POLITECNICO DI TORINO

Master's Degree Course in Communications and Computer Networks Engineering

Summary

Designing a Blockchain Network of Universities

Supervisors
Prof. Maurizio Morisio
Dr. Luca Ardito

Candidate
Javad Malek Shahkoohi
S224309

Academic year 2018-2019

Summary

The blockchain technology is going to be the next decade technology for most of the business are interested to use and apply it in their daily works and applications. These sorts of use cases are required to customize some basics for themselves such as having a model of data encryption or deciding to use the application as a public or semi-public tool based on their level of transparency.

Since the universities as a place that all the students after their graduation must have a document that proofs they are already graduated and finished their courses, this case study is a good candidate for building a service incorporate with blockchain. This service can be developed like a shared blockchain network between universities (nodes of the network) and it could provide an authorization service for the graduate students to propose their certification id wherever is interested, instead of showing a physical certification.

This thesis tries to build a blockchain network for a bunch of universities and providing a way of communication between them via blockchain technology. The aim is that everybody can refer to the public block explorer and start querying for a student and retrieving the student graduation certification which is secured and trustable. Therefore, to achieve this goal, a semi-public blockchain designed in two main sections. The first one is building a decentralized network for decision making and controlling their network management and the second section is applying the blockchain technology for data management. hence, a set of steps are followed to cover all the demands such as doing global research for proof of concept that it is possible to build a network of universities and etc.

At the end of this step, we found that there should be a decentralized network infrastructure where all the university nodes are exactly the same in all aspects. besides than that there should be a data structure for a chain that provides the customized model of the blockchain. Also, at the second step of this thesis, we investigate the different sorts of development tools and frameworks have already existed for building an application to support the blockchain technology.

At the final implementation, we built two different APIs one is for building a decentralized blockchain network and another one for rest of user interactions are

off the network like sign up/in. With respect to the stack of development in the backend, part NodeJS is used as a programming language and IPFS applied for blockchain file storage and MongoDB is used for off-blockchain data. Besides, the frontend developed mainly by javascript (REACTJS) and HTML.

For software development of authorization of graduate students, this thesis tried to provide minimum requirements like unit tests spacially for decentralized blockchain network API and block explorer endpoints. Also, a group of sequences defined like how to register a mark by a professor, how to mine a block, how to broadcast a mined block through the entire network, how to handle the pending transactions before mining and many others. these sequences are covered by different functionalities of the blockchain.

In the conclusion of this thesis since blockchain is used in different sectors and they allow people to secure their digital relationships more than it was before, data is being disclosed, secured and recorded differently. This concept is changing digital relationships, creating the ability for them to be automated in code via smart contracts and etc.

As a result, **It is possible to authorize the certification of graduated students by blockchain technology.** it means for different universities to share their public data together by creating a decentralized blockchain network of universities and sharing some services and databases together that would be also possible to make them available for the public.

Also, this thesis showed that **It is possible to expand this case study for other sectors.** It which means not only the student certification even though all the other sorts of a document can be stored and verified by using blockchain technology. Some services such as declaration letters or any other documents that somebody needs to verify a piece of information could be a fitted candidate service to publish via a block explorer of a blockchain.

with respect to data storage, **do not store all the data in chain.** it means by storing some data off the chain, mainly low and second-order priority information. it would help to reduce the complexity of the application and it would make the development process to be faster and easier, meanwhile, it would not have a significant effect in security and losing data. it is like a trade-off but a good option for productivity enhancements.

As future tasks, implementing the same functionalities or services in other sections would help people to improve the performance of their workflows in universities or any other organizations. It would right task to work more on adding more functionalities and enhancing the capabilities of developed application mainly in terms of data security and network security. Also, the recalling the network or application is challenging transitions period especially in blockchain because it would produce a huge amount of data and not easy to maintain then working in this area would be a good candidate for future works.

Acknowledgements

Thanks to everybody that who is helped me specially my family, my brother and my girl friend.

Contents

List of Tables

List of Figures

Chapter 1

Chapter One: Introducing the Blockchain Technology

This chapter contains an introduction of blockchain and would review different properties of that as a new growing technology.

The first section is dedicated to fundamental concepts and definitions of the blockchain. The second section would focus on decentralization and how is it used in the blockchain. The third section would talk about a brief introduction of Bitcoin, which is the most famous application of blockchain. At the end of this chapter, a list of potential industries is prepared. Those can use the technology of blockchain to develop their own applications

1.1 What is blockchain and basic concepts

In 2008 an unknown person or group of people known by the pseudonym Satoshi Nakamoto introduced Bitcoin which is a simpler implementation of blockchain technology as a digital currency. Bitcoin became one of the first digital currencies to use peer-to-peer technology to facilitate instant payments. Also, it introduces the term chain of blocks.

Blockchain definitions

There is two famous definition of blockchain in technical and informal ways [1,2].

