only used during development [30]. So Webpack is a tool in which you use a configuration to tell the builder how to load specific objects. You tell Webpack how you want your JavaScript files to be loaded or how your SVG, CSS, or JPEG files to be seen. After you run the Webpack builder, it'll go into our entry point, which is our fundamental record, and it'll walk up and down our program to assemble all of these assets and figure out what to do with them. So, within the case of how we're planning to utilize it for our extend, we'll have all of our JavaScript records isolated out as they are presently, and it'll go through and bundle all of these records together to form one JavaScript record for us, so this would be our bundle it's reaching to bundle everything in together so that all of our code is optimized into one area.

React

It is an open-source, Javascript library that allows developers to rapidly create user interfaces for both web and mobile applications. It does not focus on other aspects of the application like routing or HTTP requests; it is responsible only for building rich user interfaces [31]. React feature a wide ecosystem and can be combined well with other libraries which makes it more than cable of creating a full-fledged web application; in our case a decentralized one.

To build the interface of our Dapp the installation of React is needed. Note that for React to be installed the installation of Node.js is prior needed. To install react, a terminal is needed to deploy these commands,

- "`npx create-react-app my-app`"
- "`cd my-app`"
- "`npm start`"

By running "`npm start`" a Node.js server will start; note that a React App includes only the fronted build using webpack, it doesn't include backend, that's why we use Node.js as backend in interaction with Solidity

BSC Test Network

It is also known as BSC (Binance Smart Chain) Testnet which is, as the name indicates, a testing network that performs the same protocol as BSC (Binance Smart Chain) does and is used for testing purposes before deploying on the main network [32]. However, BSC (Binance Smart Chain) has several networks. Your Ethers are on the Main Net or Ethers are on the BSC Test one. Test Networks have only one disadvantage, we cannot measure gas used here. These networks' ETHs are used for testing purposes. When developers are building decentralized applications, or testing on the network, to avoid losing money by paying real ETH for transaction fees and smart contract, it is strongly suggested to use the BSC Test Network.

Solidity

We are using solidity for this project. Solidity is a statistically-typed language. Here is a little introduction to solidity. It is an object-oriented, high-level programming language for implementing smart contracts. We already described smart contracts before. Smart contracts are programs that oversee the behavior of accounts within the BSC (Binance Smart Chain) network

and state. It was influenced by JavaScript, Python, and CPP and is designed to target the EVM BSC (Binance Smart Chain) Virtual Machine [33]. It is statically typed, supports libraries, inheritance, and complex user-defined types among other features as shown in *Figure 13*.

With the help of Solidity, we can create contracts for uses such as crowdfunding, voting, multi-signature wallets, and blind auctions. While deploying contracts, one should use the latest version of Solidity. This is because there are breaking changes, as well as bug fixes and new features that are introduced regularly. We currently use the 0.6 version.

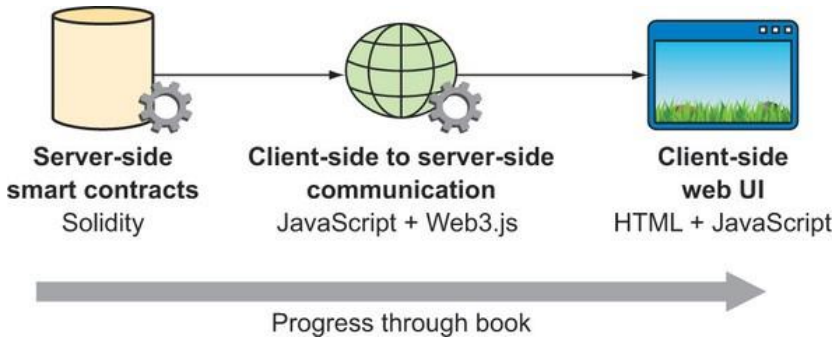


Figure 13: Solidity

Truffle

A third dependency is the Truffle framework. Truffle is a framework that gives the user not only the ability to create applications that are decentralized on the BSC (Binance Smart Chain) network but also to test them [34, 35]. It allows the use of Solidity to write smart contracts in a programming-like fashion. It provides a framework for testing smart contracts, which is useful when testing those contracts. The client-side application can be built within Truffle. Truffle can be installed by going to the command-line tool (both for Linux and Windows) and deploy the command "`npm install -g truffle`".

Truffles unique features are:

- Built-in smart contract compilation, deployment, binary management, and linking

Automatic smart contract testing with Chai and Mocha (discussed further in Chapter 5):

- This framework provides a scriptable migrations & deployment framework.
- Grid or network management for deploying to many private and public networks.
- Not good but interactive console for direct contract communication.

In this project, we have used webpack as Truffle module which is discussed below.

Ganache

Ganache is a local, decentralized blockchain used for development purposes [36]. To install it on Windows, Git [37] needs to be downloaded and installed. Then, on a command-line tool,

- `"npm install --global --production windows-build-tools"`
- `"https://github.com/trufflesuite/ganache.git"`
- `“cd ganache”`
- `“npm install”`

must be deployed. Alternatively, it can be used on Linux by downloading the last Trufflesuite’s Ganache release from GitHub [38].

MetaMask

Additionally, a web browser plug-in deemed necessary is the MetaMask extension. To use the blockchain, an individual must first interact with it, as discussed in Chapter 2, “Blockchain” of section 2.2. Metamask is an extension of your browser required to view a decentralized web application [39]. It works with a personal BSC (Binance Smart Chain) account and interacts with smart contracts by using the MetaMask browser extension as shown in *Figure 14*. An app must be installed from the browser’s application store to function.

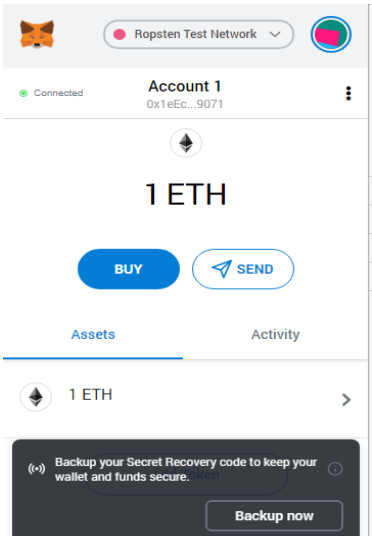


Figure 14: MetaMask

The last thing needed is syntax, highlighting the Solidity programming language that we will use to write the smart contract. Many state-of-the-art integrated development environments and text editors do not support Solidity syntax highlighting out of the box. A good editor is Sublime Text. It can be installed on Windows by downloading it from its website [40]. Alternatively, on Linux, it can be installed by deploying these commands on the command line tool:

- `"Curl -fsSL https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add –"`
- `"sudo add-apt-repository "deb https://download.sublimetext.com/apt/stable/""`
- `“sudo apt update” and “sudo apt install sublime-text.”`

Lastly, the BSC (Binance Smart Chain) package from package control of Sublime Text needs to be installed to get syntax highlighting.

Once Ganache has started, a local blockchain runs. Ganache gives ten accounts whenever it is activated. Each of these accounts has a unique address and has one hundred Ethers as shown in *Figure 15*. It is essential to mention that Ganache’s Ether cannot be used in the real network of BSC (Binance Smart Chain). These account addresses are unique identifiers that will represent the user’s identities on our decentralized application.



Figure 15: Ganache

As the local blockchain is running for the application to be structured, creating a folder is needed. The Truffle framework provides sample projects, called packages. These are already implemented projects that can be unboxed within a folder to give the user a head start. The Truffle pet shop box is used. This is a package published by Truffle for their pet-shop tutorial [41]. To unbox this, `"truffle unbox petshop"` should be deployed within the command-line tool. By examining the project folder's directory structure, it contains a folder named "contracts," It's where smart contracts are held. A smart contract named migrations.sol exists, and it is just a smart contract that is going to handle the migrations whenever all our smart contracts are deployed to the blockchain.

InterPlanetary File System

The internet is such a valuable tool in our daily life. We use it to access the media, connect with friends and colleagues, learn to manage finances, and more. Still, the web, as we know it, has a problem because the information on it is centralized. It is all on servers and operated by a single corporation. Centralization poses another security concern, censorship; since the content is hosted on just a few sites, it is easier for governments to block. In 2017, the Turkish government ordered internet providers to block Wikipedia due to its unfavorable representation of the country [42]. Centralization of the web is not a positive thing, but it is used because of what it can offer. We want pages to load in an instant and everything with the highest quality possible. Server centralization allows for full control of how quickly it can provide content. We use this model because there is no perfect alternative, but this could change soon with the IPFS project, which hopes to make the web fully distributed via a peer-to-peer network that works similarly to how BitTorrent works [43, 44].

