Ministry of higher education and scientific research

UNIVERSITY OF CARTHAGE

National Engineering School of Carthage

A GRADUATION REPORT SUBMITTED IN THE FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF COMPUTER SCIENCE ENGINEER

In Advanced Information Systems

Mutual Fund Portfolio Management DApp

Host Organization: TALAN TUNISIA CONSULTING

Authored by:
Zied KHAYECHI

Advised by: Mr. Arsalen HAGUI
Overseen by: Dr. Rim MOUSSA

Academic Year: 2018/2019

Dedications

I dedicate this work:

To my dear parents,
Thank you for believing in me. I am grateful for the guidance and the support that I have been receiving from you all these years. I hope I made you proud.

To my sister,
I hope that you find motivation in this work as you move forward with your studies.

To all my friends and everyone that I have worked with,
thank you for sticking with me through thick and thin.

This is but the beginning, yet it is an important milestone in my life that I am blessed to be able to share with you.

Zied KHAYECHI

Acknowledgements

Before presenting this work, It is particularly pleasant to express my deep gratitude to all those who supported me and helped me to accomplish this project.

I would like to express my sincere thanks to Dr. Rim MOUSSA, my advisor at the National School of Engineers of Carthage "ENICarthage" for her kindness, her modesty, her rich experience, her support, her permanent help and the warm welcome she has always given me.

I also thank Mr. Arsalen HAGUI, my technical advisor at Talan Tunisia for welcoming me, for ensuring the smooth running of my internship, for the advice he gave me and the efforts he has provided for the success of this project.

I would like to thank also Ms. Imen AYARI, head of Talan Innovation Factory and the whole Talan Innovation Factory team in particular the Blockchain Team for their valuable advises and their constructive comments.

I would then like to thank all the people who have gravitated around this work, and more particularly the whole staff of Talan, for making this experience as enriching as it is pleasant to live.

Finally, I extend my most sincere thanks to the members of the jury for having honored me by agreeing to evaluate this work.

Contents

List of Figures

List of Tables

Introduction

A mutual fund is a financial intermediary for a collective investment, in which investors put their money in a large pool managed by a professional manager and avoid the risk of picking and managing their own individual securities. The fund manager manage a diversified portfolio by allocating an increasing number of assets and aim at earning a capital income for the investors.

Each mutual fund company keeps track of share ownership in their own asset registrar and manage daily a large volume of transactions. To do this, they are equipped with computer systems covering their own needs as well as their customers needs. These systems must provide a high level of security to cope with any possible attacks. Moreover, In order to protect against known threats, a mutual fund company must ensure that no unauthorized person can access its network. This is what we call, the *Know Your Customer* (*abrev.* KYC). Examples of threats are data theft, money laundering and other malicious activities, which include multiple intermediaries and lead to slow transaction processing.

The financial sector has a considerable weight in the economies of countries around the world and it's neither agile nor fast when embracing new technologies due to its complex system. However, the last innovative technology called *Blockchain Technology* shows premises in transforming the financial services in order to handle them more efficiently. This, obviously, counter usual norms. *Blockchain Technology* brings a new solution for the exchange of the value through the internet and solves the *problem of double spending* of digital currencies by creating a decentralized consensus based on a strong cryptographic system and maintained by network members, i.e. in a peer-to-peer way. This allows the validation and the verification of transaction without a centralized authority.

Blockchain Technology offers a transparent and decentralized platform and has the potential to disrupt multiple industries. Indeed, it allows to perform their processes in a more democratic, secure, transparent, and efficient way. People can trust the digital platform for data exchange and can set up, manage and invest in digital assets as well as trade multiple tokens through a single interface.

Mutual funds companies have all identified *Blockchain Technology* as a disruptive opportunity to revolutionize the *Finance Technology ecosystem* (*abrev.* FinTech). In order to shift to more scalable systems and interconnected world, which will automate the full life-cycle of mutual fund transactions, from order placement through to the settlement and payment process. Thus, many actors established in the financial market, engage in an evaluation process through a proof of concept, to optimize the transactions,

payments and asset registrar within the *blockchain system*, and also analyze the consortium model of governance and rules in order to align the *blockchain system* with the current regulatory and legal framework.

The *Blockchain Team* in *Talan Innovation factory* believes revolving traditional Financial industry into a modern one based on advanced technologies such as *Blockchain Technology*. In this context, our graduation project consists in investigating, designing and developing a *mutual fund platform* based on *Ethereum Blockchain*.

Our graduation report outline is the following,
First chapter *Project Context* presents the host organization, the general project framework, the project requirements and the methodology adopted for carrying out this graduation project.
The second chapter *Overview of Blockchain Technologies* details the blockchain technology and presents a detailed study of the existing blockchain platforms.
The third chapter *Revolving Mutual Fund with Blockchain Technology* introduces the *Mutual Fund Industry* and presents some work-in-progress mutual fund blockchain solutions.
The fourth chapter *Requirements' Analysis* presents the analysis and specification of the system to implement must fulfills. We use Unified Modeling Language to illustrate design artifacts.
The fifth chapter *Design and Technical Study* presents the design of the application and the interactions between the different components identified through a static and dynamic view.
The sixth chapter *Achievements* presents the various implementation steps, the tools used, the test phases and the results obtained through key application interfaces.
$Finally$, We conclude the report and open new work perspectives.

Chapter 1

Project Context

1.1 Introduction

In this chapter, we present the host organization in particular its services as well as its sectors of activity. Then, we focus on some aspects of the project, namely its context, its goals, as well as our contribution and the selected methodology adopted for carrying out this graduation project.

1.2 Presentation of the Hosting Company

The work presented in this report is part of the research and development activities carried out by *Talan Tunisia* -a subsidiary of the *Talan group*.

1.2.1 Overview

For more than 15 years, *Talan* [17] has been known as a major international actor in consultancy and business transformation. *Talan group* headquarter is in Paris and it has offices in Amiens, Rennes, Bordeaux, Lille,Nantes, Lyon, Montpellier and subsidiaries in Casablanca, Geneva, London, Luxembourg, Madrid, Montreal, New York, Singapore, Toronto and Tunis.
Talan expects to reach a revenue of 310 Million Euro in 2019 with more than 3000 consultants.

1.2.2 Talan services

Talan provides its clients with high quality service based on in-depth expertise in business and technology. *Talan* activities are around the following services,

- Consult and Assist project management,
- Support and Implement project transformation,
- Redesign and optimize business processes,
- Design and Execute business transformation,

Talan group acts principally in the sectors of telecommunications, financial, energy and transport and Innovation and new Technologies.

1.2.3 Talan Innovation Factory

Talan Innovation factory is the research and development department at *Talan* Tunisia. It provide business units with expertise and knowledge related to emerging technologies such as Blockchain, Artificial Intelligence, Big Data etc.
Moreover, it experiments innovative methodologies like *design thinking*. Thus, *Talan Innovation Factory* is leveraging the technology to experiment the potential disruption in order to explore new business models, new products and new services in different domains, especially, those of their customers such as finance, telecommunication, insurance, transport and energy.
Furthermore, One of newest investments of *Talan* is the launch of an intern cryptocurrency called *TalanCoin* based on *Blockchain Technology*. *TalanCoin* is initially intended for Talan's employees in order to bring new way of relationship management in more decentralized and collaborative manner.
Our project is mainly related to the *Innovation Factory* and more specifically to the *Blockchain* team.

1.3 Project Overview

In this section, we present the context and motivations of our project, followed by the objectives and a description of the requested work.
Saving Accounts are the most popular option for people, who want to make money savings with high interest. But there are many other options in which people can invest their saving. One of the simplest option is to invest in stock market by managing and buying individual stocks. But, using this strategy investors need to manage by themselves their diversified stocks portfolio.
Mutual Fund is well-known as the most simplest investing alternative used by stock market investors to grow their savings. Thus, with the help of stocks' portfolio managers, investors can invest in a diversified portfolio of stocks which decrease the risk of loosing their funds and increase their returns.
The major issue of *Mutual Funds* is their centralized model that includes multiple intermediaries with cumbersome procedures and leads to less profit.
Figure 1.1 illustrates a typical centralized Mutual Fund industry.

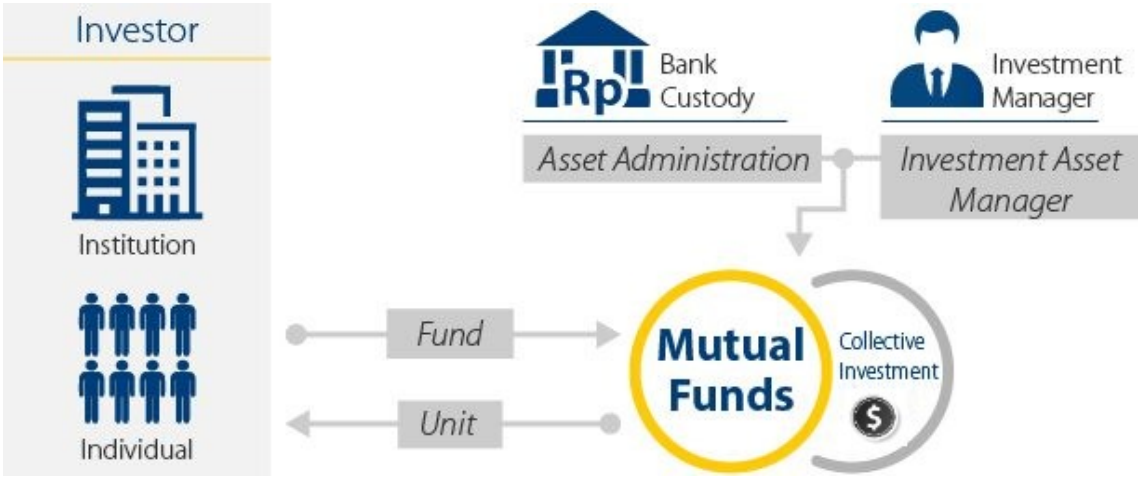

Figure 1.1: Centralized Mutual Fund Industry[1]

Basically, the main purpose of the our project is to design and develop a decentralized model of the mutual fund industry that automates the process of creating mutual funds and tokenized assets through a well regulated *Assets' Management company* and registering of new investors through an automated *Know Your Customer* process and also giving the investors the possibility to purchase or sell their shares at any time based on their current *Net Asset Value per Share* (*abrev.* NAVPS). Throughout this solution, we aim to present the benefits of the *Blockchain* in automating the *Mutual Fund Industry* with the following goals (i) transparency, (ii) risk and latency reduction, (iii) data security as well as (iv) reducing intermediaries by automating their processes with *smart contracts*.

1.4 Development Methodology Adopted

Quality in project is ensured by good practises and recommended processes in conducting a project. We adopt a method in order to follow-up a project progress and define different project phases as well as ensuring a high level of quality and efficiency which is essential criteria in the field of software development.

Methodology

In order to have an organized project process, we adopt a working methodology that formalizes and coordinates the preliminary stages of the development of a system. We *Scrum* -an agile method, and take the advantages of agile methodology. *Scrum* is also adopted by *Talan's Innovation factory team.* The agile method will allow us to implement the whole project starting with a core, exploring spike solutions, divide the work into software functional increments, tracking progress, assessing communication cetera. For each increment the product has to be tested, deployed and integrated with the previous ones. This allows to take into account and process appropriately risks and sudden changes and guarantee that the product will be accepted by the users. Figure 1.2 illustrates incremental software development.

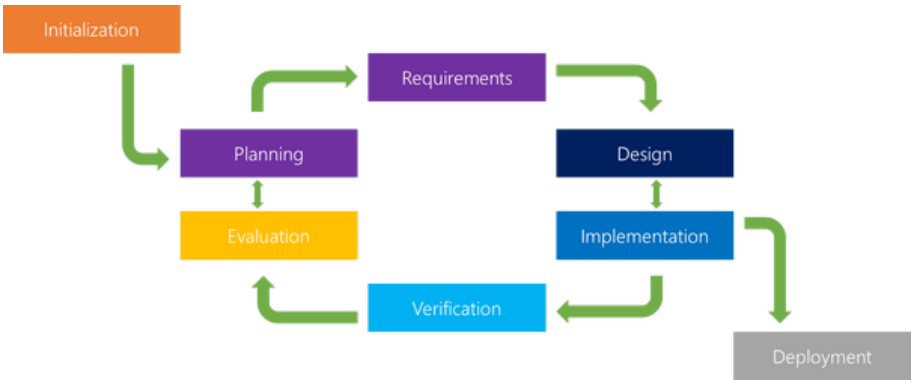

Figure 1.2: Incremental Development Cycle[2]

Formal Modeling

The design of our project allows us to well understand the needs of our project. The Modeling is a formalism used to abstract and simplify the representation of a real-world entity in order to describe and explain it. In our project, we use the *Unified Modeling Language* (*abrev.* UML) to produce graphical artefacts.
UML is a simple graphic design which offers multiple diagrams to represent our project: how it started,

how it works and the different actions that can be performed, what are the interactions, and so. Moreover, it helps to define the various activities of development process and establish a follow-up of the decisions taken all over the design and development process.

Chronology of tasks

The schedule of the various stages of our project are organized according to the internship period as follows,

- Theoretical study of *Mutual Fund industry* and *Blockchain technology*,
- Functional study,
- Requirements' analysis and specification,
- System design,
- Development and Test,
- Report Reduction.

The Gantt chart 1.3 below illustrate the schedule of our project.

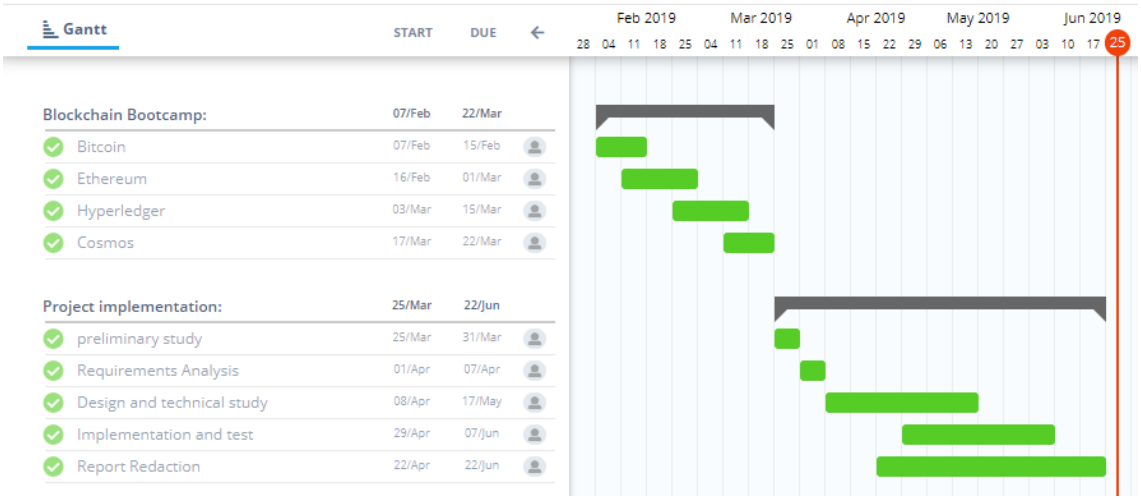

Figure 1.3: Gantt chart

1.5 Conclusion

In this chapter, we present the host company *Talan Tunisie Consulting*. Then, we describe our graduation project and the development methodology followed for its implementation. In the next chapter, we present a preliminary study of the *Blockchain Technology* in order to understand its general concepts and the different Blockchain platforms.

Chapter 2

Overview Of Blockchain Technology

2.1 Introduction

This first part of this chapter presents theoretical basis and key concepts of *Blockchain Technology*. The second part overviews major Blockchain implementations.

2.2 Blockchain Definition

Blockchain is a *Distributed Ledger Technology* (*abrev.* DLT) which uses public register to store encrypted information grouped together and put them into a shared and distributed chained blocks, accessible on a decentralized peer-to-peer network, in secure, transparent and immutable environment.
There are different types of blockchain. We make the difference between permission-less or permissionned access on the hand, and on the other public or private with respect to access to data.
Bitcoin [18] is the first implementation of the blockchain technology, as a digital money ecosystem that provides a fast, secure and open currency. Thus, *bitcoin protocol* allows for the first time the exchange of value through the internet and provides a robust solution that prevents the double spending problem.
The *Bitcoin blockchain* contain cryptographically encrypted transactions recorded in timestamped blocks. These blocks are maintained by the network through consensus algorithms in order to eliminate the need of a central authority and prevent any malicious behaviour.
Figure 2.1 shows how the Blockchain works.