- **Layman's Definition:**

  The blockchain is an every growing, secure, shared record keeping system in which each user of the data holds a copy of all records. The point is that the system can be only modified if all partners of a transaction agree to update

- **Technical Definition:**

  The blockchain is a peer-to-peer, distributed ledger that is cryptographically-secure, append-only, immutable (extremely hard to change), and updatable only via consensus or agreement among peers.

Blockchain Principles

In this section, we present the main principles behind the blockchain technology as well as cutting edge applications [6, 9].

Peer to peer network: It is a decentralized a network such that all nodes can communicate to each other directly whiteout any third-party. By this feature, it is possible to exchange the transactions through the network whiteout using a bank transaction.

Distributed ledger: The distributed ledger database is spread across several nodes on a peer-to-peer network, where each replicates and saves an identical copy of the ledger and updates itself independently.

cryptographically-secure: the cryptography is applied to make sure that the data of ledger is secure. This also provides data integrity, data authentication, and non-reputation.

Append only: This property means that data can just only be added in sequential order to the chain and would never change. This can be considered as immutable. While in cases the change is acceptable and re-mining should happen and would take a lot of time.

Updatable via consensus: Applying a right and practical consensus algorithm could be a critical bottleneck such that all the security of the ledger must be guaranteed then peer-to-peer network provides the decentralization for an application. As soon as all the nodes of the network confirm the transaction then consensus would be reached between all them then can update their local ledger.

Blockchain Myths

There are a bunch of common blockchain myths make misunderstanding about the advantage and disadvantage of blockchain technology [8].

- **Blockchain is Bitcoin** Bitcoin is just a one cryptocurrency application which made base on blockchain technology. while Blockchain is a general concept such that much different application can use it for their development.

- **Trdeitional databases would be replaced by blockchain because is better** there is a trade-off between using traditional databases or blockchain because traditional databases still perform better. while blockchain is more practical in applications with low-trust environments where there are no intermediary organizations.

- **is blockchain immutable or changing the way of exposing** since the blockchain must be appended only, there is no way to change the stored data. it is possible to be tampered by using a large amount of computing power.

- **blockchain is 100% secure enviroument** By using the immutable data structure to protect data and encrypt all the transactions and blocks, it satisfies the level of safety and security of the system. in general blockchain system, secure depends on the other cooperating applications are interacting together the reason is these partner application can be attacked and hacked.

- **Blockchain is trust machine** Blockchain verifies all the transactions before storing them and data become part of a chain. also, blockchain cannot provide the input assessment and must be done out of blockchain.

How blockchain works

As a brief explanation the way that blockchain works, it can be said that, generally, a node in a network starts a transaction that includes the processes of the creation and encryption of a transaction. Each transaction has a small portion of value that depends on the purpose of blockchain development, for example, in this study the marks of students are carried by transactions. At the next step, the transaction should be broadcasted to all peers of the network by a clear communication protocol between them through a secure network.

At the moment that a peer node receives a new transaction from network, it would start to validate the new transaction. Afterward, this transaction would be added to the block and it would consider as confirmed transaction. the point is still this transaction is not stored because it is in pending state until a node starts the mining process. when the mining is done that data is untouchable and saved for ever. Also, all the recently created blocks become part of the ledger and linked together via a cryptographically hash value. In the end, pending transactions are ready to be mined by a node and stored in the ledger.

Expelling how does blockchain works in simple way 1.1.

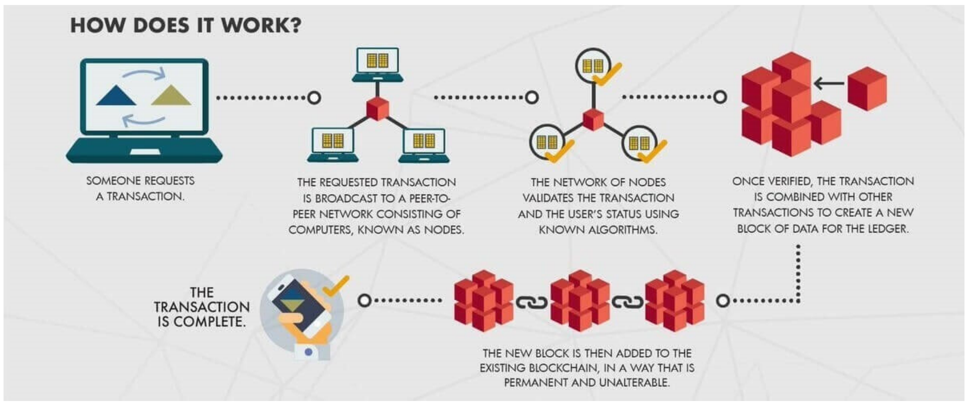
HOW DOES IT WORK?
SOMEONE REQUESTS A TRANSACTION.
THE REQUESTED TRANSACTION IS BROADCAST TO A PEER-TO-PEER NETWORK CONSISTING OF COMPUTERS, KNOWN AS NODES.
THE NETWORK OF NODES VALIDATES THE TRANSACTION AND THE USER'S STATUS USING KNOWN ALGORITHMS.
ONCE VERIFIED, THE TRANSACTION IS COMBINED WITH OTHER TRANSACTIONS TO CREATE A NEW BLOCK OF DATA FOR THE LEDGER.
THE NEW BLOCK IS THEN ADDED TO THE EXISTING BLOCKCHAIN, IN A WAY THAT IS PERMANENT AND UNALTERABLE.
THE TRANSACTION IS COMPLETE.