However, to achieve these goals, someone first needs to understand how we navigate the Internet as it is today. Suppose we try to download a photo from the Internet. In that case, when we do so, we tell the machine exactly where to locate the image. In this case, the picture's location is the I.P. address or the domain name; this is called location-dependent addressing. We tell the computer which information to collect, but this information cannot be retrieved if the server is not available. When that happens, there is a substantial risk that someone else downloaded the picture before, and you will be unable to recover it from that user. To fix this, we switch from location-based addressing to content-based addressing. Instead of asking where

to find something, you say what it is you want, and this can be done because every file has a unique hash that can be compared to a fingerprint. If we are going to download a present file on the IPFS network, someone on the web can provide it to us. Since we use a hash function to request, we can verify what we have received. We request a file with a specific hash, so when we receive the file, we check if the hash fits with what we have received; protection is built-in. An excellent advantage of using hashes to address content is how easily IPFS decreases the storage costs by only storing one version of a file. Files are stored in IPFS objects containing up to 256kb of data, and they can also connect to other IPFS objects. A small hello world text file can be stored as a single IPFS objec.

When files are more extensive than 256 kilobytes, they are broken into IPFS objects that are 256 kilobytes in size, and then the device can build IPFS objects that point to all the other parts of the file. IPFS data architecture is straightforward, but it can be compelling. This architecture enables users to use it as a file system [45]. A simple directory structure with some files as shown in *Figure 16*, can be converted into IPFS objects and create one object for each file and each directory. This is not what we see because IPFS cannot change the address until the content is added. It is like a blockchain. IPFS allows for the versioning of data. Suppose we are exchanging an essential document over IPFS or want an entity to have access to it. When we make a new commit, IPFS generates a new commit object for us. This object is fundamental; it informs IPFS that commits before it.

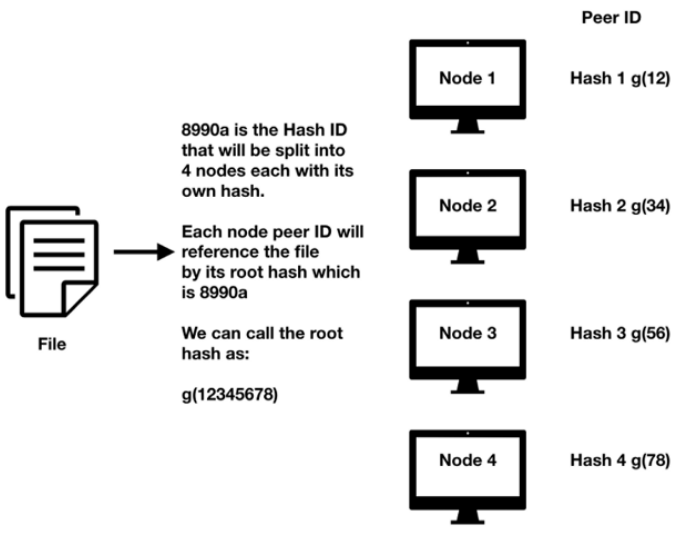


Figure 16: IPFS Structure

Furthermore, let us assume that we want to update this file, as well. So we add our modified file to IPFS. The program will then generate a new commit object for the file. This command object now links to the previous commit, and this process can be repeated indefinitely. IPFS can ensure the availability, integrity, and replicability of all data stored on the network. This is a useful tool, but it has some limitations. The main challenge in using IPFS is keeping the files open. Every node on the network maintains a cache of the files that it has downloaded and distributes them if other people need them. Still, if a particular file is hosted, for example, by four nodes and those go offline, then that file becomes inaccessible, and no one can catch a copy of it (like BitTorrent files without seeders). There are two workable solutions to this problem. One is to incentivize people to store files and make them available. Another is to distribute files and make sure they are still available proactively. Filecoin plans to do this. Filecoin was introduced by the same company that is running IPFS. It is a blockchain at the base layer, designed on top of IPFS, and aims to develop a decentralized market for file storage [46]. If a person has a spare hard drive, you can rent it to others to make money. Filecoin's site allows nodes to keep files online if possible because otherwise, they will not get paid. The framework also ensures that files are replicated on many nodes so they cannot become inaccessible. That is just a brief overview of Filecoin and how it plans to build on top of IPFS to solve some of these issues.

Chapter 5

In this chapter, we will go through the full Dapp. We will show also show how users can use this application and what the role named government can perform. Moreover, we will see how to allow other users to request data and download its content. The same data are stored in smart contract as well as in doc files in IPFS, so requester should be permitted to download or read them. The IPFS is associated with the user account address in smart contract with the use of mapping. There are two websites, one for the government to verify users' data and the other is the client's one which contains all of his information. Besides the explanation, there is a detailed evaluation of our Dapp along with a comparison of why our approach in Binance Smart Chain is better than the Ethereum ones.

Troubleshooting Guide

Ahead to UI, the most important parts of smart contract code will be explained.

Roles

```
“bytes32 public constant Access_ROLE = keccak256("Access_ROLE")”
```

In this line of solidity code, we are making a hash of values that is in brackets. The keccak256 function will return us a 256-bit hash of this string. Then we are assigning this hash value to Access_ROLE variable and we will get its value in bytes32 because of data type. This is constant and its value cannot be changed throughout the contract.

We used this function from OpenZeppelin access control to manage access for the data of the user. This smart contract library is well audited and used by many developers.

These bytes values will be used to assign unique roles to different users against their account addresses.

```
“bytes32 public constant Govt_ROLE = keccak256("Govt_ROLE")”
```

As the name suggests, this role will be used for government entities. This role will be assigned to a government address in smart constructor. That address shall verify the data of data holders on this platform. The government address will be the externally owned account, and this will be a user doing these.

Why Role Assigned in Constructor?

At contract deployment time, the constructor and its logic written in there is executed only once. So, we assigned the role of the constructor.

There are many code blocks, most important ones are shown, to improve understanding.



```
struct File {
    string hashVal;
    string firstName;
    string middleName;
    string lastName;
    string DOB;
    string residenceAddress;
    string country;
    uint IdCode;
    bool verified;
    bool hasFile;
 }
```

Custom Data Type For Storing Details

This is the custom data type we are using for storing user's data in smart contract. Data for IPFS will be stored in doc's file. There are multiple data types in this structure. Let me describe this one by one for better understanding. Please do not get confused by the names of variables, we used the camel case for naming conventions.

- **HashVal:** This is the hash value in string data type. When we input data to IPFS it returns a hash value, so we store that hash value in this variable. This is necessary to create a record.
- **firstName:** This is String data type and is used to store the name of the user. He should add his name in string format while signing in to the platform with MetaMask, which means after putting all required details about him will sign-up and his data is recorded on the blockchain through this smart contract.
- **MiddleName:** Middle name of the user.
- **LastName:** User's last name.
- **DOB (Date of Birth):** The date of birth of the user is stored in String field in the smart contract. To prevent association and complexity in a smart contract, we have used string data types. This DOB is fetched from JS, which means the user can see the calendar UI to select his date of birth. From the calendar, the user can select dates and year, etc. and this value will be stored in contract string data type.
- **ResidenceAddress:** This variable stores the address of the user.
- **Country:** This variable will store the country name.
- **IdCode:** You can think of an Id code as a user identity card number. Simply saying this data type is for storing the user's Id number.
- **Verified:** This is a Boolean type variable. It returns true or false based on data verification. If the data is verified by a government authority, it returns true. Otherwise, it will return false.
- **HasFile:** This is the main variable. It is also a Boolean type. If the user has not uploaded his data file, it returns false. If he uploaded data by calling the function from the front end it returns true.



These are mappings to store requests of users against data holder addresses and it also stores those addresses who have access to see data of other users.



```
constructor(address govt_Address) {
        _setupRole(Govt_ROLE, govt_Address);
        _setupRole(Govt_ROLE, msg.sender);
        accessorCount = 0;
    }
```

This is the constructor that executes only once after the deployment of the smart contract. We assigned three roles in this.

- `_setupRole(Govt_ROLE, govt_Address)`

This is setting a Govt_ROLE to government address to verify the data of users on this platform. In the govt_Address parameter, we pass the address of the government authority, and it is a person behind this doing verification process by matching data to various sources of the user. For example, the government has data of all users, or you can think about Google, it matches this data with the data in their database and performs a verification process.

- `_setupRole(Govt_ROLE, msg.sender)`

In this role, we assigned Govt_Role to msg sender. We have done this only for reading data from web applications. The Govt authority and Admin both can be the same entity. So, it is easy for us to show data on websites.