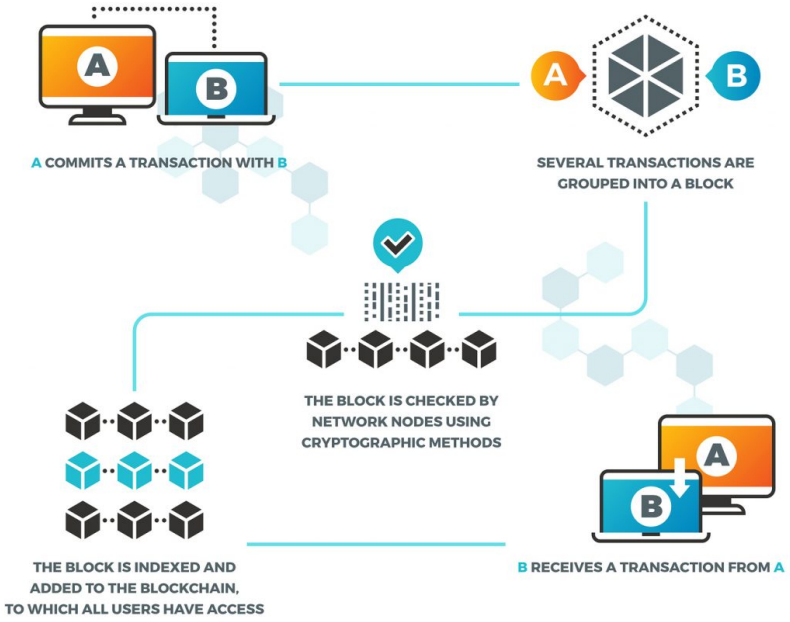

Figure 2.1: A Step-by-Step view of blockchain workflow[3]

2.3 Blockchain Components

Blockchain Technology seems complex at a high level. At a core level, this technology combines powerful well-known comuputer science concepts such as, Decentralized peer to peer network, Distributed shared ledger that records all transactions using advanced cryptographic primitives and mutliple mechanisms for decentralized consensus .
This section discusses each individual main component, namely: *cryptography*, *transaction*, *block*, *ledger*, *peer-to-peer network*, *miner*, *consensus models* and *smart contracts*.

2.3.1 Cryptography

Blockchain Technology uses cryptographic primitives as *cryptographic hash functions*, *digital signatures* and *asymmetric-key cryptography* to create mathematical proofs which increase the security of block header and data in order to create unique identifiers and prevent malicious corruption. In the sequel, we detail cryptographic primitives for blockchain.

Cryptographic Hash Functions

Cryptographic hash functions are methods applied to an input of nearly any size. They calculate a relatively unique output named *digest*. Any smallest change to the input will result in a totaly different output digest. The Secure Hash Algorithm *SHA256* is used in mutliple blockchains implementations. The table 2.1 below shows a simple example.

Input Text	SHA-256 Digest Value
1	6B86B273FF34FCE19D6B804EFF5A3F5747ADA4EAA22F1D49C01E52DDB7875B4B
1.	C977DBE785221D34114D345FDF8FF4E18BB4A11F66750751DBEA29407F9738735F

Table 2.1: Corresponding SHA-256 Digest Values for specific inputs.

Cryptographic hash functions have several important security properties, which are listed below,

- *Preimage resistant:* they are one-way; it is computationally impossible to calculate the correct input value from output value.
- *Second preimage resistant:* Using a given input we cannot find a second input which produces the same output.
- *Collision resistant:* we cannot find two inputs that hash to the same output.

Asymmetric-Key Cryptography

Asymmetric-key cryptography (a.k.a *Public key cryptography*) was introduced by Diffie and Hellman [–Hellman key exchange] to solve the problem of key distribution in a symmetric cryptography system. *Asymmetric key cryptography* uses a pair of keys, i.e. a public key and a private key. The keys mathematically relate to each other but we cannot determine the private key based on knowledge of the public key. The use of two keys can be summarized as follows,

- *Public keys* are accessible and known to everyone used to encrypt messages or to verify signatures.
- *Private keys* are extremely private to their owners used to decrypt messages or create signatures.

Digital Signatures

The primary objective is to sign messages digitally and assure the authentication of message senders by providing multiple security properties such as follows,

- *Authenticity* by using public and private keys to sign and verify messages,
- *Data integrity* because hashes will be different if the message is altered,
- *Non-reputation* since sender cannot deny that they have sent the message.

At a high level, the *Digital Signature* is used as explained in Figure 2.2

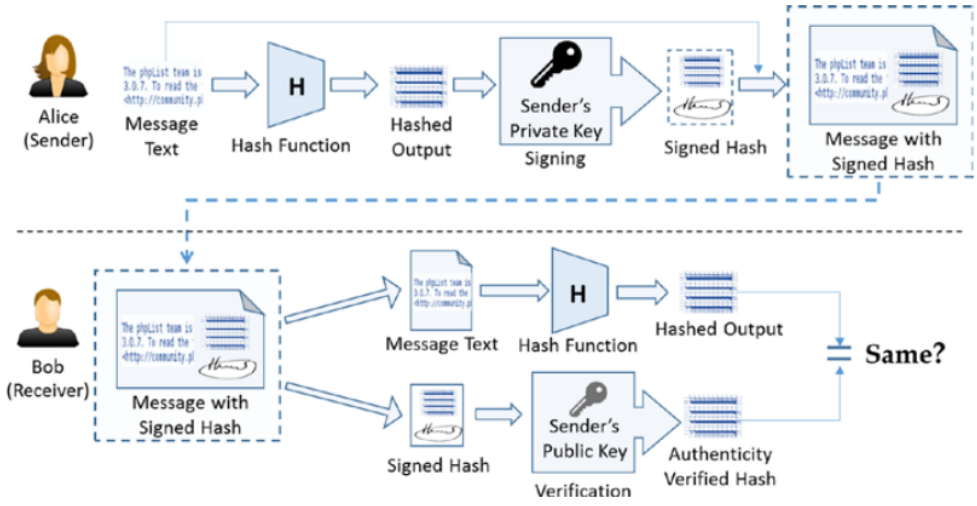

Figure 2.2: Digital Signature Algorithm[4]

2.3.2 Transaction

The *Transaction* is an interaction between users. It is cryptographically signed by private keys of each providers to generate a digital signature. The latter ensures the security in the broadcasting process and prevents any malicious changes during its transmission. Once the transaction is confirmed by the majority of users it will be published in a new block. Notice that,

- Validating a transaction is ensured by checking the formatted manner of transaction and the providers (inputs) has cryptographically signed the transaction in order to verify that they own the private keys.
- Verifying a transaction is processed by comparing the digital signature of transactions and the provider's public keys.

Figure 2.3 below explain the transactions chaining.

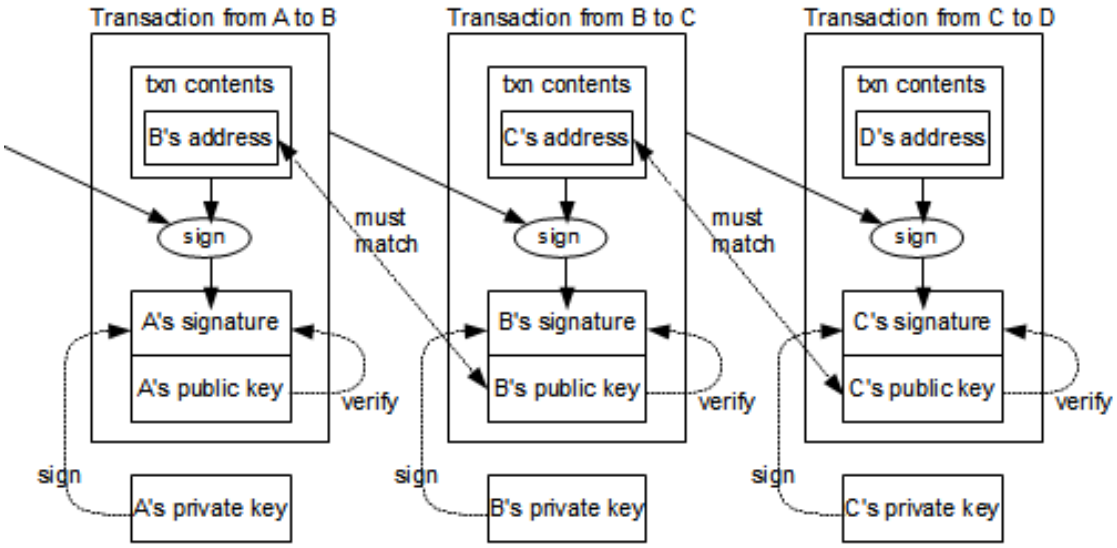

Figure 2.3: Transactions chaining[5]

2.3.3 Ledger

The blockchain distributed design creates many backup copies that are updated and synchronized to the same ledger data between all users which prevent invalid transaction from propagating into the network. Any peer can maintain his own copy of the ledger by requesting a full copy of the blockchain network's ledger. Cryptographic mechanism like hash functions and digital signatures are used to provide tamper resistant ledgers.

2.3.4 Block

The *block* is a data structure added to the blockchain by a publisher user. It contains *header* and *block data*. The *header* contains meta-data related to the *block* and the *block data* contains validated transactions represented in a *Merkle tree* data structure. Figure 2.4 illustrates the block structure.

Once the block is added to the blockchain and chained it with the hash digest of the previous block's header, any modification will be easily detected. This boosts the authenticity and the security of the blockchain network.

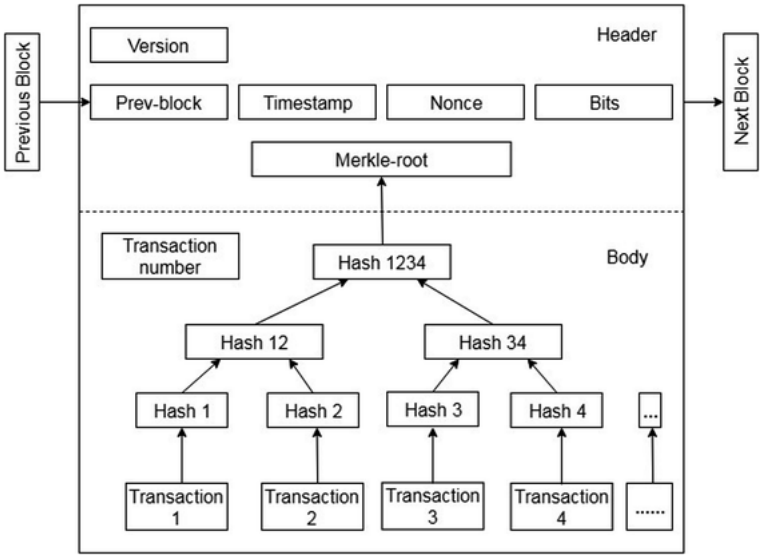

Figure 2.4: Block Structure[6]

2.3.5 Peer-to-Peer system

Peer-to-peer Networks (*abrev.* P2P nets) are types of architectures in distributed systems, [19], which allow connected computers (i.e. nodes) to make their computational resources available to all members of the network, without any point of coordination, in which nodes are suppliers and consumers of resources.
The *Blockchain Technology* uses P2P network in order to turn the user's computers into nodes that makes the whole distributed system. As a result, the more software bringing into the system the more powerful it becomes.

2.3.6 Peers

The blockchain system is equal in terms of authority but peers can have different activities depending on their actual status. *Peers* can be *Full Nodes* or *Lightweight Nodes*.

Full Nodes

Full Nodes have the entire blockchain data which contain all transactions that taken place till now. These nodes have to stay permanently connected to the blockchain network to update their ledgers. So they must have a powerful computing resources which require improvement with time. A *full node* provides multiple activities such as executing the consensus algorithm, broadcasting new blocks, wallet services for lightweight nodes and routing functions that maintain the blockchain network.

Lightweight Nodes

Lightweight Nodes are users running wallet and using the blockchain network to make transactions and verifying them using *Simplified Payment Verification* (*abrev.* SPV). But unlike *full nodes*, they

maintain only the block headers and Merkle branch tree that contain their own *Unspent Transaction Outputs* (*abrev.* UTXO).

2.3.7 Principal Consensus Models

Every Blockchain ledger has a published *genesis block* and others blocks must be added after it based on a decentralized agreement without any trusted party.
The problem of determining the next block that will be added to the blockchain has been solved through different consensus models detailed in the sequel.

Proof of Work Consensus Model

In the *proof-of-work* (*abrev.* POW) model, *full nodes* (a.k.a *miners*) have to find a new block by solving an intensive mathematical problem .
This problem is designed to be hard to solve and easy to check, which enables all nodes to validate or reject any proposed block.
To solve this problem the miners should find a specific hash digest of the block header that should be less than a target difficulty updated dynamically based on the total processing power of connected miners to maintain the average time to broadcast a new block to the network.
In order to get this specific *hash*, the *miner* have to hash the entire *block header* many times by changing a specific number called *nonce* until he/she finds a solution to the problem.
As miners have invested by their computing power and electricity , to generate a new block by solving a computationally intensive problem , the system generate a new coins as a reward in the first transaction of the block called *coinbase transaction*.
This model is used by multiple blockchains such as *Bitcoin*, *Ethereum*, *LiteCoin*, *Zcash*, et cetera.

Proof Of Stake Consensus Model

The basic idea of this model is that the users (a.k.a *Validators*) which have much stake in the system, want more likely the system to succeed and less likely want to subvert it.
The stake is often a cryptocurrency investment that users put it into the system in order to get a determining factor to publish new blocks related to their stake's ratio to the network staked cryptocurrency.
With this consensus, we can determine the next block without any intensive resources in terms of processing power, electricity and time and also ensure the distribution of cryptocurrency among users.
There are four approaches of proof of stake implementation, namely, (i) random selection, (ii) delegate systems, (iii) multi-round voting and (iv) coin aging systems.
The *Ethereum blockchain* plans to move to Proof-of-Stake consensus type: casper (phase zero : 30^th of June 2019).

Byzantine Fault Tolerance

Since the blockchain networks are not controlled by a central authority, malicious attacks try to interrupt the system or propagate incorrect information. Thus, the main idea behind the *Byzantine Fault Tolerance* consensus (*abrev.* BFT), as a solution for the byzantine problem, is ability to continue operating even if some of the nodes fail to communicate or act maliciously.
In other words, each user (i.e. validator) have to maintain the state of the blockchain and share messages between them in order to get the correct state.
Hyperledger and *tendermintCore* consensus are examples of the BFT model.

Comparative Analysis of consensus Models

	PoW	**PoS**	**BFT**
Blockchain Type	Permissionless	Both	permissioned
Trust model	Untrusted	Untrusted	Semi-trusted
Security	51% of computing power needed to attack the network	51% of network worth needed to attack the network	33% of voting right to completely manage the network
Incentive	Block reward + transaction fees	Transaction fees	none
Adversary Tolerance	<=20%	Depend on algorithm	<=33%
Transaction rate	Low	High	High
Token need	yes	yes	no
Scalability	High	High	Low
Performance	Low	Low	High
Energy consumption	very High	Low	Low
Computing Power	very High	Low	Low

Table 2.2: Analysis of consensus Models.

In Table 2.2, we compare different types of consensus, namely PoW, PoS and BFT.

2.3.8 Fork

As blockchain network is distributed and governed by the user's consensus. Thus, any update in the network's protocol or data structures are called *forks.* and can be divided into two categories:

- *Soft fork* in which the changes are compatible with unchanged publishing nodes.
- *Hard fork* in which changes will be rejected by unchanged publishing nodes and create a split in the blockchain network that leads to multiple versions of the blockchain.

The fork term is used also by some Blockchain network to describe temporary ledger conflicts.

2.3.9 Smart contract

A *Smart contract* is a collection of data and code that is deployed through a cryptographically signed transactions and must be deterministic by producing the same outputs based on the same inputs. Smart contract can not only store data and perform calculation but it can publicly change the state and automatically transfer funds. *Oracles* are used by smart contracts to access data from outside the blockchain context. *Nick Szabo* defines smart contract as *A computerized transaction protocol that executes the terms of a contract. The general objectives of smart contract design are to satisfy common contractual conditions, minimize exceptions both malicious and accidental,and minimize the need for trusted intermediaries.* [20]

2.3.10 Types Of Blockchains

Blockchain can be categorized in Two types, based on access to blockchain network (permissionless or permissioned) and based on access to blockchain data (public or private).
A general view of blockchain types is presented in Table 2.3.

	Public Blockchain	Private Blockchain	Federated Blockhain
Access	Anyone	Single organization	Multiple selected organization
Participants	Permissionless Anonymous	Permissioned Known identities	Permissioned Known identities
security	Consensus mechanism proof of work / proof of stake	Pre-approved participants voting/multi-party consensus	Pre-approved Participants Voting/multi-party consensus
Transaction Speed	Slow	Lighter and faster	Lighter and faster

Table 2.3: Blockchain Types.

2.4 Different Blockchains

2.4.1 Bitcoin

Bitcoin was proposed in 2008 by an anonymous called *Satoshi Nakamoto* as a "peer-to-peer electronic cash system" that allows on-line payment between two parties without the need of a financial institutions. Bitcoin protocol proposes a solution for the double spending problem and allows for the first time the exchange of currency in a peer to peer trusted network based on consensus algorithm called Proof of work (PoW).

Bitcoin network makes use of address that are cryptographically hashed from the user's public key and some metadata. These address are used as users identifiers in transaction process .

In order to join the bitcoin network and make transactions, users have to install a *digital wallet* in their devices, which creates automatically private and public keys, account address and QR code for signing and verifying transactions.