Figure 1.1: Bockchain works in simple word.

1.2 Blockchain data structure

The blockchain data structure is a one-way link list as a back-linked record of blocks of transactions, where all the blocks are recorded in an ordered based on their creation index number. The ledger can be saved in a file or in a plain database which depends on the application case study. Also, the structure of blockchain depends on a case study that aimed to be applied. In general, a couple of attributes must be always there as essential attributes like *address, timestamp, genesis block, nonce, Markle root and block body* that contains the transactions. Let's first look at each term more closely [1,6].

- A **Block** is a bundle of transactions which are logically organized and connected to previous block's hash pointer.
- A **Transaction** is a record of an event like the event of transferring cash (or assigning a mark to a student like our use case) from a sender to a recipient. Then transaction the smallest unit of blockchain and every block made up of a set of transactions. the number of transactions can be different based on the type and design of blockchain.
- A **Genesis Block**, every block has a reference to the previous block except the Genesis Block which is the first block of chain and mostly has hardcoded values at the time that chain is created.
- An **Address** is a unique identifier which is used in a transaction to denote its sender and recipient. Also, can be generated by a public key.
- A **Nonce** is a generated number which used only once and popular in all the cryptography methods. blockchain uses that for proof of work of consensus algorithm.

- The **Markle Root** is the hash of all nodes of Markle tree that used to validate the large-scale data structure.

The data structure of blockchain discussed more in details with sample data of implementation in chapter three.

1.3 Types of Blockchain

There are different types of blockchain in the market from technical and business point of views such that they use various sort of architectures and data management techniques. At the following you would see a general introduction of a bunch of them [2,3]:

Distributed ledgers and distributed ledger technology (DLT)

first of all, the distributed ledger is a wide term and covers all sort of shared databases therefore all the blockchains are distributed ledger but not all the distributed ledgers are blockchain. The main difference is that distributed ledger does not necessarily consist of blocks of transactions to grow the ledger. While blockchain is a sort of shared database which is built by blocks of transactions. The DLT is a model of the distributed ledger that mainly used in the financial sector. the people in this sector believed that the DLTs are permissioned blockchains which are shared between different nodes of a network. besides, the point is, DLT is a shared database between all participants ,hence, they knew and verified.

Public and private blockchains (and semi-private) blockchains

The distinction between public and private blockchain is related to who is allowed to participate in the network, execute the consensus protocol and maintain the shared ledger. A public blockchain network is completely open and anyone can join and participate in the network. While the private blockchain network is open only for the authorized group to participate there [10]. Besides, there is another version which is a hybrid model of both public and private chains such that some parts are public and anybody can cooperate there but some other parts are private and only special group access there.

Permissioned ledger

Permissioned ledger is a blockchain that participants are known and it does not need to use the consensus mechanism just an agreement protocol is enough for that. While participants need special permissions to read, access, and write information on chain. The intrinsic configuration of such blockchains controls the participants' transactions and defines their roles in which each participant can access and contribute to the blockchain. It may also include maintaining the identity of each blockchain participant on the network. Such blockchains are called permissioned blockchains.

Tokenized and tokenless blockchains

Tokenized blockchains are a standard version of blockchain that create a cryptocurrency as a result of a consensuses process via mining or initial distribution. with respect to token, fewer blockchains are used when there is nothing to be transferred. There also other sorts of blockchains are discussed [2, 3].

In this thesis, we are going to use the semi-private version of blockchain for final implementation. It is a hybrid version of public and private blockchains such that the accessibility of participants to some parts like registration of students and registration of exam's marks is closed only for a private group of users have access to them. On the other side, the data retrieval is free and available for everybody that who wants to read data from blockchain. You can read more in chapter three, where the different aspects of desire model are explained.

1.4 Decentralizations and blockchain

Decentralization is the core benefit and main service provided by blockchain technology. It distributes the decision making through the network and all the users would confirm everything [2–4]. A decentralized system is a type of network where nodes are not independent on a single master node, instead of that control is distributed among many nodes of the network. This is a reason that every department is in charge of its own database and has independent power from a central server. In between a significant innovation in the decentralized system is the decentralized consensus. This mechanism is introduced by Bitcoin.

Methods of decentralization

Two main methods can be applied to reach a decentralized system, which is disintermediation and competition. When you need a concept at the middle of your transaction to confirm a transaction, it would be great to get ride this intermediate entity and transfer to the destination directly. This method called disintermediation and can be used mainly in the financial sector. While in competition method you can remove this intermediary concept by using blockchain. Like participating different service providers for the same service. In blockchain technology, a system can use smart contracts or can choose an external data provider from a large number of providers based on their criteria score, reputation, and other params.