- `AccessorCount = 0`

This is the counter which keeps track of requester count. If there are 10 requesters it returns 10.



```
function getDataForUI(address _user) public view returns(string memory name, string memory hashVal,bool
verified,bool hasFile){

        return (userInfo[_user].firstName,userInfo[_user].hashVal,userInfo[_user].verified,
userInfo[_user].hasFile);

    }
```

This function is used to get user data and to show it on the web application front end. It returns all data that we see as I custom data type.



```
function checkVerified(address addr) public view returns(bool){

        return userInfo[addr].verified;

}
```

This function returns a Boolean value based on data verified by govt authority. It returns true if the data is verified and false if data is not verified.



```
function giveAccess(address to) public {
        hasAccess[to][msg.sender] = true;
         giveRole(to);
}
```

`giveAccess` function is used to grant requester access to access the data file and data fields. This is the most important function in this smart contract.



```
function signUp(string memory _firstName, string memory _middleName, string memory _lastName,string
memory _DOB,string memory _residenceAddress, string memory _country, uint _IdCode) public returns(bool)
{
         require(userExists(msg.sender) == false, "userExists");
          userInfo[msg.sender].firstName = _firstName;
          userInfo[msg.sender].middleName = _middleName;
     userInfo[msg.sender].lastName = _lastName;
          userInfo[msg.sender].DOB = _DOB;
          userInfo[msg.sender].residenceAddress = _residenceAddress;
          userInfo[msg.sender].country = _country;
          userInfo[msg.sender].IdCode = _IdCode;

            AddressForMapping.push(msg.sender);
             AddressForMapping.length;

       return true;
 }
```

This function is called when the user signs up from MetaMask. It stores all his details in the contract.



```
function storeHashToChain(string memory key) public returns(bool){
        require(userExists(msg.sender)== true,"Please signup first");
        userInfo[msg.sender].hashVal = key;
        userInfo[msg.sender].verified = false;
    userInfo[msg.sender].hasFile=true;
    return true;
 }
```

To store the hash of the file in the blockchain, this function is used.



```
function getRequestors(address userAddr) public view returns(address [] memory, uint){
        return (requestors[userAddr].toAccess,requestors[userAddr].toAccess.length);
 }
```

To get all users who send requests from smart contracts to get access to other user's data, this function should be called.



```
function removeRole(address addr) onlyOwner public {
 revokeRole(Access_ROLE, addr);
}
```

To revoke access of any user to data, the function should be called data holder address.



This function is used to grant access role to any address, then he can see granter data.

Now, someone can truly understand how this smart contract is working. Let us review the user manual now. We have two websites, one for Client as shown in *Figure 17* and one for the Government as shown in *Figure 18*. See the screenshots below.

Client Website Landing Page

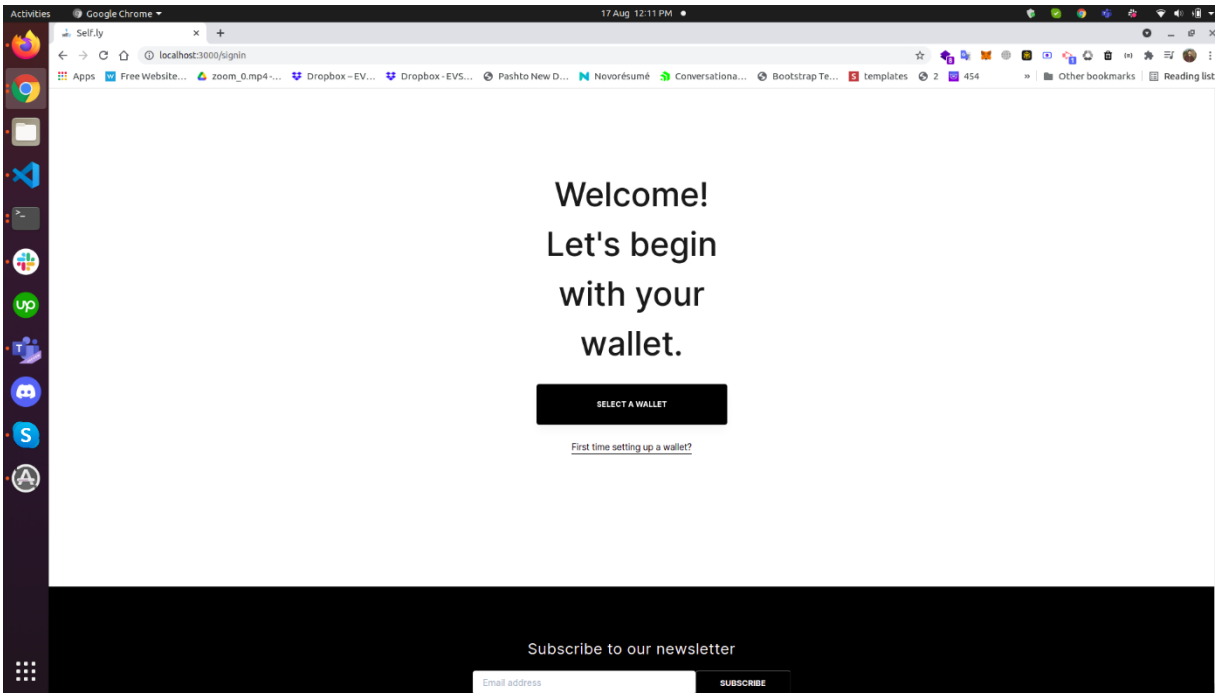


Figure 17: Landing Page

This is the landing page, the user should sign-up from here with his wallet. The most used and good wallet is MetaMask. If he is signing up for the first time, he needs to click on setting up wallet first and fill in all the necessary details.

Government Website Landing Page



Figure 18: Government Website Landing Page

It will check connecting address if he has a Govt approved address or not. It will reject him if he does not have the correct government address.

Client Website Dashboard

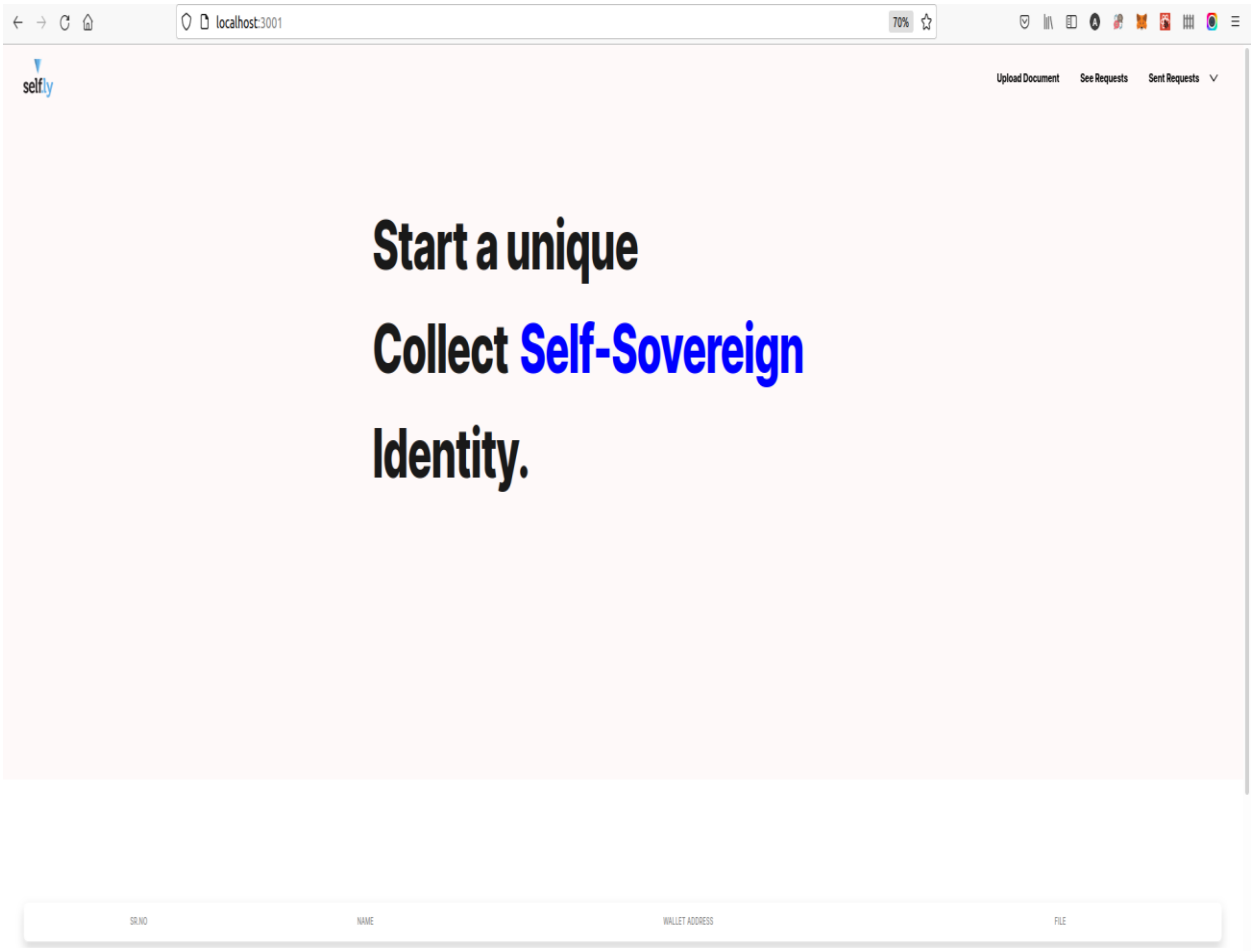


Figure 19: Client Dashboard

If you see the header section, you will notice there are three menu tabs on the right side and one logo on the left side as shown in *Figure 19*. This logo is pointing at the home page.

Logo: Pointing to home page.