The transaction flow of Bitcoin transaction is represented in Figure 2.5. The main steps are,

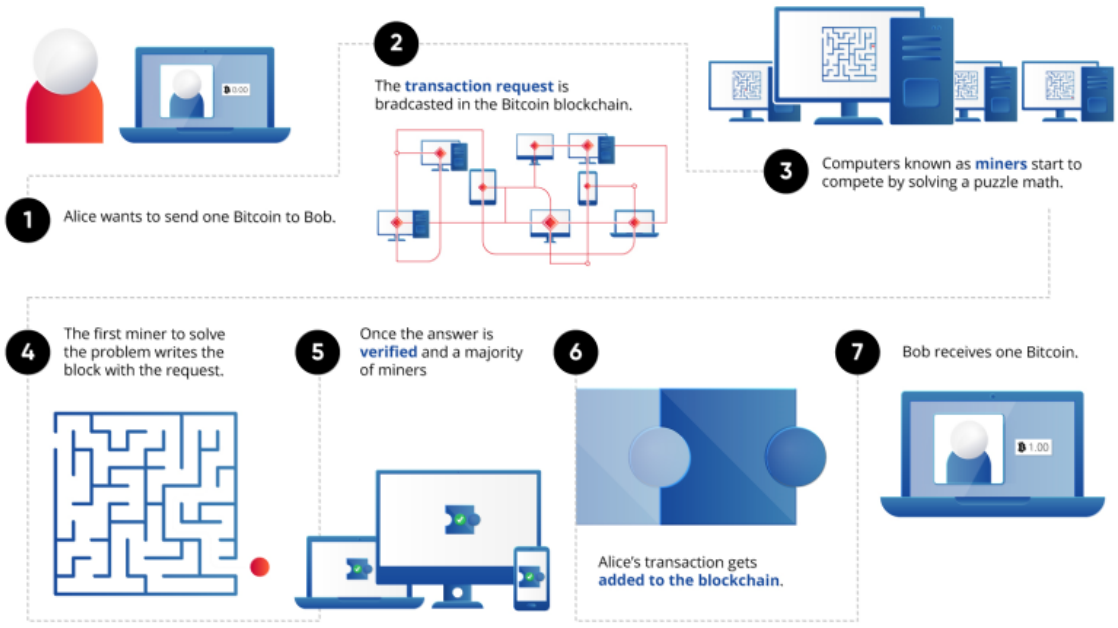

Figure 2.5: Bitcoin Transaction Flow[7]

1. In order to transfer bitcoin, *Alice* has to connect to her wallet thats sign the transaction using her private key. *Alice* has to use an *Unspent Transaction Output* (*U*TXO) as the input of the current transaction and must specify the transaction fee and change.

2. The transaction is broadcasted in the network and added to the *mempool* (i.e. pool of unconfirmed transactions).

3. Connected miners, who are seeking to add new block to the blockchain have to build a proposal block composed of block data (contain a coinbase transaction and validated transactions based on their fees hashed into a merkle tree) and block header (contain merkle root, previous block hash, timestamp, Nonce, version and bits). Then start an intensive computational problem called mining in order to get a nonce that respect the target difficulty specified by the system.

4. The first miner, who solves the problem has to broadcast his block to the network.

5. The majority of miners verify the block by checking the validity of transactions included and the nonce provided.

6. Once verified by the majority, the new block will be added to the ledger and the miner will get the reward.

7. *Bob*'s wallet receives transaction confirmation (*Bob* has to wait for at least 6 confirmations to verify that his transaction was successfully added to the blockchain ledger).

2.4.2 Ethereum

Bitcoin blockchain design allows a sufficient exchange of currency using a simplified script language and monolithic codebase (networking-consensus-application), which limits the development of new currencies and decentralized application to fork the bitcoin codebase or build on top of it.
In 2013, *Vitalik Buterin* a Canadian developer of Russian origin, who though that bitcoin is designed specifically for currency and uses the blockchain as a simple ledger of transactions. He proposes *Ethereum blockchain* that uses the blockchain as a computational platform which can deploy any kind of program by turning the application layer of the blockchain into an *Ethereum Virtual Machine* (*abrev.* EVM).
Using the *Ethereum Virtual Machine*, developers can build any kind of decentralized applications called *Dapps* using smart contract and Turing-complete programming language such as *solidity*, *serpent*, *LLL* and *Vyper* In order to achieve consensus in a decentralized application using smart contract, the execution environment have to be,

- $Deterministic$ since the same output should be found for the same input in different computers.
- $Terminable$ as a given time must be specified for each operation.
- $Isolated$ in order to save the environment from negative effects of any contract.

Figure 2.6 shows the interaction between *Smart contract* and *EVM*.

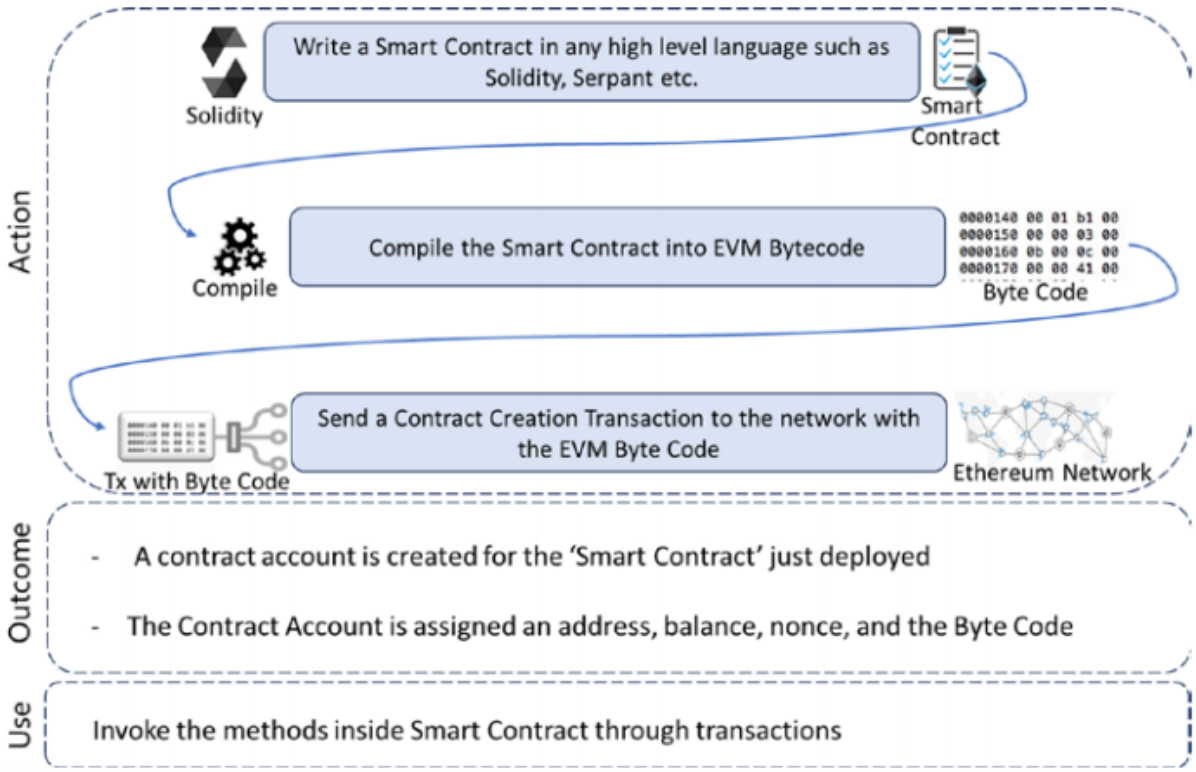

Figure 2.6: Smart contract and EVM interaction[4]

Application Binary Interface (*abrev.* ABI) is used to access the byte code of a smart contract and executes functions which consume cryptocurrency called *Ether* (a.k.a ETH) depending on gas [1] used for each operation of the executed function.

2.4.3 Cosmos

The main issue of *Bitcoin*, *Ethereum* and other blockchains is the scalability and interoperability since the number of transactions is limited and there is no direct transaction between two different blockchains. *Cosmos project* solves these issues by creating an *Internet of blockchains* [?], in which blockchains can communicate with each other in a decentralized manner. These new concept is achieved through several open source tools such as *Tendermint*, *IBC* and *Cosmos SDK* detailed below.

Tendermint BFT

Ethereum project allows developers to create customized decentralized application on top of Ethereum network using the Ethereum virtual machine as an applicatioin layer. However, the creation of new blockchain using the EVM is difficult since the EVM stack is hard to be customized or fork from.

Cosmos project facilitates the process of development of new blockchains, using the *Tendermint BFT* as a generic engine that packages the networking and consensus layers of a blockchain based on a byzantine fault tolerant (BFT).

[1] Unit of measurement of operations calculating in the Ethereum network.

Tendermint BFT allow developers to focus on application development using socket protocol called the *Application Blockchain Interface* (*abrev.* ABCI) which can be wrapped in multiple programming language.

Cosmos SDK

Cosmos SDK is a modular framework used to build secure ABCI application using *Tendermint BFT*. *Cosmos SDK* provides developer with a set of modules that can be imported in any complex application. Figure 2.7 shows the *cosmos SDK* with some modules on top of the *Tendermint core*.

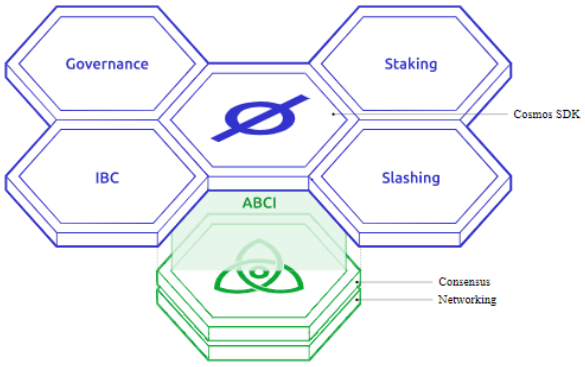

Figure 2.7: Cosmos SDK on top of Tendermint core[8]

IBC

Inter-Blockchain Communication protocol allows heterogeneous chains to transact value or data to each other. *Cosmos* uses a modular architecture of *Hubs* and *Zones*, where *zones* are different blockchains and *hubs* are blockchains designed to connect zones. when a zone connects to a hub via an IBC connection, it will have automatically access to all zones connected to the same hub. *Cosmos hub* (see Figure 2.8) was launched on 13^th of March 2019, as the first hub in the cosmos network. The *cosmos hub* is the first public *proof-of-stake* blockchain with a native token called *ATOM* used for transaction fees. *Cosmos* can connect with any kind of blockchain using direct *IBC* for any fast-finality chains that uses the *POS* as a concensus, or special proxy-chain called peg-zone for any probabilistic-finality chain that uses other consensus like *PoW*.

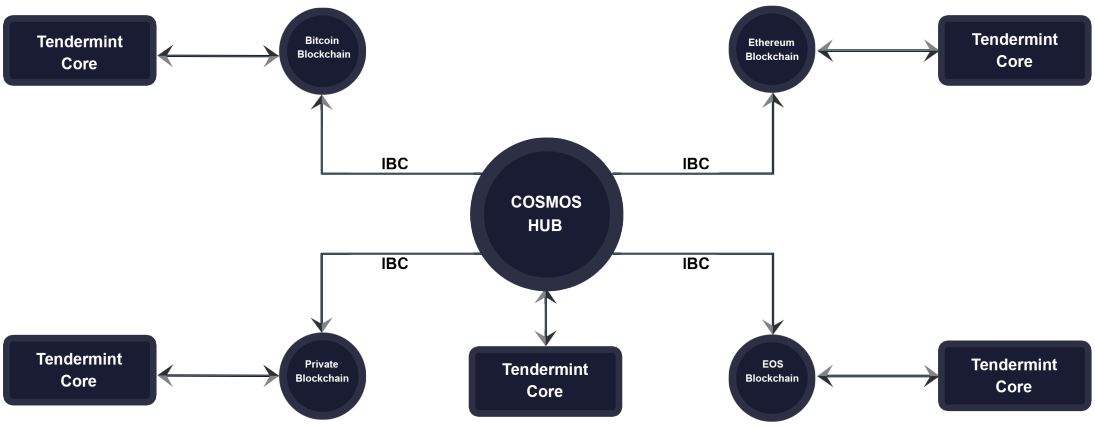

Figure 2.8: Cosmos hub architecture[9]

2.4.4 Hyperledger

Hyperledger is an open-sourced project founded in 2015, by the Linux Foundation and multiple leaders in different sectors such as finance, banking, supply chains, manufacturing and others.
Hyperledger is extremely different from other blockchains. While *Ethereum* and *Bitcoin* have their own blockchains and their own cryptocurrencies, *Hyperledger* does not maintain its own blockchain and cryptocurrency . Public blockchains allows anyone to join the network. This is not desirable for big entreprise like banks, who deal with customers data. Also, public blockchains are slow. Thus, *Hyperledger* allows companies to create their own private and permissioned blockchains.
Hperldger Fabric framework is the most important project in Hyperledger family, since it provides a *Membership Services Provider* (*abrev.* MSP) that manages the authentication of participants on the network using a specific user ID, Modular consensus and Channels that allows the creation of separate ledger of transactions. The transaction flow of Hyperledger transaction is illustrated in Figure 2.9. Main steps are,

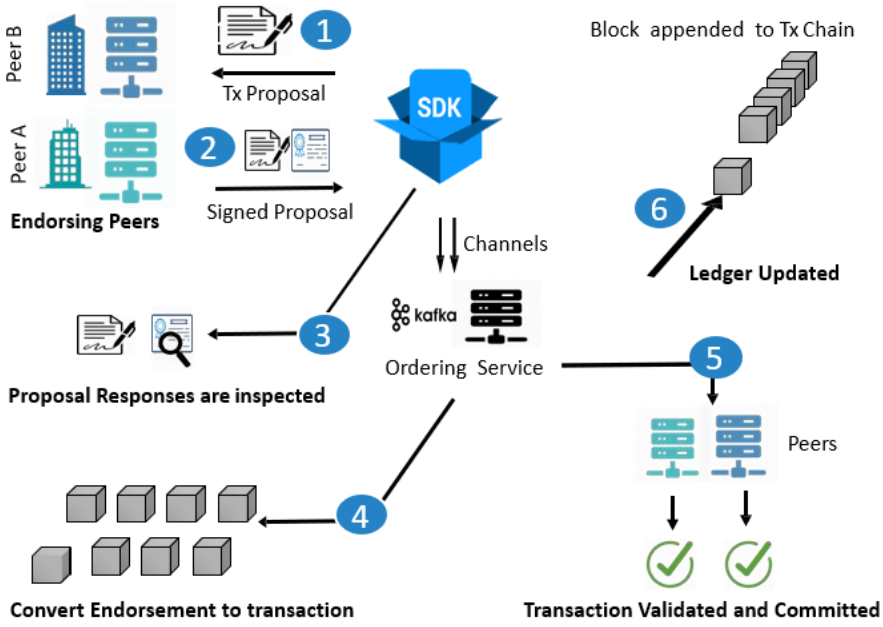

Figure 2.9: Hyperledger Transaction Flow

1. Transaction proposal packaged by the SDK application using the user's cryptographic credentials to get a unique transaction signature. (Endorsement Policy in the Chaincode state proposal have to be accepted by all parties).

2. The Endorsing Peers Verify that the proposal is well formed, does not already submitted, a valid MSP has generate the signature and the submitter is authorized to perform operation in the channel.Then,The Endorsing Peers execute the transaction proposal without updating the current ledger and send the response value, read set, and write set to the submitter.

3. The *SDK application* checks the signature of endorsing peers of the proposal responses.

4. The *SDK application* broadcasts transaction message to the Ordering Service. The latter contains the read/write sets, the endorsement signatures and Channel ID. The Ordering Service orders transactions from all the channels of the network chronologically by Channel and creates blocks of transactions per channel.

5. The blocks of transactions are delivered to peers with the same channel ID. Based on Endorsement policy the peers validate transactions then the block will be tagged as valid or not.

6. The Peer appends the Block to the Channels chain and for each VALID transactions the write-set is committed to the World State Database. Then a notification is emitted to the client application.

2.4.5 Comparing Different Blockchain Platforms

In Table 2.4, we compare different blockchain platforms, namely *Bitcoin*, textitEthereum, *Cosmos* and *Hyperledger*.

	Bitcoin	**Ethereum**	**Cosmos**	**Hyperledger**
Aim	create a cryptocurrency that allows the exchange of value in trusted and decentralized system	create a global decentralized supercomputer	Become the internet of all blockchains	Allows companies to create their own permissioned blockchain
Consensus	PoW proof of work	PoW moving to Casper PoS	TendermintBFT based PoS	PBFT practical byzantine Fault Tolerance
Distributed applications	limited to the fork of bitcoin codebase	using smart contract (solidity and soon incorporate Vyper)	using cosmos SDK to create blockchain (zone)	using chaincode in different languages
Token	BTC (bitcoin)	ETH (ether)	ATOM (atom)	-

Table 2.4: Comparing Different Blockchains

2.5 Conclusion

In this chapter, we have presented in detail the basic concepts as well as the different technologies required to conduct our project. This study will be used as a support for our system specification requirements. The next chapter will be devoted to study the impact of *Blockchain Technology* on the *Mutual Fund industry*.

Chapter 3

Revolving Mutual Fund with Blockchain Technology

3.1 Introduction

This chapter presents a theoretical study of *Mutual Fund ecosystem* in which we focus on major Mutual funds types and management strategies as well as key concepts. Then, we show that traditional management of centralized mutual fund structure is out-of-date and should be revised. Hence, we propose a software application solution based on *Blockchain Technology* to boost efficiency and performance in *mutual fund ecosystem.*

3.2 Mutual Fund

Mutual fund is the best solution for small or individual investors, who search for a professionally managed portfolio in order to get a trade-off between risk and return. Mutual fund pool investors money into a fund managed by a fund manager, who invests in various assets like equities, stocks, bonds and other securities. This gives investors an important diversification with a small investment. Mutual fund investors get a part of ownership of the mutual fund company's assets called share and participate proportionally in the gains or losses of the fund.