Blockchain decentralization eco-system

Blockchain decentralization includes contains many items are cooperating together to serve different aspects of the application such as storage, network communications, and others. these items are discussed below:

Data Storage

Data can be stored in blockchain and achieve the decentralized architecture but the point is that it is not possible to store a large amount of data in chain because of overhead cost and network data balance. There are a couple of alternative ways to store data of blockchain like,

- *Distributed Hash Table (DHT)*, which is popular for peer-to-peer file sharing systems.
- *Interplanetary File System (IPFS)*, which would be used when links are stable it means data should be ready when required and the link must be available.

There are other alternative ways like *Filecoin, Bitwasp, Ethereum Swarm, Storj and MaidSafe*. Based on the use-case and application policies, selecting a method for data storage can different, for example, in this thesis Distributed Hash Table is used and each node has a copy of all data of mark transactions, while the authentication data are stored in off-chain..

Communications

The communications of nodes can be provided by different network protocols. For this thesis, the HTTP protocol is used for data transmission between nodes and API layers. The network is built virtually by simulating the different nodes to be exactly the same as each other and communicating with each other through different ports number. Also, it is possible to add a new node to a network that explained in the next chapters

Smart contracts

Smart contracts are a decentralized program that can be executed even without a blockchain. It usually contains some business logic and a limited amount of data [1]. The point is the business logic would be executed if the specific criteria are specified in the smart contract are met. In this thesis, the smart contract concept would be implemented inside the blockchain structure not separately.

Computing power

Decentralized computation and processing power consummation are the main issues that are required for executing smart contracts with a specific logic. In this thesis, it is not a big issue and ignored.

Decentralized applications

Decentralized applications (dApps) are applications that run on a P2P network of computers rather than a single computer. Decentralized applications don't necessarily need to run on top of a blockchain network. The applications like *BitTorrent, Popcorn Time, BitMessage, Tor*, are traditional dApps that run on a P2P network [5].

Regarding the blockchain dApps, for an application to be considered a dApp in the context of Blockchain, it must meet the following criteria:

- The application must *be completely open-source.* It must operate autonomously and with no entity controlling of the majority of its tokens. The application may adapt its protocol in response to proposed improvements and market feedback, but the consensus of its users must decide all changes.
- Application's data and records of operation must *be cryptographically stored* in a public, decentralized blockchain in order to avoid any central points of failure.
- The application must *use a cryptographic token*, (Bitcoin or a token native to its system) which is necessary for access to the application and any contribution of value from miners. It should be rewarded with the application's tokens.
- The application must *generate tokens.* according to a standard cryptographic algorithm acting as a proof of the value, nodes are contributing to the application (Bitcoin uses the Proof of Work Algorithm).

1.5 Introducing Bitcoin

Bitcoin is the first and popular application of blockchain technology that everybody at least must read and know about it. Here is an introduction to it that most people believe that everybody in this field has to know about Bitcoin.

Bitcoin is a consensus network that enables a new payment system and completely digital money. It is the first decentralized peer-to-peer payment network that is powered by its users with no central authority or middlemen. From a user perspective, Bitcoin is pretty much like cash for the Internet that can be used in different applications and websites for their payments and financial transactions [6]. This section tried to talk about a bunch of most popular questions nearby Bitcoin and its technology.

Who created Bitcoin?

Bitcoin is the first implementation of a concept called *"crypto-currency"*, which was first described in 1998 by Wei Dai on the cypherpunks mailing list, suggesting the idea of a new form of money that uses cryptography to control its creation and transactions, rather than a central authority. The first Bitcoin specification and proof of concept were published in 2009 in a cryptography mailing list by Satoshi Nakamoto.

Who controls the Bitcoin network?

Nobody owns the Bitcoin network much like no one owns the technology behind email. Bitcoin is controlled by all Bitcoin users around the world. While developers are improving the software, they can't force a change in the Bitcoin protocol because all users are free to choose what software and version they use. In order to stay compatible with each other, all users need to use software complying with the same rules. Bitcoin can only work correctly with a complete consensus among all users. Therefore, all users and developers have a strong incentive to protect this consensus.

What are the main components of Bitcoin?

Based on the white paper of Bitcoins its key elements are:

- **Private Digital keys** are given to every cryptocurrency holder to allow them to be able to authenticate their identity and this is what allows them to exchange units.
- **Addresses**
- **Transactions**
- **Blockchain** of a cryptocurrency is essentially a method of recording data and acts as a digital leger that keeps track of all contracts, agreements and transactions that take place. As the block chain keeps track of the entire history of a cryptocurrency's transactions, it increases in length over time and is finite.
- **Miners** are important to the life cycle of the Bitcoin as they act as the indirect influencers on the cryptocurrencies' value and work as record keepers for cryptocurrency communities. Miners use huge amounts of computer power and

technical skill to check the security, accuracy and completeness of currencies like the Bitcoin's block chains

- **The Bitcoin networks**
- **Wallets** which is another component of cryptocurrencies is that users are required to have their own wallets that include unique and highly user specific information that confirms their ownership of the wallet and their cryptocurrency unit ownership.