Upload documents: This page is for uploading documents to IPFS.

See Requests: The user can see all requests on this page that are made to see his data so far.

Sent Requests: This page shall show all requests sent by this user.

Login Page For Client Website

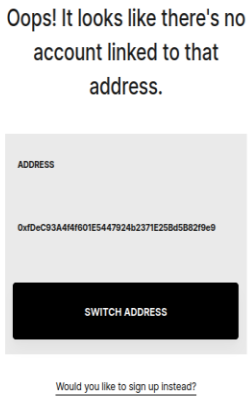


Figure 20: Login Page

If a user has signed up before. He can sign in from here as shown in *Figure 20.*

Sign-Up Page

If you remember in the upper smart-contract description section, A custom structure, was described; this data is added in those fields as shown in *Figure 21*.

This date will be selected from the date and library. The user can see the calendar UI and select data from that.



Figure 21: Sign Up

Client-Side Dashboard

As shown in *Figure 22*, the client-side website is pictured. The user can see other user's address, name, and button to request data.

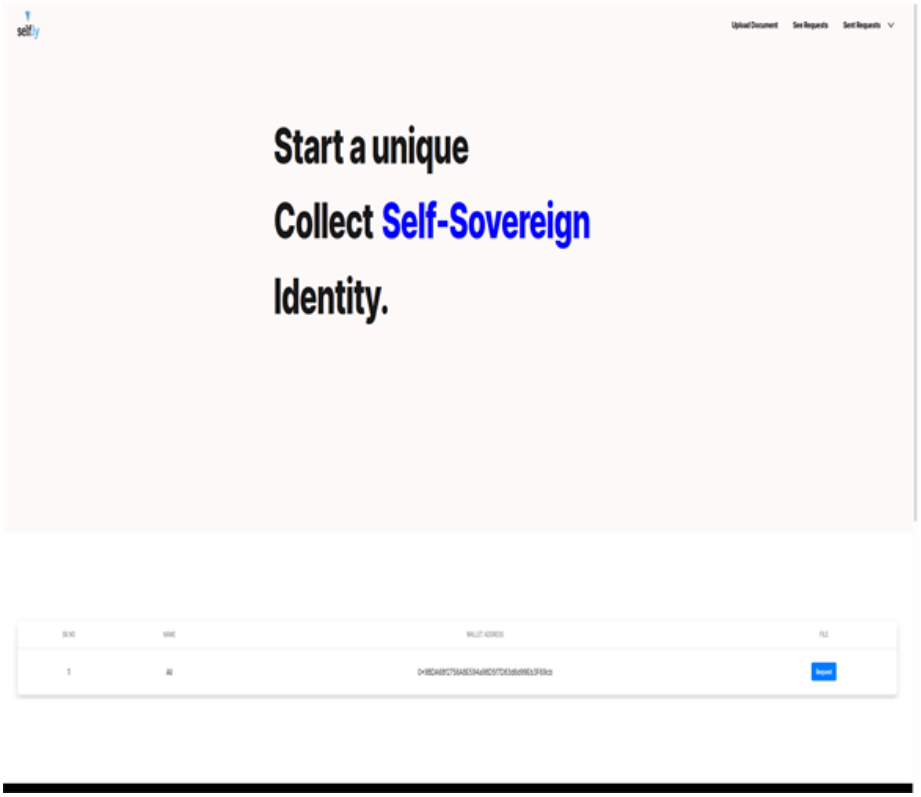


Figure 22: Client Dashboard

Upload Data to IPFS

On this page, the user will upload his data-contained file to IPFS as shown in *Figure 23*. That should contain all the details about him. The fields that the user filled while sign-up is necessary to be added to this file. Users can add more details about themselves in the file to make it more understandable by viewers and govt authorities.



Figure 23: Upload IPFS

Sent Requests Page

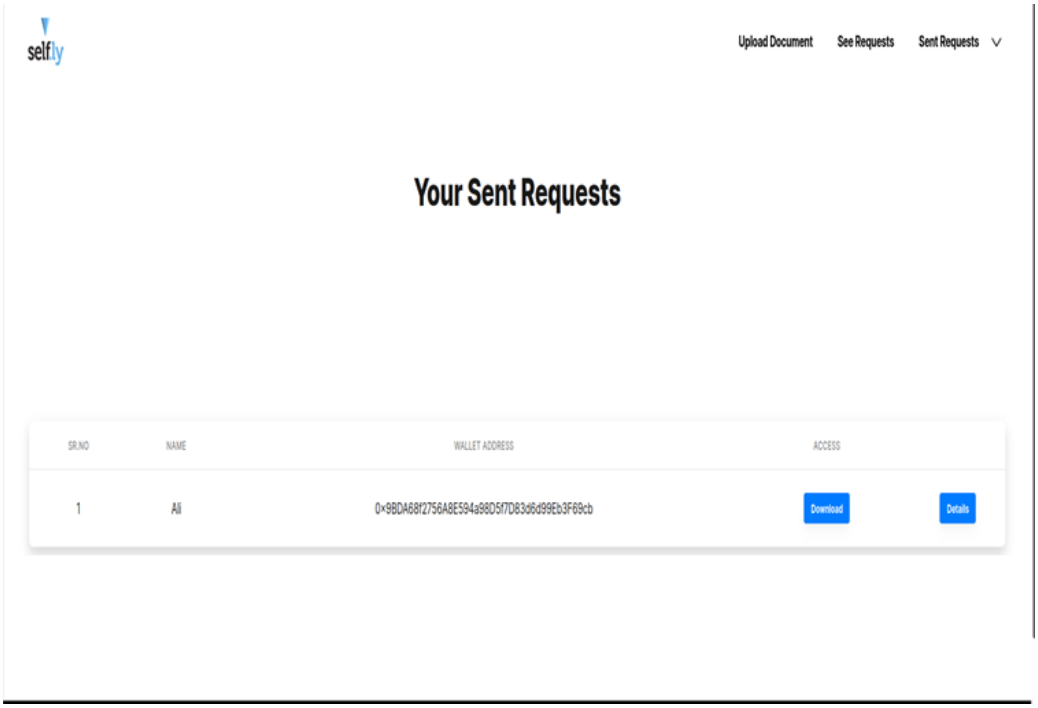


Figure 24: Sent Request

As shown in *Figure 24*, the page which shows sent requests by a user who is currently logged in; means who sent requests to get access to data of other users.

Govt Dashboard To Verify Data

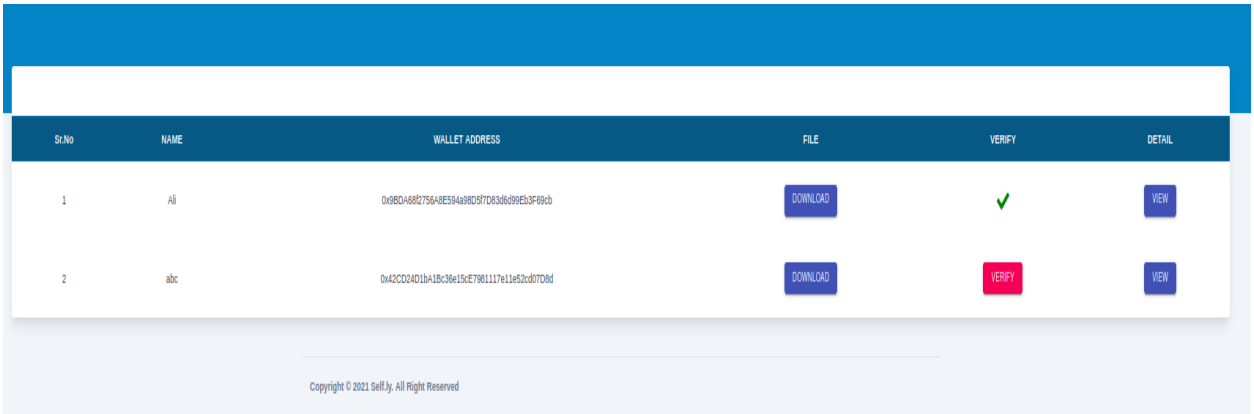


Figure 25: Govt. Verify Data

In *Figure 25*, there are two rows, one is verified and the other is not verified. The government must click on the verify button to verify the user's data as shown in *Figure 26.*