3.2.1 The Basics Concepts of Mutual Fund

The mutual fund company value depends on the performance of the securities it invests in. Thus, investors are buying shares that represent the performance of the fund manager portfolio . The value of a mutual fund share is referred to the *Net Asset Value Per Share* (*abrev.* NAVPS). The latter is derived from the division of total Net Asset Value (*abrev.* NAV) by the total amount of shares.

$$NAVPS = \frac{NAV}{Shares\ Outstanding} \tag{3.1}$$

Mutual Fund modus operandi

Depending on the objective of the fund, the fund manager investment vary and may include stocks, bonds, options, currencies, treasuries and money market securities. Figure 3.1 below shows the

interaction between Mutual fund actors. Notice that,

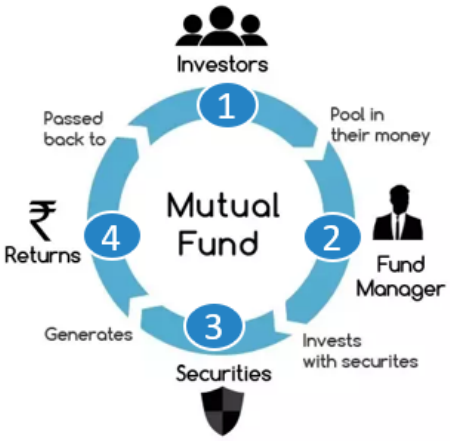

Figure 3.1: Mutual Fund work-flow[10]

1. Individual investors put their money into a fund pool,
2. A fund manager manage the fund portfolio,
3. Managers use their knowledge and experience to manage the portfolio and invest in different securities depending on the fund objective,
4. Investment return will be generated to shareholders proportionally to their contribution.

Mutual Fund Fees

There are two categories of Mutual Fund fees,

1. $Operating\ fees$: is an annual percentage of funds under management that usually rate between 1% and 3%.
2. $Shareholder\ fees$: fees of purchasing and selling funds that are paid directly by investors.

3.2.2 Types of Mutual Funds

Mutual funds can be classified by structure, objective based on funds characteristics. Figure 3.2 shows the different Mutual fund types. The principal funds types are *open-end fund*, *closed-end fund* and *Exchange Trade Fund*.

Open-end Fund

Open-end mutual fund shares are issued and redeemed on demand based on the *net asset value* of assets under management. The NAV is generally updated at the close of each trading day. Investors can buy or sell shares directly from a fund. Fund manager charge fees for asset management.

closed-end Fund

Closed-end mutual funds have a fixed number of shares opened to the public investors at an initial offering. Investors have no transaction with the fund and they trade Shares on an exchange just like stocks. Shares' prices are determined according to supply and demand and they are usually traded at a wide discount or premium to their NAVs.

Exchange Trade Fund

Exchange Trade Fund (*abrev.* ETF) combine the advantage of issuing and redeeming shares on demand closely to their net asset value (NAV) from the open-end Fund and the trading of shares like stocks from the close-end Fund. Instead of opening shares publicly, ETF are open to institution that buy a huge number of shares in order to trade them like stocks to individuals investors, which not only decrease the number of transactions between the fund and investors but also the fund management fee. This fund management type allows investors to get shares with premiums and discounts that are usually within 1% of NAV.

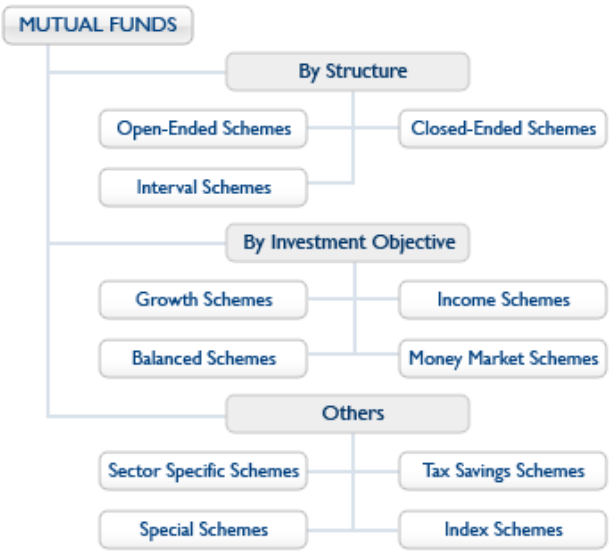

Figure 3.2: Types of MutualFunds[11]

3.2.3 Investment Strategies

Investment strategies are used to generate portfolio return. there are two main strategies: *active management* and *passive management*.

Active Portfolio Management

Active management approach needs a *fund manager* who purchases or sales investments. This decision making investment depends on the experience and expertise of portfolio manager in order to boost the potential for returns. Active investing are usually beating the market trends, economy and political changes.

Passive portfolio management

Passive management fund called also *Index funds*, are tracking the returns of a market index or benchmark as closely as possible. This is easy to understand and offer a relatively safe approach to investing in broad segments of the market.
Since active management managers are beating the market, they have to manage the market risks and human error especially in terms of securities selection. Indexing eliminates this. *Index funds* are also less frequently traded, which means that they have far lower management fees than actively managed portfolio.

3.2.4 Centralized Mutual Fund Structure

Centralized mutual fund companies keep track of their share ownership on their own asset register. These approach are complex, manually processed, prone to errors, paper-based and orders can take days to go through multiple intermediaries. Figure 3.3 presents the different Mutual Fund stakeholders. *The Fund Sponsor* is the promoter, who establishes and registers it with a securities and exchange board. *The Fund manager* is the portfolio manager and acts to the interest of investors.
The Asset Management Company (*abrev.* AMC) is like a Manager that acts under the Supervision and direction of trusted parties and have to be approved by securities and exchange board. AMC manages the investment schemes and makes the required disclosures to investors like NAV calculation.
The Registrar is the responsible for safe keeping of Mutual fund securities.

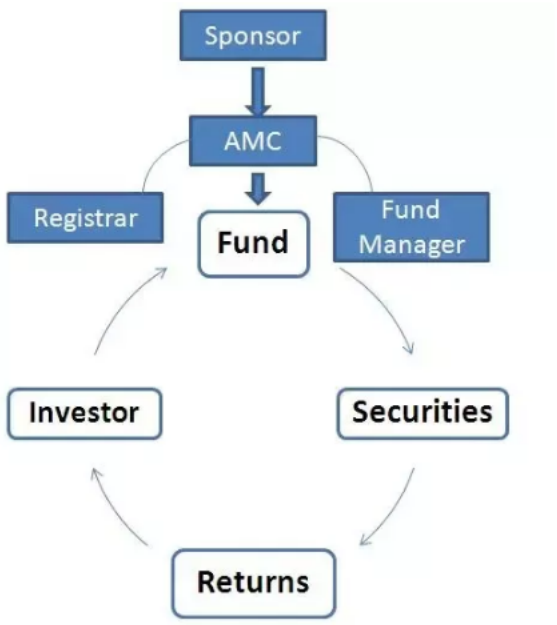

Figure 3.3: Mutual Funds structure[12]

Advantages And Disadvantages Of Mutual Fund Industry

The Mutual Funds are a common choice for many investors because of multiple features offered, that have attract investors to invest their money in such industry. However there is no perfect asset, and Mutual funds have some drawbacks too. Table 3.1 below shows the advantages and disadvantages of Mutual Fund.

Advantages	**Disadvantages**
Investment diversification	Centralized : Time lost in processing transactions
low-risk and high-reward	Fluctuating Returns
Professional Management	Investors have no control of stocks
Explicit investment goals	Charge fees (active management)
Flexibility and Liquidity	Heavy Cash Supplies

Table 3.1: Advantages And Disadvantages of Mutual Fund.

3.3 Implementing Blockchain technology in Mutual Fund Ecosystem

The centralized environment is one of the biggest challenges in the Mutual Fund industry, because of the cost of maintaining digital infrastructure and uptimes. Further, The overall investing process is slow and time consuming. For example, there is an important time lag from the closure of stock market for the computation of *Net Asset Value* (*abrev.* NAV).
So, there is a crucial need to move from manual process of funds trading to a "*FinTech*" industry that uses new technologies to automate the finance infrastructure. This Mutual Fund industry transformation have to automate the global funds markets by improving the fund management, accelerate the transaction cycle, increase transparency, ensure asset security and reduce management fees.
Blockchain have a direct impact on businesses, particularly on the financial services and banking. Thus, the blockchain technology can improve the Mutual Fund industry with its decentralization, transparency, tamper-resistance, privacy and accountability. All will allow to save time and cost of transaction processing.
Smart contracts can be used to ensure that any update of information is available for all on the blockchain and automate transaction processing. Multiple types of tokens can help us also for digital assets in the Mutual Fund industry.

3.3.1 Overview of the Related Solutions

Multiple Mutual fund companies are looking to implement the blockchain technology into their current system to benefit from the exciting features offered by this technology.
The majority of these projects are still studying the proof of concept of this integration or under development. From these projects we can state *Calastone* as a global funds network based in London which announced that it will switched their fund trading system into a clearing service using its Blockchain *Distributed Market Infrastructure*(*abrev.* DMI). This migration will affect over 1,800 calastone's customer in 41 markets, which will represents the largest financial community connected and transacting through a distributed ledger technology.
Julien Hammerson, Calastone's Chief Executive Officer, comments; "*Through leveraging blockchain technology, the DMI transforms the way in which funds are traded, enabling an investment management community that can meet the changing needs of investors.*"
This blockchain based platform and others are in progress of their implementation,in this context,our project is to study,design and develop a demonstration of a mutual fund platform using the blockchain technology.

3.3.2 Proposed solution

Blockchain technology offers the Mutual fund industry an opportunity to implement innovative management approach.Typically, Fund managers use a third party to record the shareholders and realize complex identification process such as *Know Your Customer*(*abrev.* KYC) and *Anti Money Laundering*(*abrev.* AML), which makes it difficult to manage a large number of shareholders. while blockchain technology simplifies all this process because of its immutable, permanent and independent ledger, as well as the use of smart contracts to automate complex process and adding more effective features such as decentralized management through a *Decentralized Autonomous Organization*(*abrev.* DAO) between

the investors and their fund managers. Instead of using traditional asset management in which assets are administrated separately, tokenized assets can be used in the mutual fund industry in order to make all the assets accessible and efficiently managed by any mutual fund manger from one only account. This will simplify and reduce the cost of the asset management and maximize investors' returns. Figure 3.4 presents a decentralized structure of the mutual fund industry.

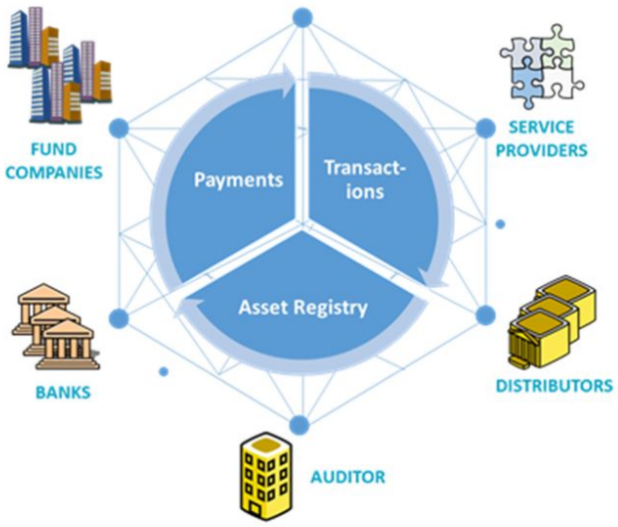

Figure 3.4: Decentralized Mutual Funds structure[13]

So, a decentralized model of the mutual fund industry will improve the management efficiency, increase data transparency and the transaction processing and also the asset security and safety.

selected Platform

After studying the various types of mutual funds and the different blockchain platforms. we need to select a mutual fund type and a blockchain platform that fits our project's requirements,

- Investors have to interact directly with their funds.
- We need Private blockchain to test our solution and then Public blockchain to publish it.
- Our business logic should be automated via Smart contracts.
- The support of Crypto-currencies in order to transfer crypto-assets.
- Well documentation is needed to understand the development technique of our project.

In order to respect all these criteria, we have selected the *open-end fund investment* and the *Ethereum blockchain* to develop our own proof of concept of implementing the blockchain technology in the mutual fund system.

3.4 Conclusion

In this chapter, we present the mutual fund Industry, Followed by the need of implementing the blockchain in the mutual fund and finally, an explanation of our choice for the Open-end fund and the Ethereum platform. In the next chapter, we will cover requirements' analysis and specification's details.

Chapter 4

Requirements Analysis

4.1 Introduction

After presenting the general context of our project, we focus on the analysis and specification of the requirements of our application. Requirements analysis is a crucial stage in the development cycle of any information system. It allows to to determine the main features to be implemented. Next, we start by identifying the actors of our platform. We will then enumerate the functional requirements and the non-functional requirements that our application must fulfills. All design artefacts presented in this chapter are illustrated using Unified Modeling Language.

4.2 Identification of actors

An *actor* specifies a role played by a user or other external entity which interacts directly with the system. In our application actors fall into *main actors* category or *secondary actors* category.

4.2.1 Main actors

- **Mutual Fund Manager** : After Registration, the *Mutual Fund Manager* can interact with the system and take advantage of managing the mutual fund shares in a decentralized manner.
- **Investor** : After Registration, this *Investor* can check the existing mutual fund and can choose to purchase or withdraw multiple mutual fund shares in real-time.

4.2.2 Secondary actors

- **Blockchain** : This actor holds the *smart contract*, creates the transactions and distributes them on different nodes of the network.
- **Miner** : This actor confirm and validate the transactions made in our system using a well known consensus protocol.

4.3 Requirements Types and Analysis

In this section, we present the different functional and non-functional requirements to implement.

4.3.1 Functional requirements

These features are the result of several meetings with the *Product owner*. Our solution, at the end, must imperatively meet the following functional requirements:

- **Create Mutual Fund Token** : Mutual fund manager have to create their own and only Mutual-fund Token(*abrev.* MF Token) by providing some customized information (such as name, symbol, decimals). This allows our system to create a specific token and gives permission for the manager to register in the Mutual fund.
- **Register Mutual fund** : After creating his/her MF Token, a manager has to provide other information such as NAV value, withdraw/purchase fees, et cetera. Thus, to successfully publish the mutual fund to the public investors.
- **Update NAV** : One of the main features of our decentralized application is the simulation of an open-end mutual fund in which shares price are updated by the fund manager through the update of net asset value (*abrev.* NAV) of asset under management.
- **Consult Dashboard** : The dashboard should contain all mutual fund registered in the system in order to help the investors to purchase shares.
- **Purchase/withdraw Mutual fund shares**: Investors can purchase and withdraw shares of any mutual fund registered in the system. These requests are ensured by a *smart contract* deployed in the blockchain network.
- **Consult Portfolio** : Users can consult their own portfolio. For the manager the portfolio must contain a mutual fund related information and for the investor the portfolio must contain the owned shares.

4.3.2 Non-Functional requirements

Apart from these functional requirements, our system must respect the technical constraints related to the general operations. Basically, our system must respond to the following non-functional requirements:

- **Speed** : The system must be efficient and guarantee a very low latency. In order to get a fast and transparent access to the data.
- **Security** : The system must be protected against any modification or deletion attacks. For the Blockchain technology, this part is already secured by protocols and cryptographic algorithms.
- **Ergonomics** : The system must be easy to operate by any simple user and not necessarily experienced user finance. That's why the application interface must be simple, easily manipulated and user-friendly.
- **Maintainability and scalability** : Our system must allow easy maintenance and must scale as needed. Thus, this feature is provided by the composition of our source code on files and the use of coding best practices.

4.4 Requirements' specification

In the following section, we go into the description of the requirements previously specified by modeling them using *use case diagrams* and *system sequence diagrams* with a description of the associated scenarios. This is to clarify the designed solution by identifying chronology of the interactions between the system proposed and the different actors.

4.4.1 Product backlog and sprints identification

The Product Backlog is the most important artefact for Scrum and a key part of Scrum's process.
In the product backlog, presented in Table 4.1, we specify the product features, represented by user stories, ranked by priority and assigned estimates in days. Thus, from the product Backlog we can identify the different sprints.

Sprint	ID	Story	Estimation	Priority
1	1	As a fund manager I want to be able to create Fund Token.	7	1
	2	As a fund manager I want to be able to register a new mutual fund.	7	2
	3	As a fund manager I want to be able to check portfolio and update the related net asset value.	7	3
2	4	As an investor I want to be able to purchase mutual fund shares.	7	4
	5	As an investor I want to be able to withdraw mutual fund shares.	7	5
	6	As an investor I want to be able to check my portfolio and transfer fund shares.	7	6

Table 4.1: Product backlog

4.4.2 General requirements' specification

In this part, we describe the general system through a diagram of the overall use case presented in Figure 4.1.
We note that the functional requirement "*Create Mutual Fund*" and " *Mutual Fund Shares Management*" requires the intervention of the secondary actor *Miner* in order to validate transaction information submitted by the users. On the other hand, the rest of functional requirements require the verification of user address generated by the blockchain. If successful, the users get access to all features offered by the system.