More information is provided in Bitcoin.com as a main reference [6].

1.6 Transactions and mining

The transaction is the smallest unit in blockchain and it should be customized based on the target application. It defines a life cycle for blockchain to make clear what should be transferred between nodes and how must be done.

A possible life cycle of the transaction can be as following [1]:

- A user as a sender of a transaction sends a request to a node.
- Node signs and confirms the transaction then it would add the new transaction into it is local pending transactions list.
- The transaction should be broadcasted through the network by a flooding algorithm.
- Each receiver node would confirm and update its own pending transactions list and reply Ack to the source of the request.
- Each node or a group of nodes can do mining. During the mining process, all pending transactions are removed and encrypted to be a new block. At this point, the miner would flash the pending list to be ready for next block.
- After mining the all the nodes have a new mined block and abort if they were doing mining at the simultaneously.

Besides, in some applications, there is a fee for each transaction and mining reward for miners. In this thesis use case, there is no reward and fee for transactions. While as further feature, it is suggested to have a transaction fee to make a prime account for universities. But it is skipped for this thesis application.

1.7 Smart contract

Definition of the smart contract says that [2,5], a smart contract is an unstoppable and secure programming code is running on top of a blockchain containing. It carries a set of interacting rules that all the participants of the network are agreed on them. Therefore, when all or part of the rules is satisfied then there would be an automatic agreement between them.

There are a couple of features that the code of smart contract code should provide for transactions of blockchain such as it must be automatically executable without and manual interactions, it must be enforceable, and it must be secure and unstoppable. The first two features are minimum requirements that every smart contract should provide and the third one is an optional feature that mostly depends on an application case study.

1.8 Use cases of blockchain

A look at some real-world uses of blockchain in the enterprise and the various applications of blockchain that can help revolutionize industries. Here are some of the most compelling blockchain use cases from different industries that they used blockchain technology in their daily routine.

Supply chain management: There is a lack of transparency between suppliers. Hence, increasing accountability for all the middlemen is a priority. With blockchain's distributed ledgers, this issue can be resolved. Blockchain allows multiple people to access the same database which will increase transparency in the supply chains. Blockchain ledgers are immutable and all the valid transactions are appended with a timestamp which can help in auditing and stop theft and counterfeits. This use case can improve regulatory compliance, reduce paperwork and help cut cost significantly. Examples of projects working on this include *Vechain* and *Origin Trail.*

Protecting digital identity: Identity theft has become very common today and improvements in digital security haven't been able to provide a definitive solution to the ever-growing need for Internet security. With blockchain's immutable ledgers, users can store their personal data securely. Due to its decentralized network, data on blockchain is not vulnerable to hacking. A sovereign user ID can help users access their data without any hassle. Examples of projects include *Civic* and *Uport.*

Smarter predictive analysis: With all the transaction data stored on the blockchain ledgers, these ledgers can help excavate huge amounts of data and insights. Blockchain-based artificial intelligence can help make accurate forecasts.

Endor, a blockchain-based AI, uses natural language processing to answer questions in real time. With its help, businesses can make use of blockchain data in their projects, privately.

Health care: Currently, people's medical history and records are not properly documented. The data sits on legacy silos, which leads to restrictions in sharing of data instantly. With the help of blockchain, the medical history of an individual can be stored securely on the distributed ledgers. This data will be easy to access when the need arises. Different stakeholders can have conditional access to the distributed ledger. Blockchain will help keep an audit trail on individual's medical history through time stamp. Users will also be able to monetize on their medical data for research purposes. Examples of healthcare data exchange platforms include *Medicalchain.*

Energy: Blockchain provides a compelling solution for the energy market. By recording people's utility in a ledger, the data can help people deal with energy like any other commodity. Currently, large corporations sell energy at fixed prices, but with the implementation of this use case, the energy market will be forced to follow supply and demand in a particular region. Examples of peer to peer energy trading include *Power Ledger* and *Grid+*.

Land title registration: Frauds are common in property registration. However, using blockchain ledgers, the record keeping will be efficient and the chances of fraud will be reduced significantly. Blockchain's solution will also help reduce labor and lessen paperwork. This use case has seen some serious traction in recent years where countries after countries are joining in and tying up with blockchain based startups to implement it.

1.9 Summary

The blockchain is a distributed database system that means, instead of storing files on a single node, information is stored across many nodes through a network. In the other hand, as the name of Blockchain says it is a chain of different blocks that are linked together one after the other. All nodes on the network have the same and full replication of all transactions that have taken place on the blockchain ever since the genesis block is mined. Based on the type of blockchain the distributed ledger can be open and the transaction between users or members will be displayed on the ledger for the whole the users might have access to see them. The transactions are encrypted by using a cryptographically encryption method and the same would be done to encrypt the next block.