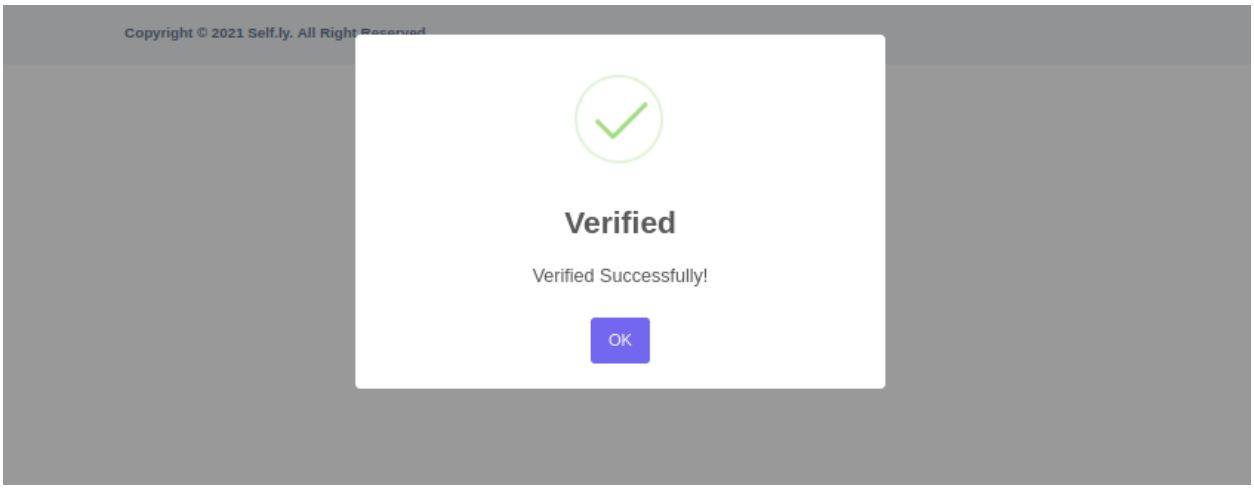


Figure 26: Govt Verified

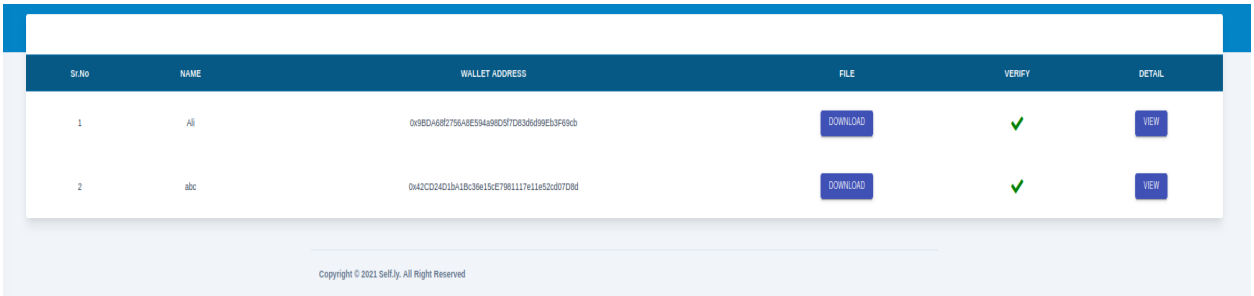


Figure 27: Govt Req

Figure 27 shows the result of verifying data. It is visible that there is a verified checkmark.

Grant Access Page

The user can grant data view and file download access to the requester as shown in *Figure 28*.

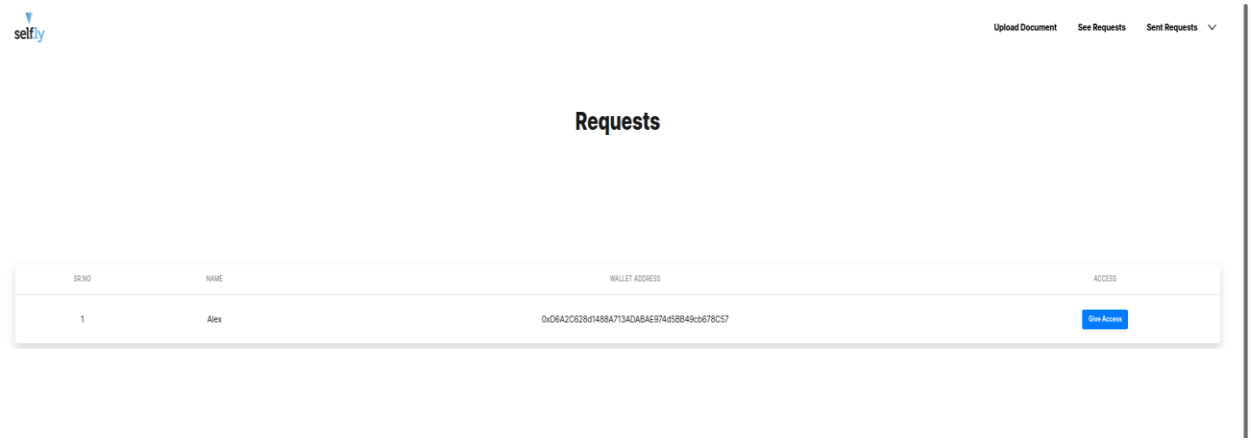


Figure 28: Grant Access

Experimental Evaluation

Black Box Testing

This test includes the information admin looks for while admin can see the details of the user.

Test Case ID		1	Test Case Description		Test the Lookup User Info						
Created By		I	Reviewed By		I		Version		0.1		
QA (Quality Assurance) Tester's Log		Review comments from I incorporate in version 0.1									
Tester's Name		I	Date Tested		5-Jul-2021		Test Case (Pass/Fail/Not Executed)		Pass		
S #	Prerequisites:				S #	Test Data					
1	Access to Chrome Browser				1	Look Up for User Info					
2	Access to the BSC (BINANCE SMART CHAIN) Test Network										
Test Scenario	Verify on entering valid User Info										
Step #	Step Details		Expected		Actual Results		Pass / Fail / Not				

		Results		Executed / Suspended
1	Transaction from MetaMask	Test Network should open	As Expected	Pass
2	Put It on User Info Section	Credential can be entered	As Expected	Pass
3	See Requests	Reject & Approve	As Expected	Pass

Table 6: Test 1

This test case ensures the data is uploaded on the IPFS properly.

Test Case ID		2	Test Case Description		IPFS					
Created By		I	Reviewed By		I		Version		0.2	
QA Tester's Log		Uploadation on IPFS								
Tester's Name		I	Date Tested		5-Jul-2021		Test Case (Pass/Fail/Not Executed)		Pass	
S #	Prerequisites:				S #	Test Data				
1	Access to Chrome Browser				1	Login				
2	Access to the BSC Test Network									

3	Balance in BNB wallet	

Test Scenario	Login with MetaMask		
Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not Executed / Suspended
1	Put File upload	Test Network should open	As Expected	Pass
2	Save the address	By which User Uploads File	As Expected	Pass
3	Click on the Upload Button	Uploading	As Expected	Pass

Table 7: Test 2

This test case represents granting access to the user so the user can access the documents.

Test Case ID	3	Test Case Description	Granting Access to users		
Created By	I	Reviewed By	I	Version	0.3
QA Tester's Log	Review comments from I incorporate in version 0.1				

Tester's Name	I	Date Tested	5-Jul-2021	Test Case (Pass/Fail/Not Executed)	Pass

S #	Prerequisites:		S #	Test Data
1	Access to Chrome Browser		1	Upload file
2	Access to the BSC Test Network			
3	Balance in Ether Wallet			

Test Scenario	Ask Access Transaction Should be Confirmed & Access Granted			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not Executed / Suspended
1	Requesting	Putted	As Expected	Pass
2	Select file	Available In Request Section	As Expected	Pass
3	SR # No	Incremented	As Expected	Pass

Table 8: Test 3

White Box Testing

Integration Tests

Integration tests help blockchain developers and testers to ensure the communication of their code between different mechanisms, and between external and internal systems like databases.

User Interface

UI (User Interface) testing exposes how the request works from the end-user perspective. It is significant to ensure that we carry out UI testing to make sure the experience is optimal or that they at least get the correct feedback from the Dapp when it does not perform well. The following figure shows the flow of tests in JavaScript:

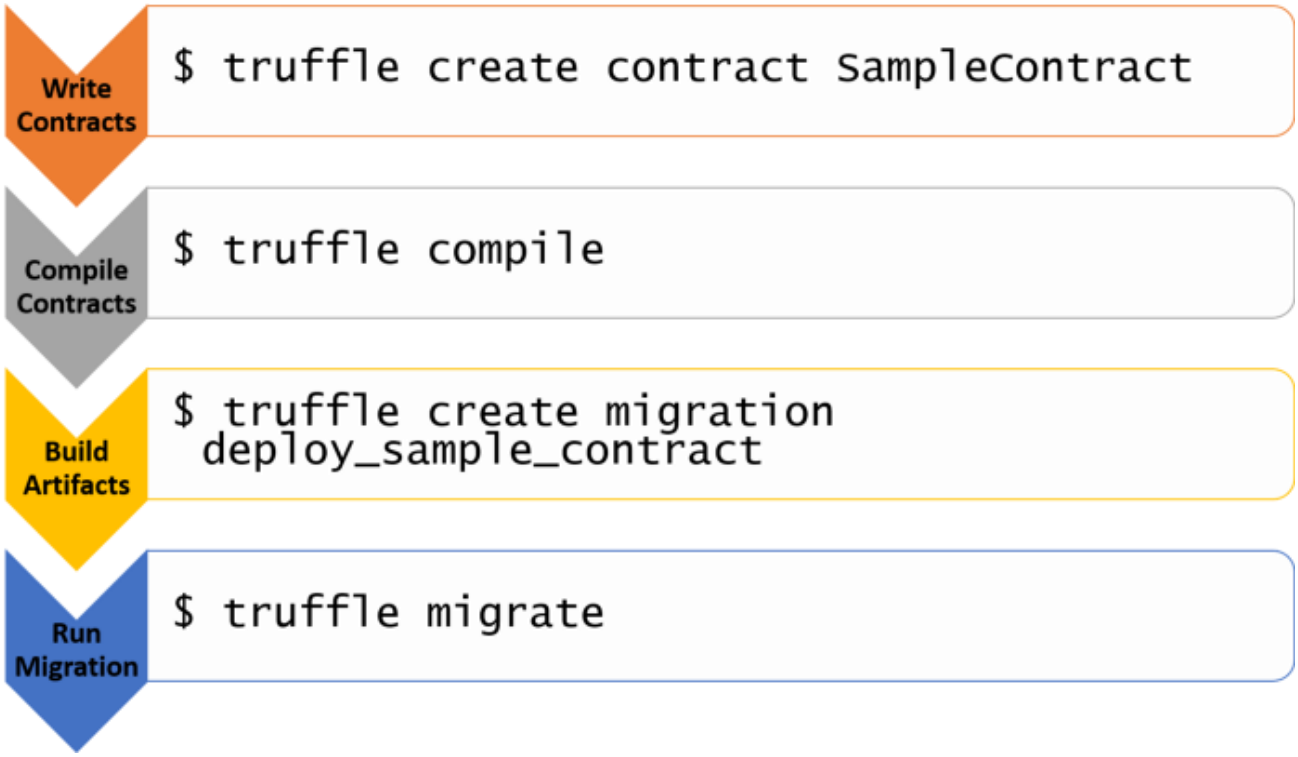


Figure 29: Truffle Flow

Prerequisite:

The machine must have node and npm installed before proceeding.

We must have nodes LTS version from Node's official website.

Then we need to install Mocha:

Mocha:

- It is a JavaScript framework.
- Only runs on node.js
- It can be installed by executing the command "`npm install --global mocha`".

Testing a Function with Mocha

Each function does a unique task, although the test function needs to be called in from the test or any specific file with desired inputs. In the end, we will put a declaration to validate the output or task of the function.

What is Chai

- Chai is a TDD/BDD assertion library.
- Chai can be paired with any JS testing framework.
- Declaration with Chai provides natural language declarations, readable and expressive style.
- It can be installed by executing the command "`npm install --global chai`".

Testing Smart Contracts

First, we need a truffle project which is created by the following command "`truffle unbox webpack`" which unboxes webpack:

After running this command, we can see different directories as a project could be.

- Truffle-config.js
- Contract
- Migration
- Test

The testing directory is the directory in which all the tests fall. All tests like JavaScript and smart contracts fall into it.

First, we start ganache-cli so transactions can happen in the personal blockchain.

- `"Ganache-cli"`

Then we compiled our project by using the following command.

- `"Truffle compile"`

After writing migration we must migrate our project.

- `"Truffles migrate"`

Then we interact with the truffle console

- `"Truffle console"`

State of the art comparison

Without a question, the Binance Smart Chain (BSC) has seen a surge in activity. The Binance Smart Contracts team has developed smart contracts for tokens on the Binance-perceived blockchain, ranging from decentralized money markets to token exchanges. Many clients are switching from Ethereum-based blockchain to Binance-branded blockchain in this regard [47, 48]. The following paragraphs of this blog present a full examination of the benefits of BSC over ETH, as well as reasons why many people are adopting it.

BSC provides a network that is designed for widespread adoption and scalability. During last year's DeFi summer, Ethereum transaction fees skyrocketed. This discouraged client participation and hampered DeFi's growth. Developers were obliged to create another more accommodating network, and thus BSC was born. Furthermore, the transaction cost and gas price on a BSC network are lower than on an ETH network. On average, the gas price on BSC is just 10% to 20% higher than the price on Ethereum. As a result, retail markets or Dapps which need smaller wallet sizes choose BSC [49]. The Binance Smart Chain now offers a new capability for easily and quickly deploying smart contracts in a fresh new blockchain application. This feature is not available in ETH.

Furthermore, BSC is fully interoperable with ETH, and all users can use the same wallet address. ETH does not accept all of its clients, particularly those with below-average test scores. The Binance Chain, best of all, creates a direct link between several blockchains by leveraging p-Tokens (p-Network), which removes any barriers between the chains. As a result, because it is

safeguarded by cross-chain composability, it allows for frictionless movement of both users and crypto liquidity [50]. The p-Network bridges the gap between individual blockchains. In turn, BSC connects to Ethereum, giving users access to both ecosystems.

To summarize, the Binance Smart Chain is seeing rapid growth and user engagement is increasing all the time. Compared to ETH, this platform has various advantages, making it a better solution for Dapps.

	Binance Smart Chain	Ethereum Network
Gas Price	16 Gwei	260 Gwei
Block Time	3 secs	13 secs
Unique Addresses	5.2 million	140 million
Daily Transactions	2.6 million	1.3 million
Scalability	Yes	No

Table 9: Comparison of BSC & Ethereum Network

Chapter 6

Conclusion & Future Work

Self-sovereign identity introduces a revolutionary and innovative solution for the management of individual digital identities. This model has the potential to solve the problems and inconveniences presented by the current identity management systems in terms of regulation, technology, and security. In doing so, this model combines two innovative technologies: digital wallets and decentralized ledgers. Digital wallets will allow individuals to manage all their digital assets with total independence, sovereignty, and privacy. We will be able to have fast and safe access to digital versions of identity documents, academic diplomas, and property titles, among other valuable identifying information. We will also be able to manage tokenized fiat currencies such as the dollar, euro, libra, pesos, yuan, and even cryptocurrencies. Decentralized ledgers, such as blockchain networks, allow for the storage of cryptographic proofs of the existence and ownership of digital credentials and assets, increasing the trust, interoperability, security, and efficiency of electronic transactions while preserving data privacy.

About regulations, it is necessary to work on the areas of electronic transactions and signatures as well as data privacy and protection to develop and update regulatory policies in countries that lack them. In Latin America and the Caribbean, 31 out of 42 countries (74%) have regulations on electronic signatures. However, only 17 out of 42 countries (40%) have regulations on data protection.

As stated by the ISO, ITU, NIST, and the European Union, among others, in the next few years we will see an intense and rapid growth in activity surrounding self-sovereign identity [51]. It is possible that in a short time, we will be managing all of our digital assets with a mobile wallet and using blockchain to guarantee its veracity. As with any important and disruptive technological innovation, there is a large and timely opportunity for governments and private sector companies to take their first steps forward.

To conclude, the SSI community aims to build a universal identity layer for the web and enable technical interoperability between identity systems of all sorts that put the user in control. Promising projects like Hyperledger, Uport, Sovrin, Eurocom, bloom, and many more already enable us to experience a small taste of what is coming. Nevertheless, further exploration, standardization, and integration will be necessary to develop a holistic framework and unleash the immense potential of self-sovereign identity.

Appendix A

WBS (Work Breakdown Structure)