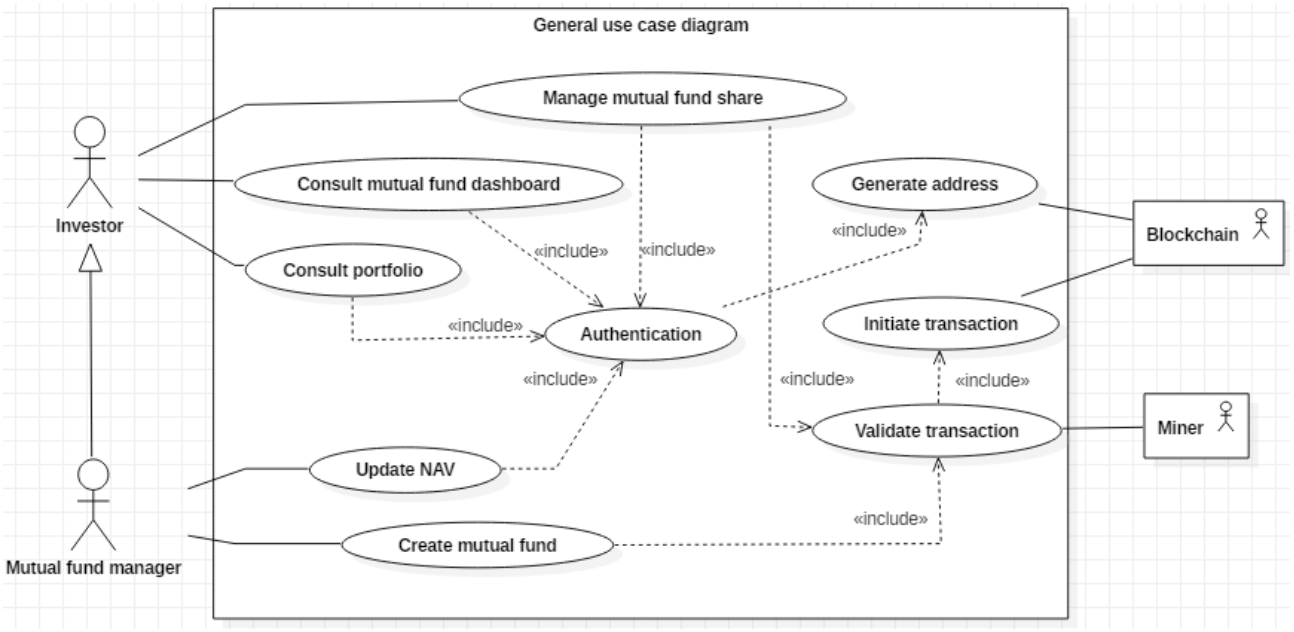

Figure 4.1: General Use case Diagram.

4.4.3 Detailed specification

After presenting our *general use case diagram*, we detail the most important use cases, namely *Create Mutual Fund* and *Mutual Fund Shares Management* by providing a refinement to the use case diagram.

Use Case: *Create Mutual Fund*

(a) *Refinement of "Create Mutual Fund" use case* : the use case diagram, illustrated in figure 4.5, describes the different use cases included and features accessible by the *Mutual Fund Manager* in order to create a Mutual Fund.

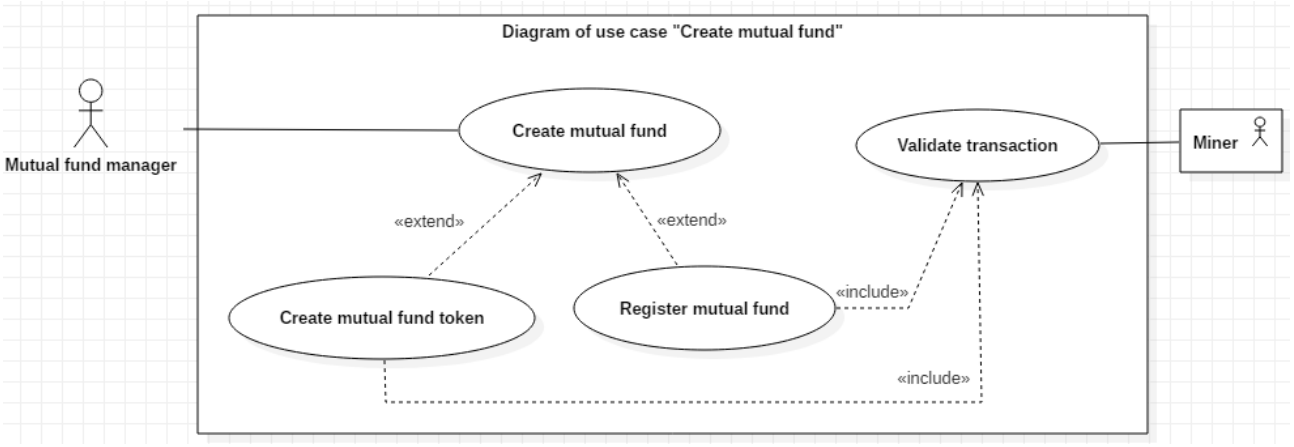

Figure 4.2: Create Mutual Fund Use case Diagram.

(b) Text description of "Create Mutual Fund" use case: next, we describe the scenarios of different use cases related to the creation of mutual fund and indicate investigated restrictions.

- Text description of "Create Mutual Fund Token" use case: Table 4.2 present the possible scenarios in relation to the creation mutual fund token use case.

Title	Create Mutual Fund Token
Purpose	Allows mutual fund manager to create a unique MFToken
Actors	Mutual Fund Manager
Precondition	The manager must be authenticated via account address and the connection must be established
Post condition	The manager is an MFToken holder
Nominal scenario	N1. The manager accesses the MF token creation interface N2. The system displays the interface *Create MF token* . N3. The manager enters the information of the MFToken to be created. N4. The system prepares this operation. N5. The system submits the transaction proposal to the miners. N6. The miner validates the transaction. N7. The system receives a validation response of the transaction. N8. The system returns a message of success of mutual fund token creation.
Alternative scenario	A1 : The manager has already an MFToken **1.** The system indicates that this account has already an MFToken. **2.** The nominal scenario is repeated in point 3.
Exceptional scenario	E1 : No internet connection • The system displays an alert to the manager. E2 : Ethereum account not detected. • The system informs the manager to use a web3 provider (metamask)

Table 4.2: Text description of *Create Mutual Fund Token* use case.

- Text description of "Register Mutual Fund" use case: Table 4.4.3 present the possible scenarios in relation to the registration mutual fund use case.

Title	Register Mutual Fund
Purpose	Allows manager to register a new Mutual Fund Using a holded MFToken
Actors	Mutual Fund Manager
Precondition	The manager must be the MFToken holder and must not be the manager of an existing mutual fund
Post condition	Mutual Fund Registered
Nominal scenario	N1. The manager accesses the Mutual Fund registration interface. N2. The system displays the interface *Register Mutual Fund.* N3. The manager enters the information of the Mutual Fund for registration.

Nominal scenario	N4. The system prepares this operation. N5. The system submits the transaction proposal to the miners. N6. The miner validates the transaction. N7. The system receives a validation response of the transaction. N8. The system returns a message of success of mutual fund registration.
Alternative scenario	A1 : The manager has not an MFToken **1.** The system indicates that the manager is not the MFToken owner. **2.** The nominal scenario is repeated in point 3. A1 : The manager has already an existing Mutual Fund **1.** The system indicates that the user is already a mutual fund manager. **2.** The nominal scenario is repeated in point 3.
Exceptional scenario	E1 : No internet connection • The system displays an alert to the manager.
Exceptional scenario	E2 : Ethereum account not detected. • The system informs the manager to use a web3 provider (metamask)

Table 4.3: Text description of *Register Mutual Fund* use case.

(c) System sequence diagram of "Create Mutual Fund" use case : in this part we present the sequence diagrams related to the creation of Mutual Fund.

- Use Case Sequence Diagram "Create Mutual Fund Token": the creation of Mutual Fund Token follows the sequencing illustrated in Figure 4.3

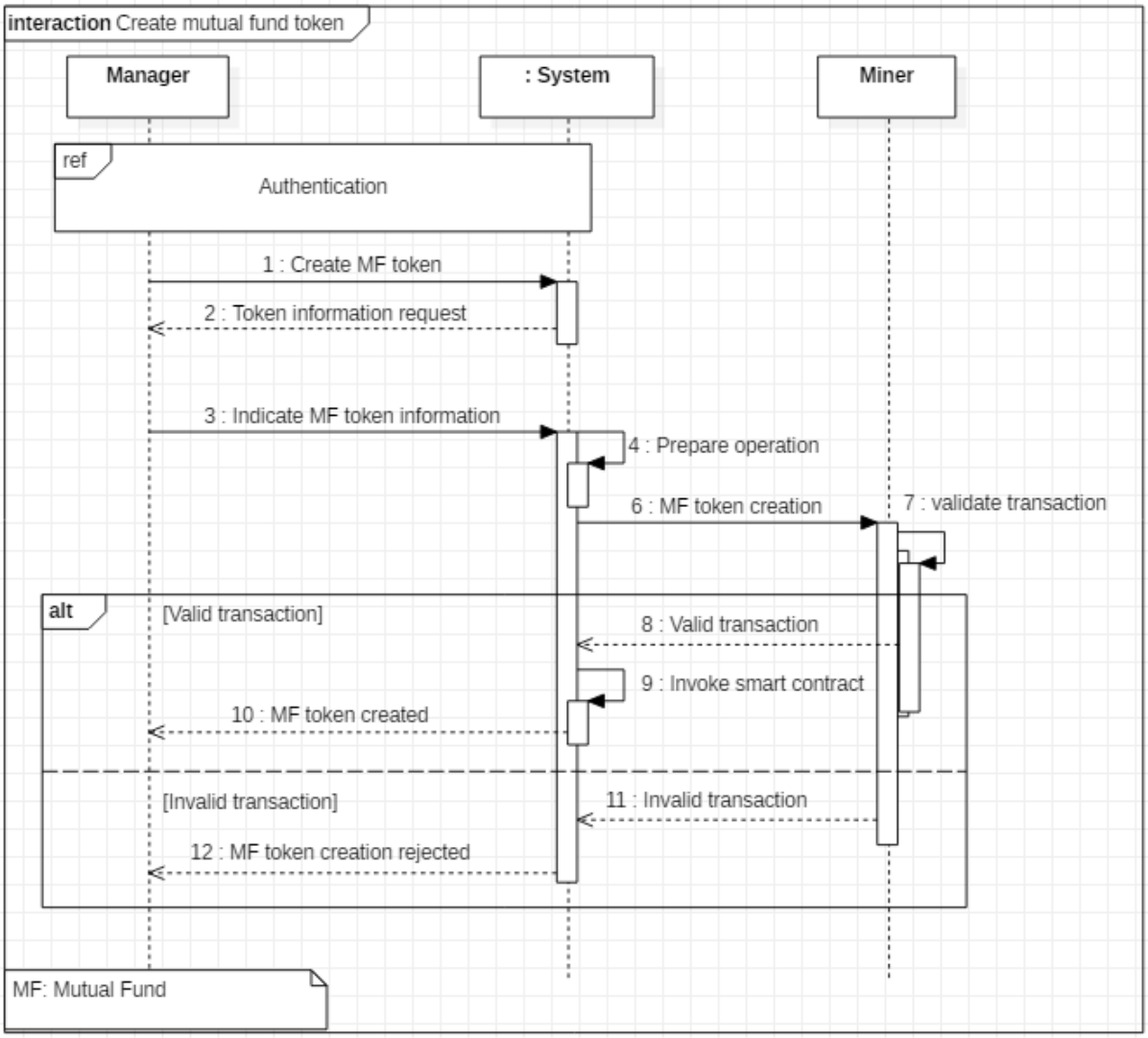

Figure 4.3: *Create Mutual Fund Token* Sequence Diagram.

- Use Case Sequence Diagram "Register Mutual Fund": the registration of Mutual Fund follows the sequencing illustrated in Figure 4.4.

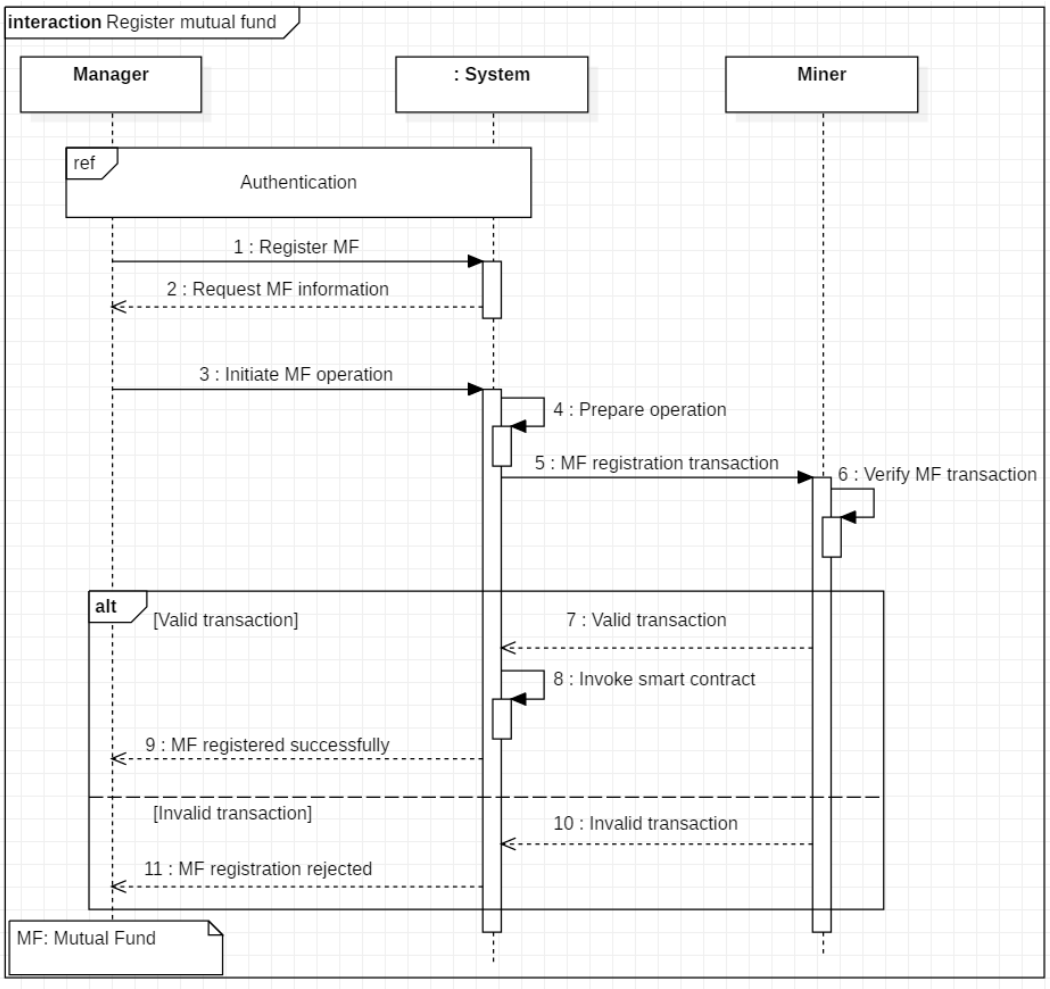

Figure 4.4: *Register Mutual Fund* Sequence Diagram.

Use Case: *Manage mutual fund share*

(a) *Refinement of "Manage mutual fund share" use case* : the use case diagram, illustrated in Figure 4.5, describes the different use cases included and features accessible by the *investor* actor in order to purchase or withdraw Mutual Fund shares.

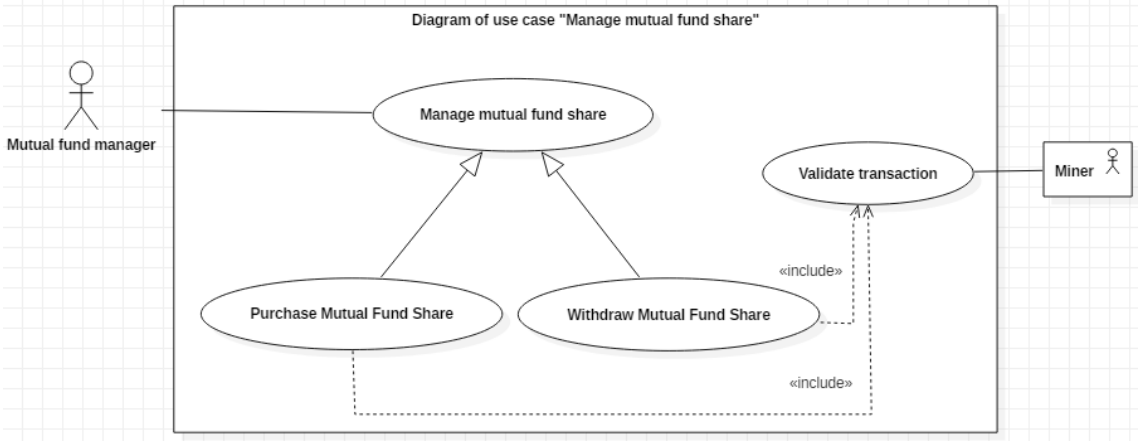

Figure 4.5: *Manage mutual fund share* Use case Diagram.

(b) Text description of "Manage mutual fund share" use case: in this part we describe the scenarios of different use cases related to the management of mutual fund shares and indicate all known restrictions.

- Text description of "Purchase Mutual Fund Share" use case: Table 4.4.3 present the possible scenarios in relation to the purchase Mutual Fund Share use case.

Title	Purchase Mutual Fund Share
Purpose	Allows Investors to purchase Mutual Fund Shares
Actors	Investor
Precondition	The user must be authenticated by his address and the connection must be established The value to purchase must be greater than one share value
Post condition	Mutual Fund Share purchased
Nominal scenario	N1. The investor accesses the application dashbord interface. N2. The system displays the interface *Dashbord*. N3. The investor selects a mutual fund to get more information. N4. The system displays the mutual fund information. N5. The investor enters the information of the Mutual Fund and the value to purchase with. N6. The system prepares this operation by taking the necessary value based on the current NAV value. N7. The system submits the transaction proposal to the miners. N8. The miner validates the transaction. N9. The system receives a validation response of the transaction. N10. The system returns a message of success of purchasing mutual fund share.
Alternative scenario	A1 : The purchased value is less than one share value **1.** The system indicates that investor must invest more to get MF Shares. **2.** The nominal scenario is repeated in point 3.
Exceptional scenario	E1 : No internet connection • The system displays an alert to the manager. E2 : Ethereum account not detected. • The system informs the manager to use a web3 provider (metamask)

Table 4.4: Text description of *Purchase Mutual Fund Share"* use case

- Text description of "Withdraw Mutual Fund Share" use case: Table 4.4.3 present the possible scenarios in relation to the withdraw Mutual Fund Share use case.