The blockchain is used in different sectors and mainly in the finance market and all of the applications are developed by blockchain are allowing people to secure their digital relationships more than it was before. Data is being disclosed differently, secured differently and recorded differently. This is changing digital relationships, creating the ability for them to be automated in code via 'smart contracts'.

Chapter 2

Chapter Two: Blockchain Development Frameworks and Tools

This chapter talks about popular frameworks and techniques for developing decentralized applications and blockchain. It would also try to introduce a bunch of practical tools can be used for application development. in the last part, the similar works are mentioned and would talk about the similarities and differences in the range of various aspects.

2.1 Hyperledger

Hyperledger is a consortium of communities which works on two different levels [7]. These levels are related projects and working groups. Projects focus on implementations of blockchain frameworks, tools, and modules. While working groups focus on different aspects, typically independent of any specific implementation.

Hyperledger projects

- **BURROW** Permissioned Ethereum smart-contract blockchain and its state currently is in Incubation state.
- **FABRIC** Distributed ledger in Golang and its state currently is in Active
- **INDY** Distributed ledger purpose-built for decentralized identity and its state currently is in Incubation state.
- **IROHA** Distributed ledger in C++ and its state currently is in Active state.

- **SAWTOOTH** Distributed ledger with Multi-Language Support and its state currently is in Active state.

Hyperledger Tools

Hyperledger has a set of tools to cooperate with projects and already existed blockchains:

- **CALIPER** Blockchain benchmark framework which allows users to measure the performance of a specific blockchain implementation with a set of predefined use cases. It supports Fabric, Sawtooth, Iroha.
- **CELLO** Blockchain management/operation. It supports Fabric,Explorer (Sawtooth, Iroha, Composer)
- **COMNPOSER** Development framework/tools for building Blockchain business networks. It supports Fabric, (Sawtooth, Iroha) Blockchain Web UI. It supports Fabric, (Sawtooth, Iroha)
- **EXPLORER** An interoperability solution for blockchains, DLTs and other types of ledgers
- **QUILET** A shared cryptographic library that would enable people (and projects) to avoid duplicating other cryptographic work and hopefully increase security in the process.

in this thesis no one of them used and its mentioned here as an information.

2.2 Ethereum Platform

Ethereum is a decentralized platform that runs smart contracts exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference [11]. These apps run on a custom built blockchain, a powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middleman or counterparty risk.

Key Features of Ethereum Platform:

- Smart money, smart wallet: The Ethereum Wallet is a gateway to decentralized applications on the Ethereum blockchain. It allows you to hold and secure ether and other crypto-assets built on Ethereum, as well as write, deploy and use smart contracts.

- Developing, Design and issue a cryptocurrency
- Creation of democratic autonomous organizations (DAOs)
- Building the decentralized application
- Command line tools built in Go, C++, Python, Java etc.

2.3 BigchainDB Platform

BigchainDB is a scalable blockchain database. That is, it's a "big data" database with some blockchain characteristics added, including decentralization, immutability and native support for assets. At a high level, one can communicate with a BigchainDB cluster (set of nodes) using the BigchainDB Client-Server HTTP API, or a wrapper for that API, such as the BigchainDB Python Driver. Each BigchainDB node runs BigchainDB Server and various other software [12].

Main Features of BigchainDB:

- **Native Support of Multiassets:** With no native currency on BigchainDB, any asset, token or currency can be issued.
- **Customizable:** Design your own private network with custom assets, transactions, permissions and transparency.
- **Rich Permissioning:** Set permissions at transaction level to ensure a clear separation of duties and enforce selective access.
- **Federation Consensus Model:** Decentralized control via a federation of voting nodes makes for a super-peer P2P network.
- **Open Source:** Open sourced to the community so that everyone can use it and build their own applications on top of it.
- **Public or Private:** Roll out your own public or private networks for specific industry use cases.
- **Query:** Leverage efficient big data query capabilities out of the box.
- **Petabytes Capacity Coming soon:** Store arbitrary asset and transaction metadata right in BigchainDB, not elsewhere.
- **Linear Scaling Coming soon:** The more nodes added, the higher the throughput and the higher the storage capacity.

2.4 Chain Core Platform

Chain Core is enterprise-grade blockchain infrastructure that enables organizations to build better financial services. Chain proposed an open-source protocol, The Chain Protocol defines how assets are issued, transferred, and controlled on a blockchain network. It allows a single entity or a group of organizations to operate a network, supports the coexistence of multiple types of assets, and is inter-operable with other independent networks [13].