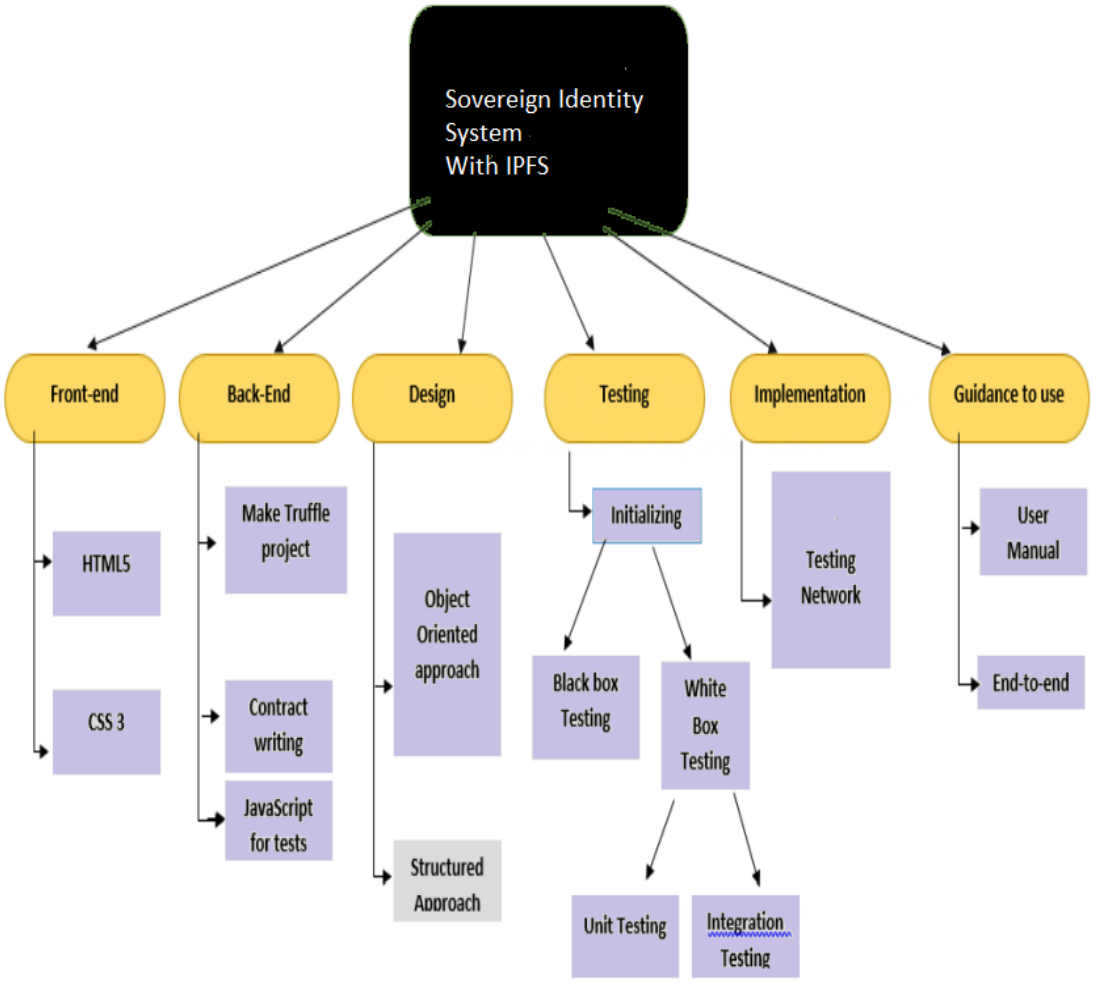


Figure 30: WBS

Appendix B

PERT Chart

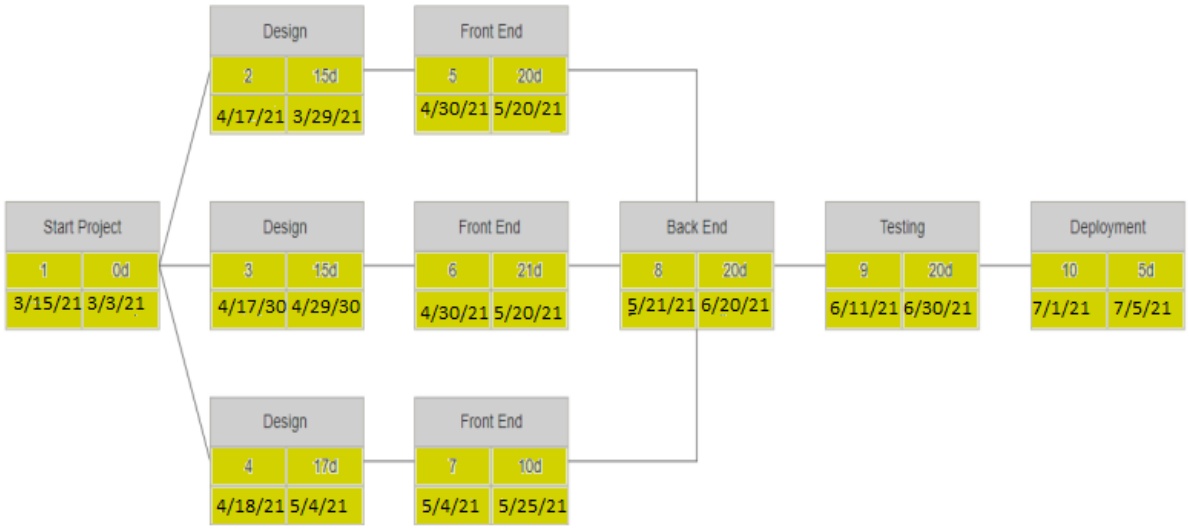


Figure 31: Pert

Appendix C

Critical Path

1 → 3 → 6 → 8 → 9 → 10

Appendix D

Blockchains & Their Approach

This figure represents gas costs and time for different blockchains. We've used Binance Smart Chain as it is stable and fast.



	SOLANA	AVALANCHE	COSMOS	Polkadot	BINANCE SMART CHAIN	ethereum
TPS	50,000	4,500	1,400	400	100	15
Finality Time	0.4 second	1-5 second	1-3 second	-	75 second	5 min
Avg Tx Fee	$0.00001	$0.03	$0.03	-	$0.01	$4-$40
Number of Validator	685	865	125	297	21	6,833
Total Transactions	14 B	1.9 M	290 K	1.4 M	144 M	1 B
Fully Diluted Valuation	$7.1 B	$21.6 B	$4.6 B	$37.4 B	$44.2 B	$196 B
Type	Layer 1	Layer 1	Layer 1	Sharding	Layer 1	Layer 1
USDC or USDT support	Yes	No	Yes	No	No	Yes

Figure 32: Blockchains Approach

Appendix E

Roles & Responsibility

Nikolaos Papadakis	The person provided motivation for the project and gave us a path to do that.

Responsibilities

- Acts as chairperson of the project.
- She is accountable for the delivery of planned benefits relevant to the project.
- Ensures resolution of issues escalated by the Project Manager or the Project Board.
- Sponsors the communications program.
- Establishes goals within the organization.
- Makes beneficial commercial decisions for the project.
- Also assures availability of essential project resources.
- Approves the decision tolerances.
- He has the ultimate authority and responsibility for the project.

Title	Role
I	The person who is responsible for developing, in conjunction with the Project Sponsor or a definition of the project. He ensures that the project is submitted on time also maintain the budget and to the required standard quality.

The person who is responsible for developing, in conjunction with the Project Sponsor or a definition of the project. He ensures that the project is submitted on time also maintain the budget and to the required standard quality.

Responsibilities

- Championed our project
- Raised awareness at sense level about blockchain in local peoples.
- Made strategies, implementation, project scope, and milestones.
- Resolved strategic and policy issues.
- Managed change through implementation.
- Smart contracts & JavaScript
- SQA

Milestones & Deliverables

Main Task	Start	End
Project Selection	25-Feb	3-March-2021
Approval	15-March-2021	16-March-2021
Design	17-April-2021	29-April-2021
Front-End	30-April-2021	20-May-2021
Back-End	21-May-2021	10-June-2021
Testing	11-June-2021	30-June-2021
Deployment	1-July-2021	5-July-2021

Potential Risk & Issues

Issue / Risk	Description	Probability (H, M, L)	Impact (H, M, L)
Bounded Version Language	Solidity is used in this project for smart contracts writing. However, recently there are too many versions of this and specific frameworks like truffle for every version. It is extremely hard to write code.	H	L
Hardware	This is an excessively big issue that we need the best version of hardware for blockchain development.	M	M
Lack of Logics	As we all know that blockchain is a developing technology. It is hard to make logic in programming with low usage of gas.	H	H

References

1. Westjohn, Stanford & Arnold, Mark & Magnusson, Peter & Zdravkovic, Srdan & Zhou, Joyce. (2009). Technology readiness and usage: A global-identity perspective. Journal of the Academy of Marketing Science. Available: http://dx.doi.org/10.1007/s11747-008-0130-0

2. Giannopoulou, A. & Wang, F. (2021). Self-sovereign identity. Internet Policy Review, 10(2). Available: https://doi.org/10.14763/2021.2.1550

3. Kuperberg, Michael. (2019). Blockchain-Based Identity Management: A Survey From the Enterprise and Ecosystem Perspective. IEEE Transactions on Engineering Management. pp 1-20. Available: http://dx.doi.org/10.1109/TEM.2019.2926471.

4. A. Zwitter and J. Hazenberg. (2020). Decentralized Network Governance: Blockchain Technology and the Future of Regulation. Front. Available: https://research.rug.nl/files/120767599/fbloc_03_00012.pdf

5. W3C VCWG. Verifiable Claims Working Group Frequently Asked Questions. [Online]. Accessed: 25/09/2021. Available: https://w3c.github.io/webpayments-ig/VCTF/charter/faq.html

6. V. Buterin. (2013). A next-generation smart contract and decentralized application platform. Ethereum white paper. Accessed: 25/09/2021. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper.

7. Meinel et al. (2018). Blockchain: Hype or Innovation. ISBN: 978-3-86956-394-7. [Online]. Accessed: 25/09/2021. Available: https://publishup.uni-potsdam.de/opus4-ubp/frontdoor/deliver/index/docId/10314/file/tbhpi113.pdf.

8. Halpin, Harry & Piekarska, Marta. (2017). Introduction to Security and Privacy on the Blockchain. 1-3. Available: http://dx.doi.org/10.1109/EuroSPW.2017.43.

9. Binance Academy. (2020). An Introduction to Binance Smart Chain (BSC) [Online]. Accessed: 25/09/2021. Available: https://academy.binance.com/en/articles/an-introduction-to-binance-smart-chain-bsc

10. BSC (Binance Smart Chain) white paper. [Online]. Accessed: 25/09/2021. Available: https://github.com/BSC(Binance Smart Chain)/wiki/wiki/White-Paper.