Title	Withdraw Mutual Fund Share
Purpose	Allows Investors to withdraw Mutual Fund Shares
Actors	Investor
Precondition	The user must be authenticated by his address and the connection must be established The investor must hold greater or equal than shares number to withdraw
Post condition	Mutual Fund Share withdrawed
Nominal scenario	N1. The investor accesses the his portfolio interface. N2. The system displays the interface *Portfolio*. N3. The investor selects a mutual fund to get more information. N4. The system displays the mutual fund information. N5. The investor selects Withdraw and chooses the shares number. N6. The system prepares this operation by checking the share number and get returned value based on the current NAV value. N7. The system submits the transaction proposal to the miners. N8. The miner validates transaction. N9. The system receives a validation response of the transaction. N10. The system returns a message of success of Withdrawing mutual fund share.
Alternative scenario	A1 : The Withdrawed shares amount is less than holded ones. **1.** The system indicates that investor does not hold the indicated shares amount. **2.** The nominal scenario is repeated in point 3.
Exceptional scenario	E1 : No internet connection ● The system displays an alert to the manager. E2 : Ethereum account not detected. ● The system informs the manager to use a web3 provider (metamask)

Table 4.5: Text description of *Withdraw Mutual Fund Share* use case.

(c) System sequence diagram of "Manage mutual fund share" use case : in this part we present the sequence diagrams related to the Mutual Fund Shares Management.

- Use Case Sequence Diagram "Purchase Mutual Fund Share": the purchase of Mutual Fund shares follows the sequencing illustrated in Figure 4.6.

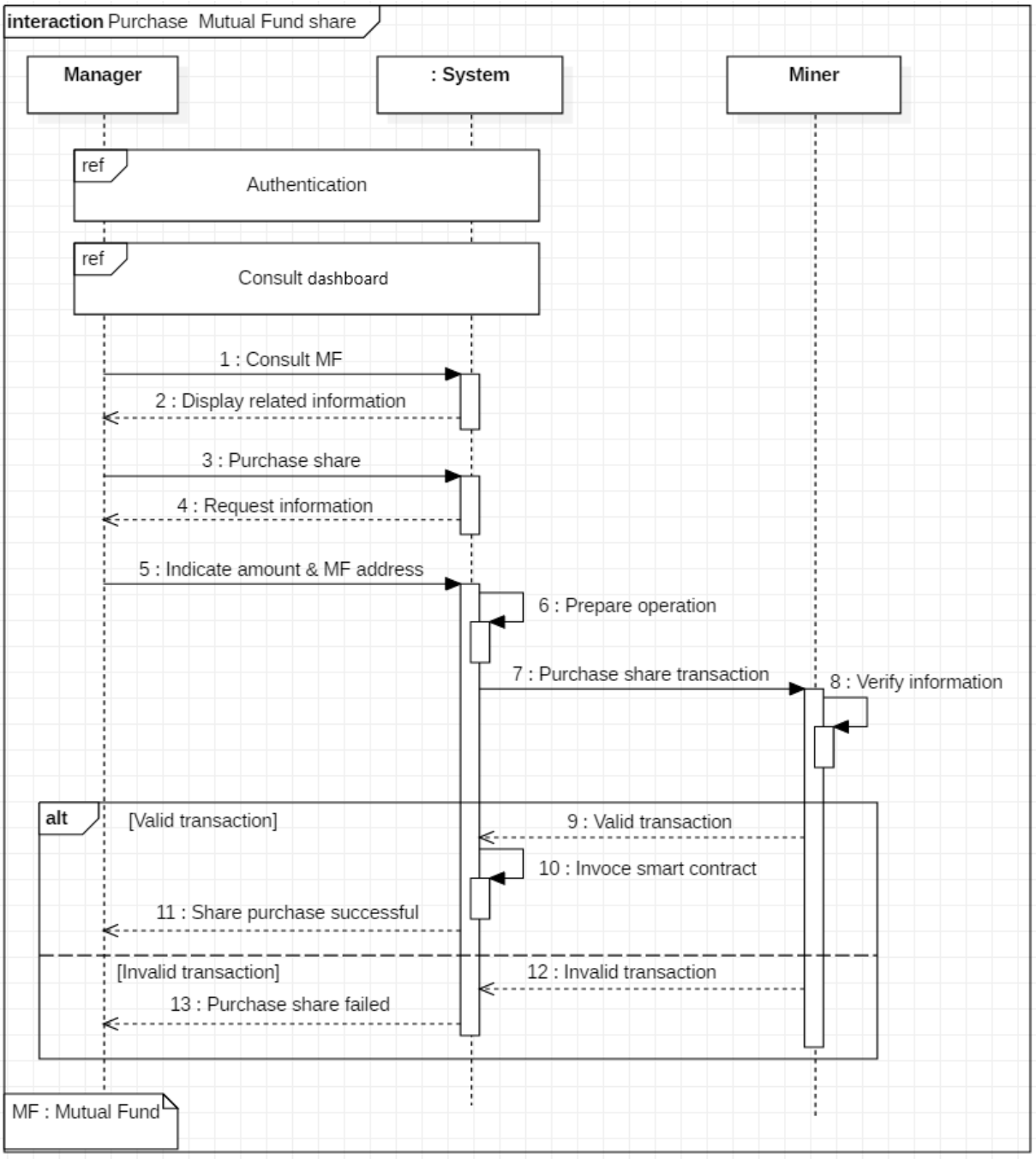

Figure 4.6: *Purchase Mutual Fund Share* Sequence Diagram.

- Use Case Sequence Diagram "Withdraw Mutual Fund Share": the withdraw of Mutual Fund shares follows the sequencing illustrated in Figure 4.7

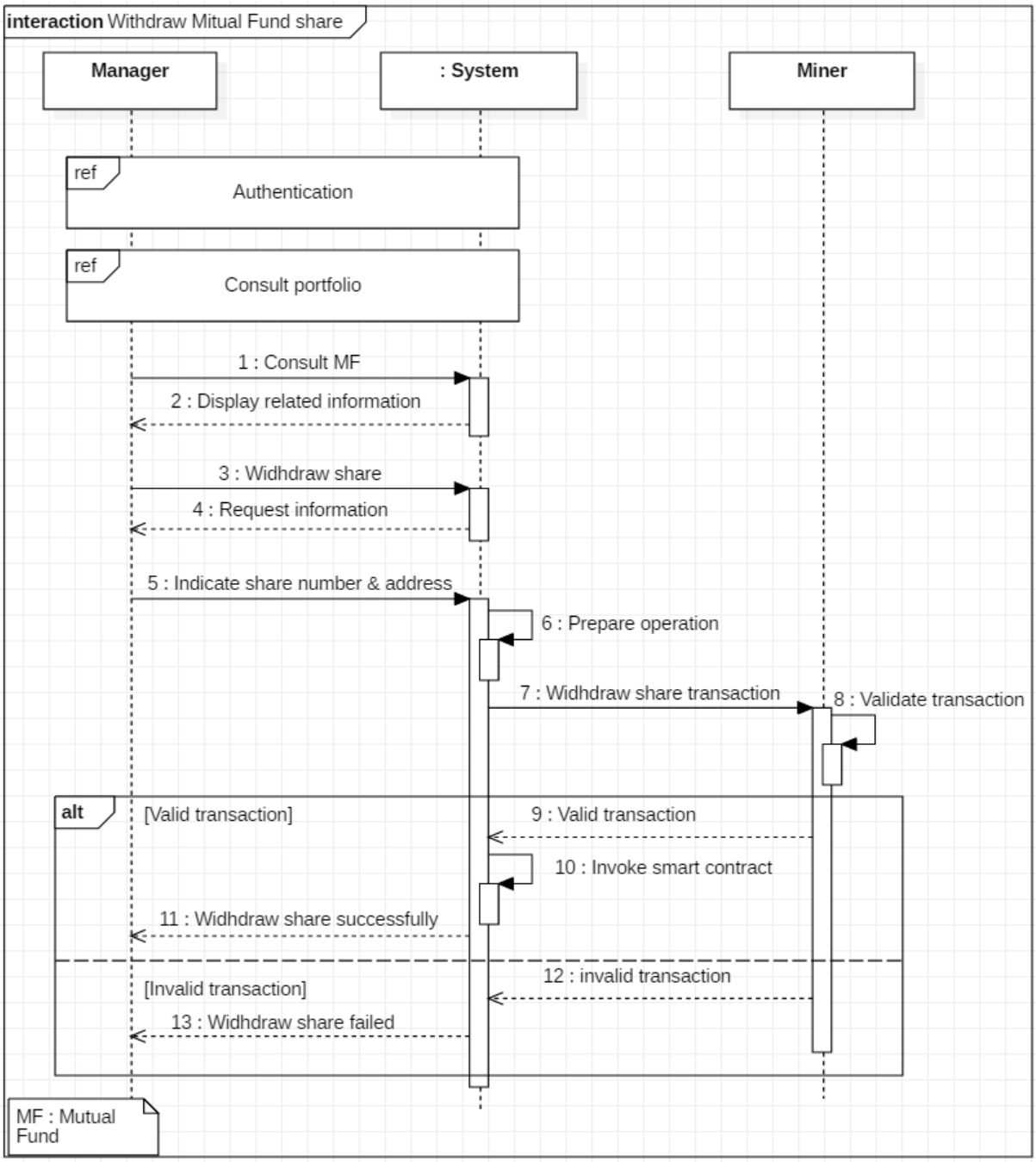

Figure 4.7: Withdraw Mutual Fund Share Sequence Diagram.

4.5 Conclusion

In this chapter, we identified the goals of our project. This phase generates a net view of the fundamental features of our system. This vision is the basic support for the design of our solution that we start in the following chapter.

Chapter 5

Design And Technical Study

5.1 Introduction

In this chapter, we continue with the design phase of our system. We start with a presentation of the overall project architecture, by showing the global architecture and the architectural patterns established. Then, we present our detailed design through a structural diagram to better explain the features of our solution.

5.2 System architecture

Our system must provide a marge of extensibility and scalability for each component. In order to respect these requirements, we choose to follow a modular aspect.

5.2.1 Global System Architecture

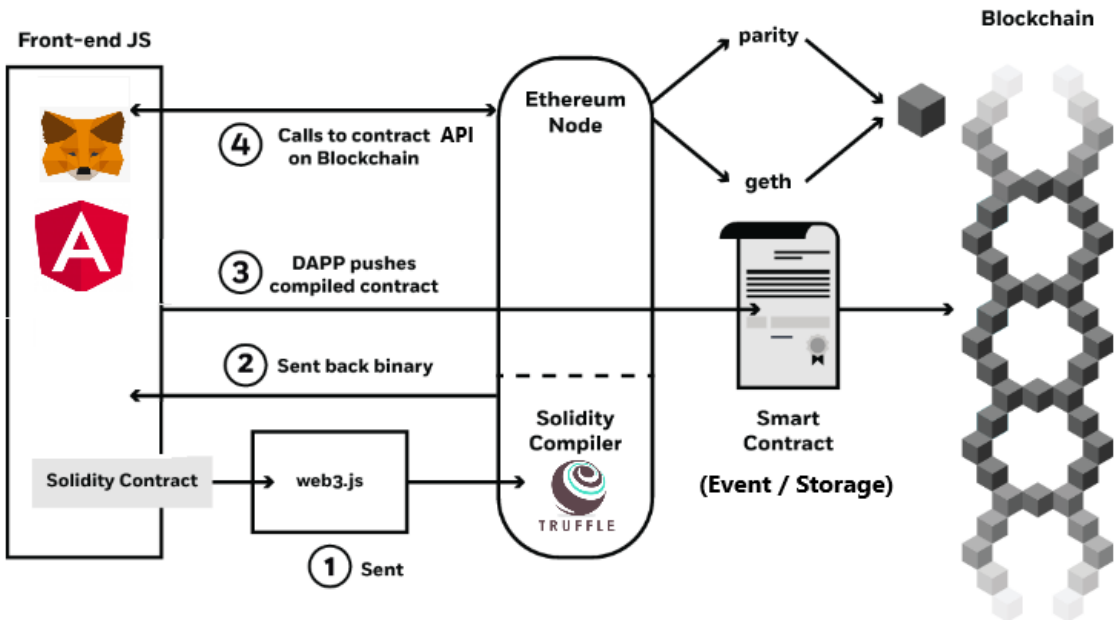

Figure 5.1: Global System Architecture[14]

Before going into the details of the system design phase, it is essential to present the architecture that encompasses the previous chapter's needs. This architecture as illustrated in Figure 5.1 is essentially composed of two parts :

- **Front-end** : This part contains the different interfaces with which the users can interact. These interfaces are developed with *Angular7* and other web technologies.
  In order to invoke *Ethereum network API*, we have to use client libraries such as *web3.js* and connect to an *Ethereum node*. We may run nodes ourselves or connect to existing one via bridge/proxy using browser plugin such as *Metamask*.

- **Smart Contract** : This is the crucial part for our decentralized application which provides communication with the Blockchain network in order to benefit from the blockchain revolutionary feature of no single point of failure/truth. Thus, we no longer need for any "server" to run as a back-end and secure our application against tampering, which can fail and make our application unavailable. So, once our *smart contracts* are deployed they will run on the blockchain. Notice that the decentralized consensus used by distributed network of participants is secure by design since no data modification could happen without the global agreement. The *smart contract*, running in the Ethereum network,

  - Must be written using a turing-complete language like *Solidity* and then deployed to the network.
  - *Miners* have to run their *Ethereum Virtual Machines* and handle *contract API calls*.

5.2.2 Design pattern

The architecture of this part is offered by *Angular framework*, that is based on the separation of view, logic and data access and the use of binding principle and event. This architecture named as *Model-View-View Model* (*abrev.* MVVM) illustrated in Figure 5.2 and is defined as a framework for organizing views in relation to models and data access.

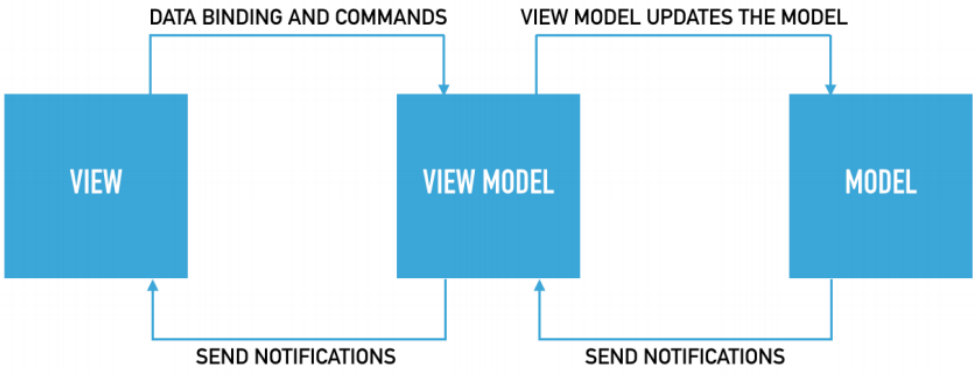

Figure 5.2: MVVM architecture[15]

The roles of each entity is presented below

- **The model**: refers to the data access layer and all related manipulation methods.

- **The View**: refers to the visual layer that is coupled to the data through Data Binding to invokes related View Model methods.
- **The View Model** : refers to models and their properties and connected to the view through Data Binding, In order to ensure a bidirectional communication between the model and the view.

5.2.3 Detailed architecture

Throughout this part we illustrate in detail the Front-end architecture as well as our smart contract architecture.

Front-end Architecture

The *Angular framework* has a scalable and robust architecture and also a modular application thanks to *Angular modules* (*abrev. NgModule*). Thus, we present in Figure 5.3 our *front-end* architecture.

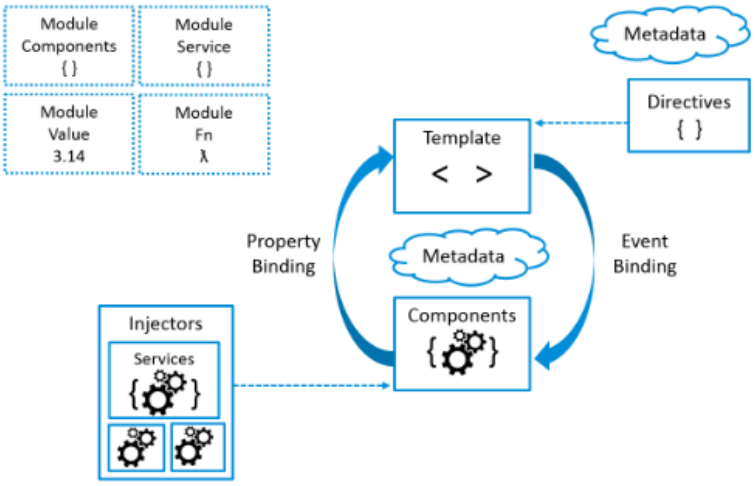

Figure 5.3: Front-end Architecture[16]

The architecture presented previously includes the following elements:

- *Module*: *Angular* defines *NgModule* which ensures the compilation mechanism for a group of components. The *root module*, named *AppModule* is responsible for the launch of the application through a boot mechanism, and *Angular* provides other functional modules that establish a closely related features.
- *Component*: It defines a class associated with data, application logic and view through an HTML template. Each component depends on modules and/or services that have to be configured through a dependency injection mechanism.
- *Template*: It defines the view of each component using HTML and other Angular directives.
- *Service*: Service is not specific to any view, it contains tasks that are injected through dependency injection mechanism.
- *Routing*: It defines the hierarchies of the views and application state through a service that defines the navigation path with which the users can interact and redirect to the corresponding view.