Chain Core Features for Financial activities:

- **Native digital assets:** A new medium for currencies, securities, and other issued financial instruments.
- **Permissioned network access:** Role-based permissions for operating, accessing, and participating in a network.
- **An immutable ledger:** A perfectly auditable record of transaction activity that cannot be forged or altered.
- **Multi-signature accounts:** Secure, multi-asset account management for individuals, businesses, and institutions.
- **Full-stack security:** HSM integration, stable cryptographic primitives, and an auditable, open source stack.
- **Instant settlement:** Federated consensus that allows for immediate transaction confirmation with absolute finality.
- **Smart contracts:** Programmatic transactions, facilitating complex agreements with automatic enforcement and no counterparty risk.
- **Transaction privacy:** Only the parties involved in a transaction (as well as those they authorize) can view transaction details.
- **Reference data:** Assets definitions, compliance data, and arbitrary annotations are included directly in the transaction structure

Features for developers:

- **Developer dashboard:** The Chain Core dashboard provides a powerful interface for building applications and exploring blockchain data.
- **Intelligent queries:** The Chain Core APIs enable intelligent queries on any part of a transaction, from account aliases, to asset IDs, to arbitrary reference data.

- **Multi-language support:** SDKs are available in several languages, including Java, Ruby, and Node.JS.

Chain Core Developer Edition is free while the Chain Core Enterprise Edition is a commercial product. Chain Core proposed Sequence which is a secure, cloud-based ledger service for managing balances. Sequence lets software teams focus on shipping and scaling their product instead of building and maintaining ledger infrastructure. It does this by combining the convenience of SaaS with the security of cryptographic transaction signing. Also, Sequence can serve as a system of record for applications like mobile wallets, lending platforms, sharing economy apps, exchanges, payment services, funding sites, asset managers, and more. Any application that has a balance is well suited.

2.5 Corda Platform

Corda is an open source blockchain project, designed for business from the start. Only Corda allows you to build interoperable blockchain networks that transact in strict privacy. Corda's smart contract technology allows businesses to transact directly, with value [14]. In another word, Corda is a decentralised database system in which nodes trust each other as little as possible.

Codra's main Features:

- Smart contracts that can be written in Java and other JVM languages
- Flow framework to manage communication and negotiation between participants
- Peer-to-peer network of nodes
- "Notary" infrastructure to validate uniqueness and sequencing of transactions without global broadcast. A notary is a network service that provides uniqueness consensus by attesting that, for a given transaction, it has not already signed other transactions that consumes any of the proposed transaction's input states.
- Enables the development and deployment of distributed apps called CorDapps
- Written in Kotlin, targeting the JVM

Also a set of tools are provided by Corda team for developers are going to build applications based on Corda Platform such as Network Simulator, Demo Bench, Node Explorer, Building a Corda Network on Azure Marketplace, Load testing, and Next/Previous Block.

2.6 Domus-Tower Platform

Domus Tower Blockchain is an interesting solution that has been designed for regulated environments such as securities trading where participants know each other and can independently decide whom to trust [15].

Domus Tower Blockchain's key features include:

- Creation of linked blockchains where the assets of an account on one blockchain must match the liabilities on the account of another blockchain.
- Capability of recording a high rate of transactions in a scalable manner.
- Recording of double-entry balance sheet that tracks credits and debits.
- Consensus mechanism: Any agent that has write access to a blockchain has 100

2.7 Elements Blockchain Platform

As open source, protocol-level technology, developers can use Elements to extend the functionality of Bitcoin and explore new applications of the Blockchain [16].

Features of the Elements blockchain platform:

- **Asset Issuance:** Create new assets on a sidechain, with optional confidentiality.
- **Confidential Transactions:** Preserve security while simultaneously obscuring transaction values.
- **Segregated Witness:** Reduce the required space for transactions in a block by a factor of 4.
- **Relative Lock Time:** Allows a transaction to be time-locked, preventing its use in a new transaction until a relative time change is achieved.
- **Schnorr Signature Validation:** A new way of constructing signatures for transactions, both improving efficiency of validating a transaction and offering new modes of multi-signature.
- **New Opcodes:** Introduces DETERMINISTICRANDOM and CHECKSIGFROMSTACK, in addition to re-enabling several scripts previously enabled in Bitcoin.

- **Signature Covers Value:** This allows the signature on a transaction to be invalidated if the inputs have been spent, making it faster and easier to validate a transaction, simply by checking its signature.

- **Deterministic Pegs:** Deterministic Pegs allow cross-chain transactions to be constructed in a decentralized fashion. Tokens can be moved from one blockchain to another.

- **Signed Blocks:** Blocks can be cryptographically signed, allowing the creator of the block to verify their identity in the future.

- **Bitmask Sighash Modes:** Allow arbitrary, miner-rewritable bitmasks of transaction inputs and outputs.

Compared to Bitcoin itself, it adds a couple of new features such as Confidential Assets, Confidential Transactions, Additional opcodes, Deterministic Peg, and Signed Blocks. It is possible for a developer to develop a sidechain and attach to the platform. Sidechains are extensions to existing blockchains, enhancing their privacy and functionality by adding features like smart contracts and confidential transactions. Sidechains can be used to test new features for Bitcoin, change network and security properties, and even stand alone as production networks.