11. Z. Li, X. Dong, D. Jin, W. Zhong and C. Dai. (2020). Management platform for second-use of retired power battery based on IPv6 and blockchain technology. 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). pp 1465-1469. Available: https://doi.org/10.1109/ITOEC49072.2020.9141687

12. W. Metcalfe. (2020). Ethereum, Smart Contracts, DApps. In Yano M., Dai C., Masuda K., Kishimoto Y. (eds), Blockchain and Crypto Currency. Economics, Law, and Institutions in Asia Pacific. Springer, Singapore. Available: https://doi.org/10.1007/978-981-15-3376-1_5.

13. Khan, S.N., Loukil, F., Ghedira-Guegan, C. et al. Blockchain smart contracts: Applications, challenges, and future trends. Peer-to-Peer Netw. Appl. 14, 2901–2925 (2021). Available: https://doi.org/10.1007/s12083-021-01127-0

14. P. Windley. (2017). Fixing the five problems of internet identity. [Online]. Accessed: 25/09/2021 Available: https://www.windley.com/archives/2017/10/fixing_the_five_problems_of_internet_identity.shtml

15. A. Tobin & D. Reed. (2016) The inevitable rise of self-sovereign identity. The Sovrin Foundation. vol. 29. [Online]. Accessed: 25/09/2021 Available: https://sovrin.org/wp-content/uploads/2017/06/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf

16. N. Naik & P. Jenkins. (2016). A secure mobile cloud identity: Criteria for effective identity and access management standards. 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud 2016). IEEE. Available: https://doi.org/10.1109/MobileCloud.2016.22

17. N. Naik and P. Jenkins, (2017), Securing digital identities in the cloud by selecting an opposite federated identity management from SAML, OAuth and OpenID Connect. 11th International Conference on Research Challenges in Information Science (RCIS). pp. 163–174. IEEE. Available: https://doi.org/10.1109/RCIS.2017.7956534

18. C. Lundkvist, R. Heck, J. Torstensson, Z. Mitton, & M. Sena. (2018). Uport: A platform for self-sovereign identity. [Online]. Accessed: 25/09/2021. Available: https://blockchainlab.com/pdf/uPort whitepaper DRAFT20161020.pdf.

19. Sovrin.org. (2018). Sovrin: A protocol and token for self-sovereign identity and decentralized trust. [Online]. Accessed: 25/09/2021. Available: https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf.

20. C. Allen. (2016). The path to self-sovereign identity. [Online]. Accessed: 25/09/2021. Available: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

21. A. Tobin & D. Reed. (2016). The inevitable rise of self-sovereign identity, Sovrin Found. [Online]. Accessed: 12/16/20. Available: https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf.

22. A. Kassem, J. Sayeed, et al. (2019). DNS-IdM: A Blockchain Identity Management System to Secure Personal Data Sharing in a Network. Applied Sciences. 9, 2953. Available: https://doi.org/10.3390/app9152953

23. H.-Y. Paik, X. Xu, H. D. Bandara, S. U. Lee, S. K. Lo. (2019). Analysis of Data Management in Blockchain-Based Systems: From Architecture to Governance. IEEE Access. pp 1-1. Available: http://dx.doi.org/10.1109/ACCESS.2019.2961404.

24. Sharma, Aakanksha. (2018). Introduction to HTML (Hyper Text Markup Language) - A Review Paper. International Journal of Science and Research (IJSR). 7. 1337-1339.

25. Powers D. (2012) Introducing CSS - the Language of Web Design. In: Beginning CSS3. pp 3-15. Apress, Berkeley, CA. [Online]. Accessed: 25/09/2021. Available: https://doi.org/10.1007/978-1-4302-4474-5_1

26. Guha A., Saftoiu C., Krishnamurthi S. (2010) The Essence of JavaScript. In: D'Hondt T. (eds) ECOOP 2010 – Object-Oriented Programming. pp 126-150. ECOOP 2010. Lecture Notes in Computer Science, vol 6183. Springer, Berlin, Heidelberg. [Online]. Accessed: 25/09/2021. Available: https://doi.org/10.1007/978-3-642-14107-2_7.

27. Fabian Vogelsteller. 2015. web3.js. (2015). [Online]. Accessed: 25/09/2021. Available: https://www.npmjs.com/package/web3.

28. Pereira, C. R. (2016). Building APIs with Node. js. pp 1-3. [Online]. Accessed: 25/09/2021. Available: http://dx.doi.org/10.1007/978-1-4842-2442-7

29. Node.js. [Online]. Accessed: 12/16/20. Available: https://nodejs.org/.

30. Bouzid, Mohamed. (2020). Webpack for Beginners: Your Step-by-Step Guide to Learning Webpack 4. [Online]. Accessed: 25/09/2021. Available: http://dx.doi.org/10.1007/978-1-4842-5896-5.

31. Thakkar M. (2020) Introducing React.js. In: Building React Apps with Server-Side Rendering. pp 41-91. Apress, Berkeley, CA. [Online]. Accessed: 25/09/2021. Available: https://doi.org/10.1007/978-1-4842-5869-9_2.

32. Binance Testnet. [Online]. Accessed: 25/09/2021. Available: [Online]. Accessed: 25/09/2021. Available: http://dx.doi.org/10.1007/978-1-4842-5896-5.

33. Dannen C. (2017) Solidity Programming. In: Introducing Ethereum and Solidity. pp 69-88. Apress, Berkeley, CA. [Online]. Accessed: 25/09/2021. Available: https://doi.org/10.1007/978-1-4842-2535-6_4

34. Truffle Suite Overview. [Online]. Accessed: 25/09/2021. Available: https://www.trufflesuite.com/docs/truffle/overview.

35. Hartel, Pieter & Staalduinen, Mark. (2019). Truffle tests for free - Replaying Ethereum smart contracts for transparency. Accessed: 25/09/2021. Available: https://arxiv.org/pdf/1907.09208.pdf

36. Ganache. [Online]. Accessed: 25/09/2021. Available: https://www.trufflesuite.com/docs/ganache/overview.

37. Git. [Online]. Accessed: 25/09/2021. Available: https://git-scm.com/.

38. Latest releases of Ganache on GitHub. [Online]. Accessed: 25/09/2021. Available: https://github.com/trufflesuite/ganache/releases.

39. MetaMask. [Online]. Accessed: 25/09/2021. Available: https://MetaMask.io/

40. Sublime [Online]. Accessed: 25/09/2021. Available: https://www.sublimetext.com/

41. Truffle Pet-Shop Tutorial. [Online]. Accessed: 25/09/2021. Available: https://www.trufflesuite.com/tutorial

42. Block of Wikipedia in Turkey. [Online]. Accessed 25/09/2021. Available: https://en.wikipedia.org/wiki/Block_of_Wikipedia_in_Turkey

43. IPFS: A peer-to-peer hypermedia protocol designed to make the web faster, safer, and more open. [Online]. Accessed: 25/09/2021. Available: https://ipfs.io/

44. InterPlanetary File System. [Online]. Accessed: 25/09/2021. Available: https://en.wikipedia.org/wiki/InterPlanetary_File_System

45. Benet, Juan. (2014). IPFS - Content Addressed, Versioned, P2P File System. [Online]. Accessed 25/09/2021. Available: https://arxiv.org/pdf/1407.3561.pdf

46. Sin, L., & Lim, W. (2018). Filecoin's Initial coin offering: Using Blockchain to decentralize storage. HBR Store. Available: https://store.hbr.org/product/filecoin-s-initial-coin-offering-using-blockchain-to-decentralise-storage/NTU182

47. Patnaik, S., Wang, T., Shen, T., & Panigrahi, S. K. (2021). Blockchain Technology and Innovations in Business Processes (Ser. 219). Springer. [Online]. Accessed 25/09/2021. Available: https://www.springer.com/gp/book/9789813364691

48. Ethereum and BSC Blockchains—The Best of Both Worlds With pNetwork. (2021). [Online]. Accessed 25/09/2021. Available: https://medium.com/pnetwork/ethereum-and-bsc-blockchains-the-best-of-both-worlds-with-pnetwork-c9d71df8f453.

49. Binance Smart Chain vs. Ethereum: Let's Look at the Stats (2021). [Online]. Accessed 25/09/2021. Available: https://hackernoon.com/binance-smart-chain-vs-ethereum-lets-look-at-the-stats-ow403341?__cf_chl_managed_tk__=pmd_n9qbpfpQd6LaXvStybetdFq5ipvsw9ixRctZJv9224o-1632590114-0-gqNtZGzNAvujcnBszQk9.

50. Binance Smart Chain vs. Ethereum: What's the Difference?. (2021). [Online]. Accessed 25/09/2021. Available: https://academy.binance.com/en/articles/binance-smart-chain-vs-ethereum-what-s-the-difference.

51. Doerk, Adrian. (2020). The growth factors of self-sovereign identity solutions in Europe. Available: http://dx.doi.org/10.6084/m9.figshare.14182586