- *Data-binding*: It defines the communication between the template and its component.
- *Directives*: they define the dynamics of the application by transforming the DOM using their instructions. There are two types of directives structural and attribute where structural directives alter elements in DOM and attribute directives alter the appearance of element.

Smart Contract Architecture

Smart contracts are deployed and engaged with users with no going back, which means that there is no altering allowed in the contract. In case of mistake or we need an upgrade, we have to re-deploy a new smart contract version and move all the users to the newest version released of the contract. The problem is that through the last operations, we consume a high gas price and lose any data already stored in the old contract. Thus, it's important to focus on code efficiency and the smart contract design.

- **Smart Contract Design** : In order to get a well smart contract design, we have to separate as much functionality as we can into multiple contracts. In our project we use the design illustrated in Figure 5.4. Notice that we separate business logic contract, data model contract and main controller contract that run on the whole application.

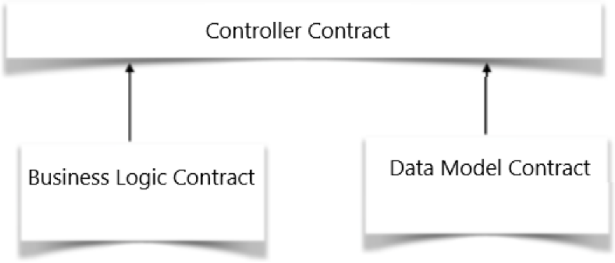

Figure 5.4: Smart Contract Architecture.

Using the previous design, we get the smart contract architecture described in Figure 5.5. In which we have the *proxy* contract as a controller that can be used to update or add any smart contract. The *DMF Storage* and *FundToken Storage* contracts represent our data model and *DMF, FundToken, safeMath and ERC20* contracts that contain our business logic.

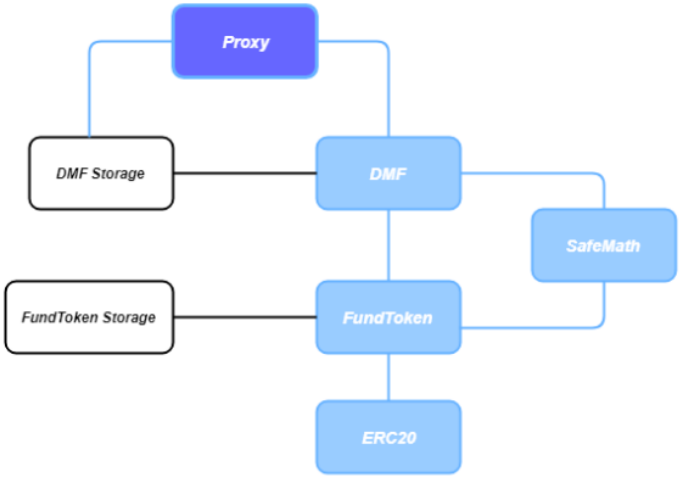

Figure 5.5: Contract Design.

- **Smart Contract Storage** : Storing on blockchain consume a high gas price, so contract design needs to have a lot of considerations of what is stored and used types. Another feature that we can used is called *Event Sourcing*. In the latter, events are used as the main source of truth about the current state, since they are cheaper and can carry data that can remains stored in the blockchain forever and queried through the client application in order to build their internal state.

5.3 Detailed Design

In this section, we present our project design. we use an Agile Software Engineering Design Method dedicated for Blockchain Applications [21], which is submitted and accepted at Software Engineering Conference Russia (SECR 2018), as well as the UML language as a modeling language to clarify our system modules.

5.3.1 Smart Contract Structure Diagram

Class diagrams can be used to model the smart contracts and represent the structure as well as the relationships between them. We introduce various stereotypes in this kind of diagram such as «contract», «struct», «library» and «map».
The class diagram presented in Figure 5.6 shows the structure of our main smart contract DMF.

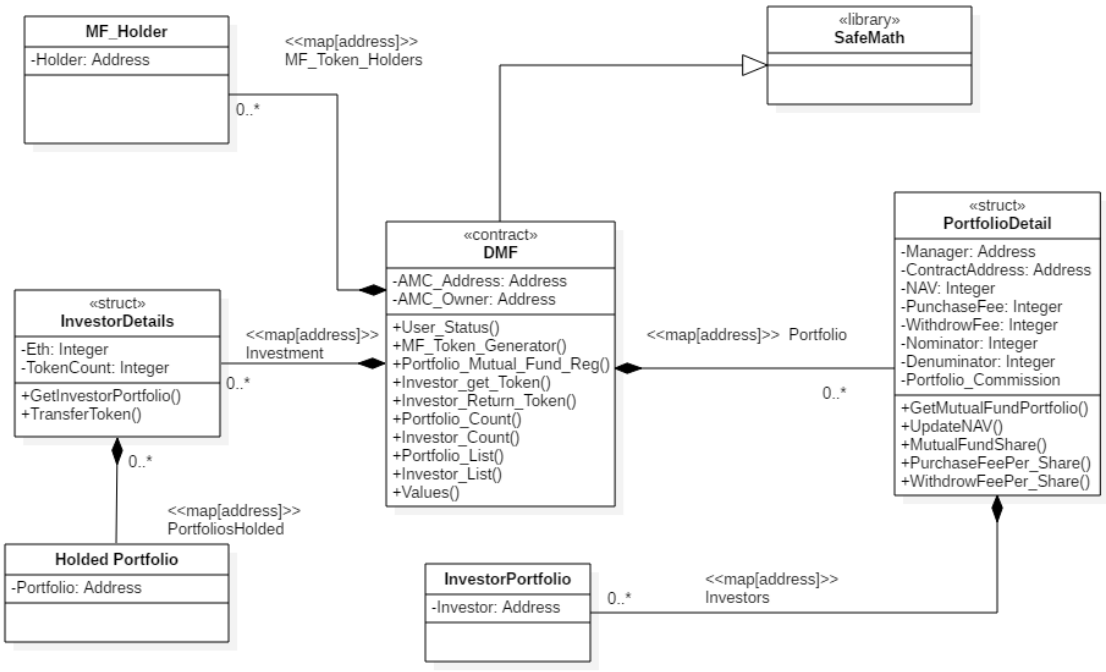

Figure 5.6: Smart Contract Structure Diagram.

5.4 Sequence diagrams

Throughout this section, we provide an overview of the control flow between the different system components and their collaboration in order to achieve a given behavior.

5.4.1 Register Mutual Fund Use case

Next, we detail the scenario of registering Mutual Fund through an object sequence diagram illustrated in Figure 5.7

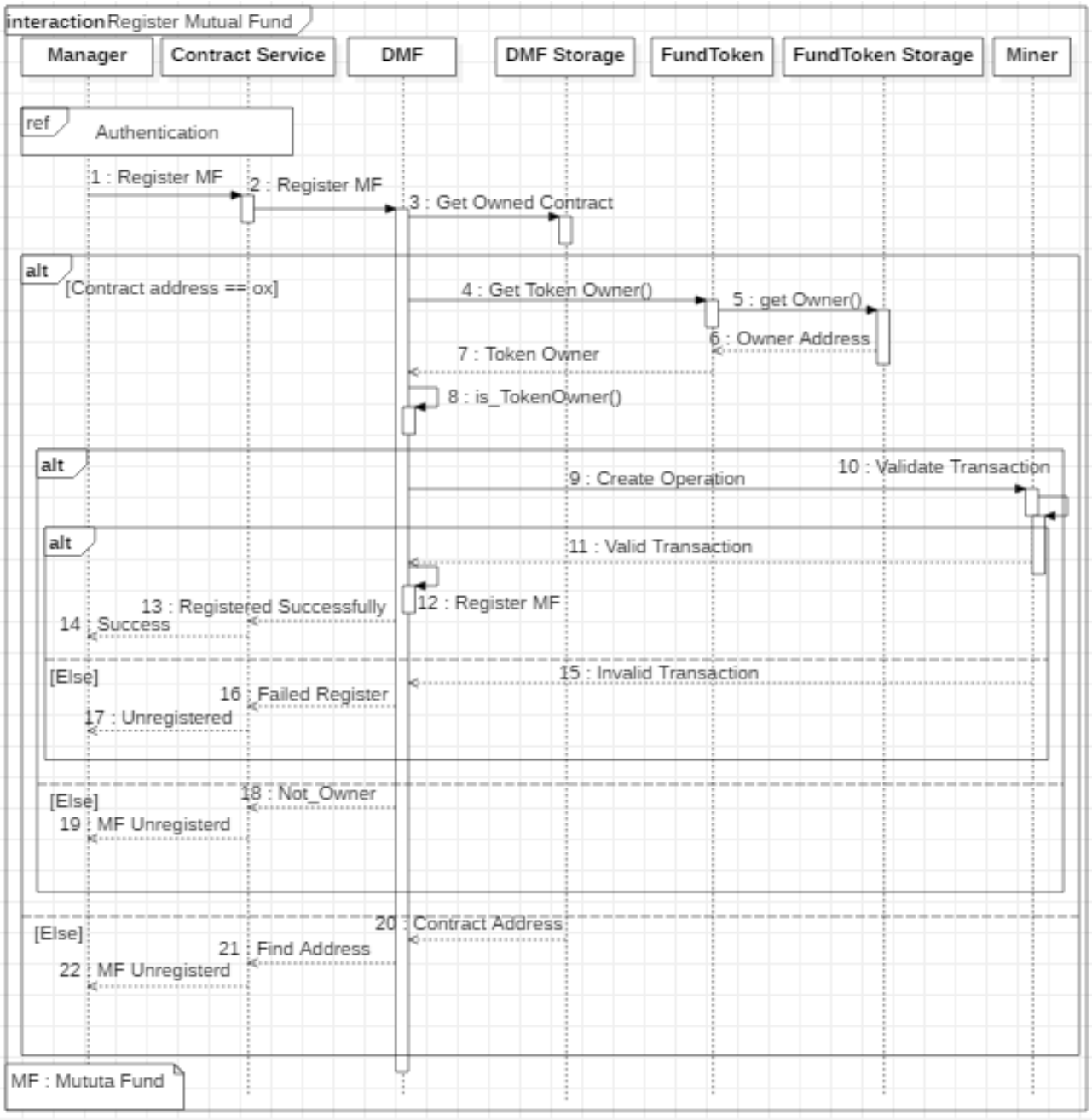

Figure 5.7: Object sequence diagram of Register Mutual Fund Use case.

5.4.2 Purchase Mutual Fund Share Use case

Next, we detail the scenario of purchasing Mutual Fund shares through an object sequence diagram illustrated in Figure 5.8

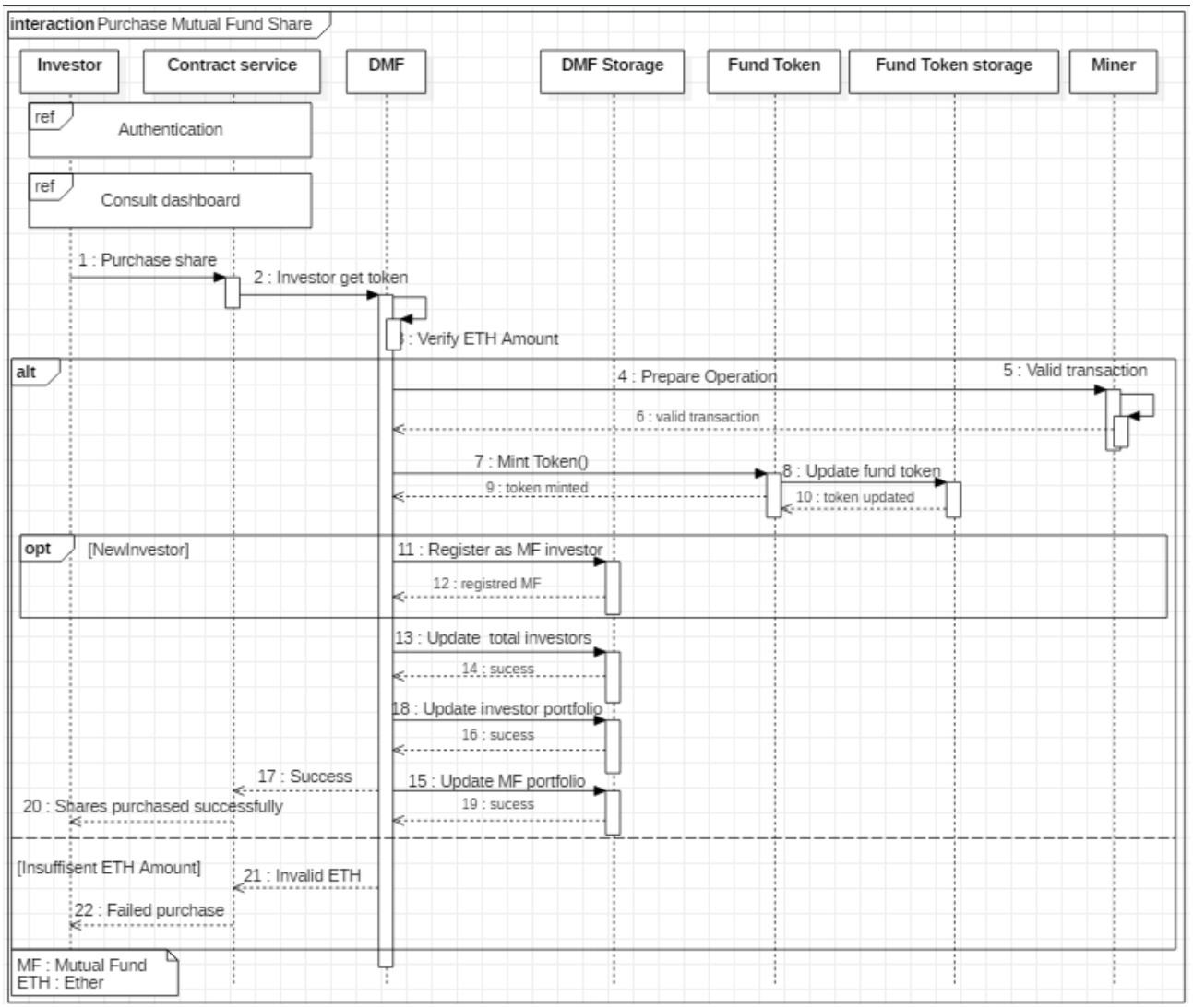

Figure 5.8: Object sequence diagram of purchase Mutual Fund share Use case.

5.5 Conclusion

Throughout this chapter, we come up with with the design study of the system by presenting a detailed architecture and used architectural patterns. We also present useful UML graphics design artefacts. The following chapter presents the implementation as well as obtained results.

Chapter 6

Implementation

6.1 Introduction

The Implementation of our system have to translate the conceptual and design phases. Thus, we present in this chapter the software tools to achieve our project goals. Subsequently, we present an illustrated typical execution scenarios though different interfaces printscreens.

6.2 Work Environment

In this section, we present the working environment part, which allowed us to implement our project.

6.2.1 Hardware environment

The implementation of the project was done on a desktop computer that has the following features:

- Processor Intel(R) Core i7-6500U @2.50GHZ.
- 7 GB of RAM.
- 339GB as hard disk capacity.

6.2.2 Software environment

After clarifying the hardware tools used for development, we present in this section the software tools and technologies used during our implementation phase.

- *Visual Studio Code* : It's an extensible code source editor developed by Microsoft and has a rich ecosystem of extensions for many programming languages and runtimes.
- *StarUML*: It's a sophisticated software modeling tool that supports UML modeling.
- *Remix*: It's a Browser-based IDE and compiler which enables developers to create Ethereum smart contracts with *Solidity* or *Vyper language* as well as debugging transactions.
- *Metamask*: It's a web browser extension used as Ethereum wallet, which allows signing smart contracts and interface with *dApps* without the need of running an Ethereum full node.

- *Ganache-CLI*: It's an Ethereum client behavior simulator used for testing and development. It includes most popular RPC functions and features like events, in order to make developing dapps much safer, faster and easier.
- *Truffle*: It's an Ethereum development environment and testing framework.
- *Mocha*: It's a JavaScript test framework used to test our smart contracts.

6.3 Selected Technologies

In what follows, we introduce the selected technologies used to implement our application.

- *Solidity*: It's an object-oriented programming language for smart contract development.
- *JavaScript*: It's a programming language or scripts used for creating dynamic web pages as well as in servers.
- *TypeScript*: It's designed by Microsoft that includes JavaScript with object oriented features in order to scale the development of application.
- *Angular*: It's designed by Google as Open Source JavaScript framework that uses the *Model View ViewModel* (*abrev.* MVVM) architecture. *Angular* helps in structuring the code and separates the view from the models.
- *Shell*: It's a command line interface for Unix operating systems that provides an access to the internal features of Unix OS.

6.4 Illustrations

In this part, we present the most important interfaces related to typical execution scenarios.

- When launching the application, the user is in front of the Home page presented in the following figure 6.1. Users are automatically authenticated when they unlock their wallets using Metamask.

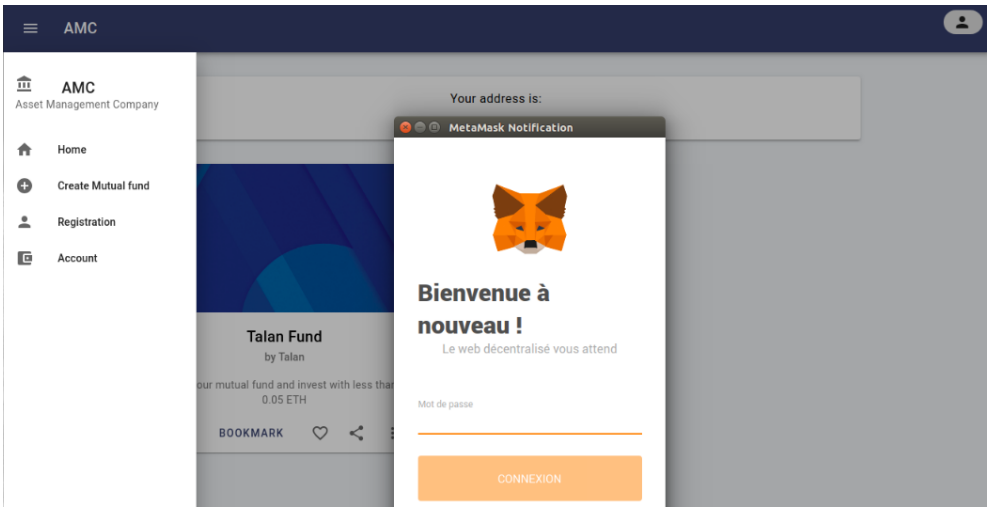

Figure 6.1: Home page

- Once logged into the application, managers can choose to create their own mutual fund that will be accessible for all investors registered in the asset management company *(abrev. AMC)*, the following interfaces, groups all the operations that a manager can do after its authentication in order to establish his mutual fund.
  - The figure 6.2 illustrate the creation process of a mutual fund that will be associated to the manager address.

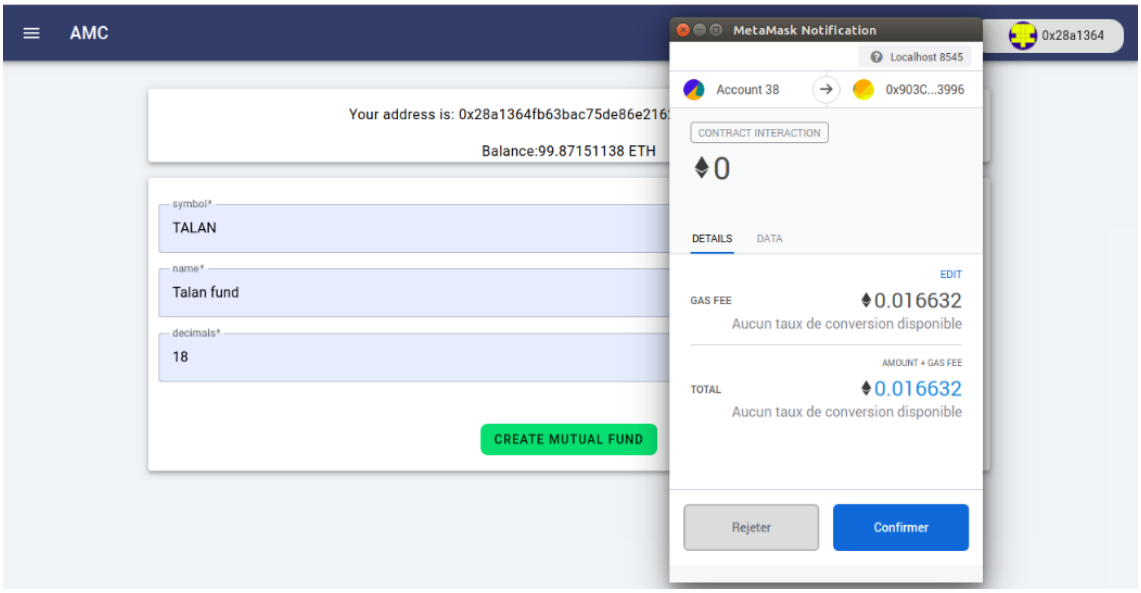

Figure 6.2: Create Mutual Fund

  - Ones the fund is successfully created the manager can register his fund, the figure 6.3 shows the registration process of a customized mutual fund by specifying the net asset value of the fund, management fees and the number of shares created.

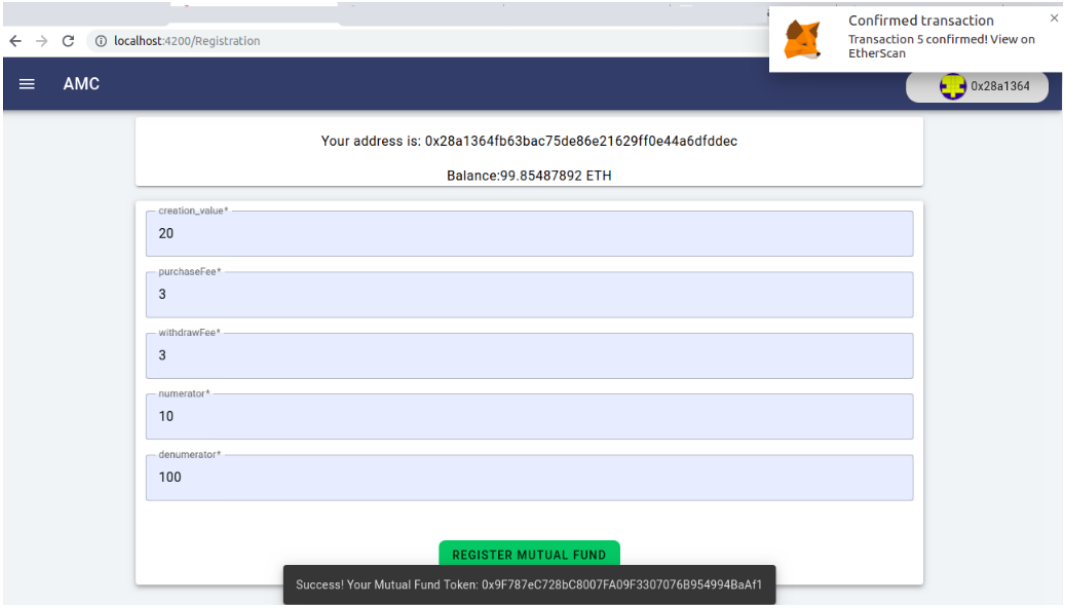

Figure 6.3: Register Mutual Fund

- For an investor, to be able to invest in AMC's mutual funds, he have to log in using his address wallet , consult mutual funds and he can choose to purchase shares from any registered mutual fund. The figure 6.4 present a successful purchase transaction.

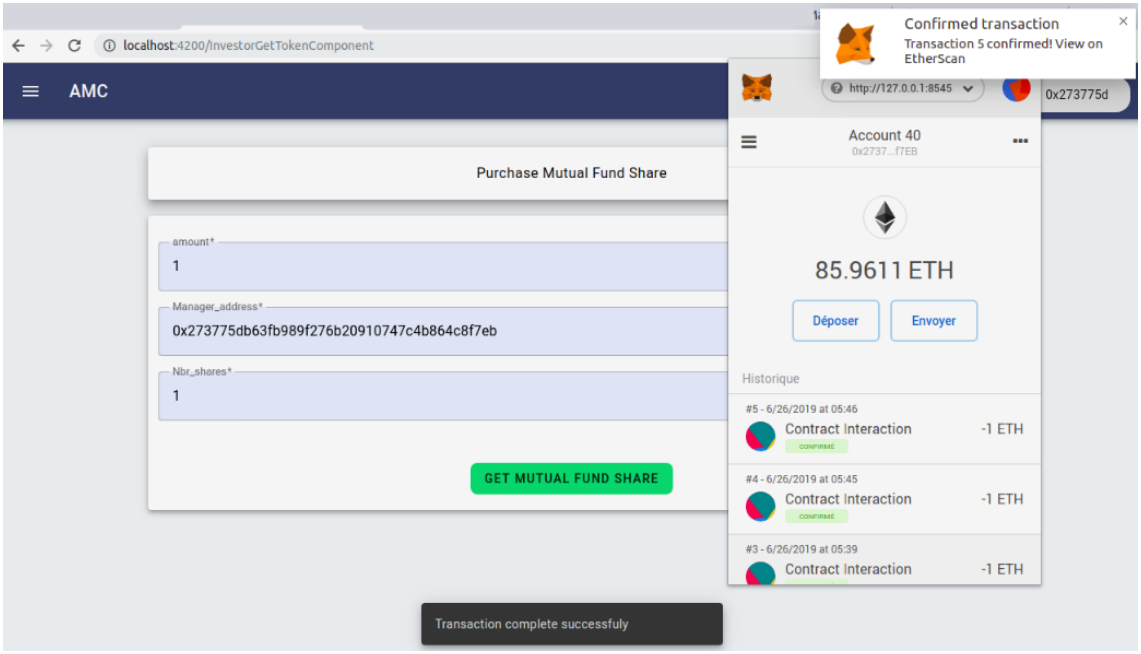

Figure 6.4: Purchase Mutual Fund Shares

- Whenever an investor wants to withdraw mutual fund share, he have to select from his account the mutual fund address and the desired amount of shares to withdraw. The figure 6.5 shows a successful withdraw transaction.

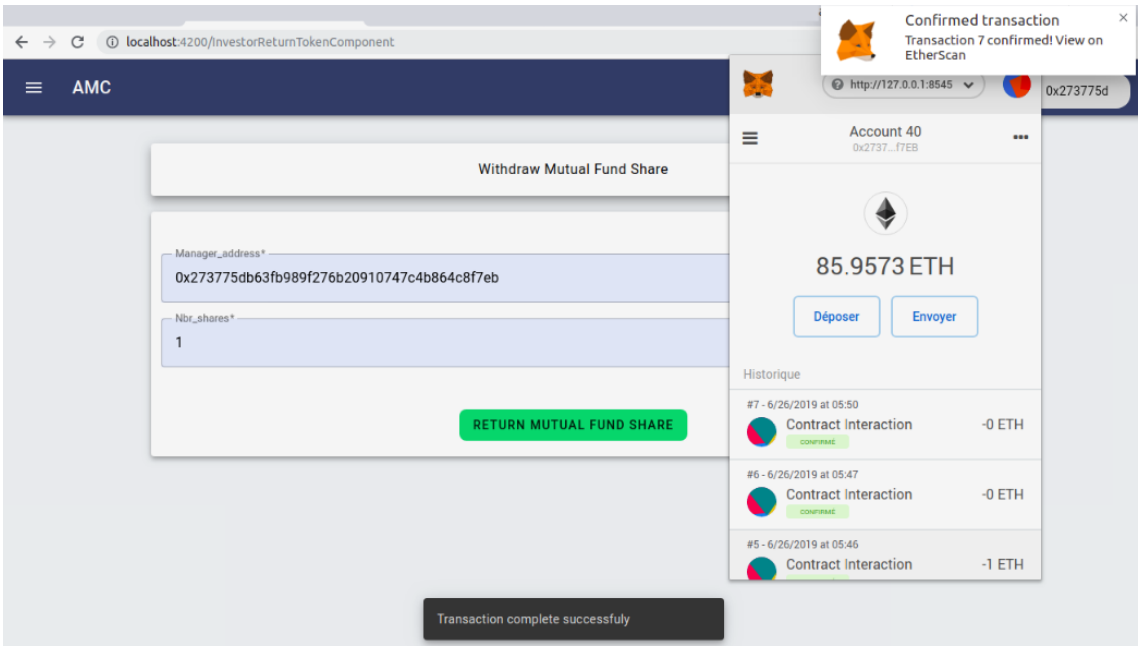

Figure 6.5: Withdraw Mutual Fund Shares

6.5 Conclusion

In this chapter, we present the hardware and software tools as well as the technologies used for the implementation of our application. We subsequently introduced an execution scenario through some interfaces printscreens.

General Conclusion And Perspectives

Internationally, the number of cryptocurrencies used over the internet is highly increasing, As a new cryptocurrency can be created at any time. In this context, we can set facebook cryptocurrency named *Libra* that will be launched in 2020.

Thus, the number of investors, both private and institutional, has risen sharply. As a result, countries, institutions and politicians are planning to regulate cryptocurrencies and to integrate its uses in the actual financial system, since the Blockchain technology behind these cryptocurrencies brings multiple features such as the automation of formalities, increasing transparency and information sharing which ensure the transaction supervision and operations speed.

In this context, our graduation project focuses on the design and implementation of *decentralized application (abrev. Dapp)* for mutual fund portfolio management. which uses the blockchain technology to increase transparency, shares management speed and eliminate any central authority, from mutual fund creation to investors purchase and withdraw of shares based on the update of their Net Asset Value.

In this report, we have detailed the various steps we have taken in order to realize this application, while following Scrum development method.

In conclusion, although the main objectives of the project are achieved, the *Proof of concept (abrev. POC)* developed could be enriched to develop a professional product by including other enhancements and advanced features like adding smart trading component that track market index and update the Mutual Fund NAV automatically instead of being updated manuely by the manager, which can decrease the management fees and and increase the fund income.

Bibliography

[1] *Mutual fund work flow*, *A*vailable at https://www.broto.id/2018/06/tentang-reksadana.html .[Consulted 19-03-2019], [Online].

[2] *Iterative Model*, *A*vailable at https://airbrake.io/blog/sdlc/iterative-model .[Consulted 17-03-2019], [Online].

[3] *Blockchain Workflow*, *A*vailable at http://tiny.cc/drzv8y .[Consulted 10-02-2019], [Online].

[4] *Mastering Blockchain*, *I*mran Bashir,[E-Book], *A*vailable at https://fr.scribd.com/book/382269351/Mastering-Blockchainl .[Consulted 26-02-2019], [Online].

[5] *Transaction*, *A*vailable at http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html .[Consulted 09-02-2019], [Online].

[6] *Block structure*, *A*vailable at https://www.researchgate.net/figure/The-internal-structure-of-block_fig1_325898060 .[Consulted 09-03-2019], [Online].

[7] *Blockchain Transaction Flow*, *A*vailable at https://bull.io/3-blockchain-use-cases/ .[Consulted 26-02-2019], [Online].

[8] *Cosmos*, *A*vailable at https://cosmos.network/intro .[Consulted 03-03-2019], [Online].

[9] *Cosmos Hub Architecture*, *A*vailable at http://tiny.cc/24zv8y .[Consulted 11-03-2019], [Online].

[10] *Mutual Fund Workflow*, *A*vailable at https://www.fincash.com/l/mutual-funds-india .[Consulted 28-03-2019], [Online].

[11] *Mutual Fund Types*, *A*vailable at https://www.jmfinancialmf.com/KC/?SubReportID=ED362A01-64E4-4055-86A5-84E80A1C5A50 .[Consulted 07-04-2019], [Online].

[12] *Mutual Fund structure*, *A*vailable at http://tiny.cc/vvyv8y .[Consulted 03-04-2019], [Online].

[13] *New Mutual Fund Structure*, *A*vailable at https://business.nasdaq.com/marketinsite/2018/MT/How-Blockchain-Will-Revolutionize-Swedens-Mutual-Fund-Industry.html .[Consulted 03-04-2019], [Online].

[14] *Global System Architecture*, *A*vailable at http://tiny.cc/dvzv8y .[Consulted 03-06-2019], [Online].

[15] *MVVM Architecture*, *A*vailable at https://medium.com/clean-code-channel/mvvm-d01c432217af .[Consulted 17-05-2019], [Online].

[16] *Front-End Architecture*, *A*vailable at https://www.edureka.co/blog/angular-tutorial/ .[Consulted 13-05-2019], [Online].

[17] *Talan Tunisia*, *A*vailable at https://tn.talan.com/ .[Consulted 10-02-2019], [Online].

[18] *Bitcoin*, *A*vailable at https://fr.wikipedia.org/wiki/Bitcoin .[Consulted 14-02-2019], [Online].

[19] *Distributed systems*, *A*vailable at https://www.sciencedirect.com/topics/computerscience/distributed-systems .[Consulted 18-02-2019], [Online].

[20] *Smart Contract*, *A*vailable at http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html .[Consulted 20-02-2019], [Online].

[21] *An Agile Software Engineering Method to Design Blockchain Applications*, *A*vailable at https://arxiv.org/ftp/arxiv/papers/1809/1809.09596.pdf .[Consulted 07-03-2019], [Online].

Abstract

This work is part of the End of Study Project realized within Talan Tunisia consulting to obtain the national computer engineering diploma at the National School of Engineers of Carthage. The goal of this project is to create an Ethereum based application to perform Mutual Fund operation by increasing the security and transparency in mutual fund shares management as well as reducing transaction cost and time consuming.
Keywords: Blockchain, Ethereum, Finance, Mutual Fund.

Résumé

Ce travail fait partie du projet de fin d'études réalisé au sein de l'entreprise Talan Tunisie en vue d'otention du diplôme national d'ingénieur en informatique de l'École nationale des ingénieurs de Carthage. L'objectif de ce projet est de créer une application basée sur Ethereum afin d'exécuter des opérations de fonds communs de placement en renforçant la sécurité et la transparence de la gestion des parts de fonds communs de placement, ainsi qu'en réduisant les coûts de transaction et le temps requis.
Mots clés: Blockchain, Ethereum, Finance, fonds communs de placement.

ملخص

يعد هذا العمل جزءًا من مشروع التخرج الذي تم إنجازه في Talan Tunisie من أجل الحصول على شهادة مهندس في الإعلامية من المدرسة الوطنية للمهندسين بقرطاج. الهدف من هذا المشروع هو إنشاء تطبيق يستند إلى Ethereum يهدف إلى زيادة الأمن والشفافية في إدارة أسهم صناديق الاستثمار المشترك اضافة إلى تقليل تكلفة المعاملات و ربح الوقت.