2.8 Monax Platform (Erisdb)

The Monax platform is an open platform for developers and DevOps to build, ship, and run blockchain-based applications for business ecosystems [17]. The Monax platform is for building, testing, maintaining, and operating ecosystem applications with a blockchain backend. Wrangling the dragons of smart contract blockchains has never been easier.

Available paid SDKs and Add-On Modules:

- **Base SDK:** The Base SDK contains all the smart contracts that help to set up permissions, authorizations, organization structure and manage the lifecycle of any ecosystem application developed on the Monax platform.

- **Finance SDK:** contains all the smart contracts included in the Base SDK as well as financial account management and simple asset lifecycle contracts which help with clearing, netting, and other necessities.

- **Insurance SDK:** contains all the smart contracts included in the Base SDK as well as insurance account management and simple risk instrument lifecycle contracts which help with sufficiently understanding risk profiles.

- **Logistics SDK:** contains all the smart contracts included in the Base SDK as well as logistics account and vendor management along with simple asset tracking contracts.

Burrow

burrow is Monax's blockchain node. It is a controllable (permissionable), smart contract-enabled, proof-of-stake based blockchain design. burrow can be configured to work with a wide variety of individual blockchain networks. Its Key Features are:

- **Consensus:** burrow uses the Tendermint consensus engine, a deposit based proof of stake protocol which is much more environmentally friendly, decentralisable, speedy, and final than proof of work.

- **Interface:** burrow comes with a range of interfaces from CLI tooling to RPCs.

- **Virtual Machine:** burrow comes with a built-to-specification Ethereum Virtual Machine. It runs any contract which has been compiled with the compilers or any of Ethereum's compilers.

- **Permission Layer:** Permissioning your blockchains is more than simply running them behind a gated VPN if one is running burrow.

2.9 Openchain Platform

Openchain is an open source distributed ledger technology. It is suited for organizations wishing to issue and manage digital assets in a robust, secure and scalable way. The consensus mechanism used by Openchain differs from other Bitcoin-based systems, it uses Partionned Consensus [18].

The key features of Openchain Platform are:

- The platform allows anyone to spin up a new Openchain instance within seconds.

- End-users can exchange value on the ledger as per those rules.

- The administrator of an Openchain instance defines the rules of the ledger.

- Every single transaction on the ledger is digitally

2.10 Similar works

All the explained frameworks or tools can be the similar to in terms of creating a decentralized network or they may use another partner program to provide all the development requirements. while in this thesis objective is to have a two layers Blockchain to make it semi-public and be private for universities.

on the other hand, the building an application for authorizing the graduated students form different universities might be similar to other thesis that [21]. while there a big differences between them as follows:

- The first difference is that in this thesis instead of using an already existed framework for Blockchain, whole the blockchain architecture implemented which contains all main features of Blockchain and defining its own data structure.
- The second difference is that a dedicated decentralized network is built by flexible node network instead of using a network simulator such as MetaMask.
- Architecture design As it is explained in chapter three and four, two different REST-API for two different purposes are developed one for blockchain networking another one is for website management and intercommunications. While Yokubov used a framework for everything. for optimization of application MongoDB is used for blockchain off data to avoid storing a lot of unwanted transactions in a chain. as software development method, in this thesis Test Driven Development is used and providing the automation test for BE part.
- multi-tenant management. We both have three levels of users are Administration, Professors, and students. While their capabilities are different.
- Block explorer The block explorer is available for everybody to open and search for whatever is looking for and there is no limitation for extracting data

2.11 Summary

Blockchain is a new concept for modern and secure application development and it is still not well matured. This fact is true even for its development tools whereas a lot of different companies and groups already started or built some tools and frameworks for that. but all are in competition with together. Among all of them, Ethereum is the most popular and trending framework for blockchain application development nowadays.

The point is these tools are mainly needed to be compatible to cooperate with other sorts of aside tools that all are built in different languages that makes a bit hard to prepare a fully reliable environment. In this thesis, an overview of a bunch of them provided and tried to build its own blockchain and not using them while applying their strong points. after then that an application of universities developed based on that.

The table 2.1 shows a comparison between different frameworks for blockchain development.

Framework or Tools	**Application**	**Language**	**Data Model**	**Consensus Algorithm**
Hyperledger	Smart contracts	Java, GoLang	Account-basedn	PBFT
Ethereum	Crypto curence and smart contracts	Solidity, LLL	Account-basedn	PoW
Corda	Smart contracts	Kotlin, Java	Account-basedn	Raft
Ripple	Crypto curence and smart contracts	-	UTXO	PoS
Eris-DB	Smart contracts and operating ecosystem	Solidity	Account-basedn	BFT
ChainCore	Crypto curence	Java, NodeJS	UTXO	PoW

Table 2.1: Table to compare Blockchain Frameworks

Chapter 3

Chapter Three: Problem Definitions and Software Design

The third chapter would focus on the problem definition such that the first section is dedicated to the technical approach and software design. the second part digs more in details and explains how the development process would it be and talks more about front-end and back-end communications.

3.1 Problem Definition

The aim of this thesis is to **Design a Blockchain Network of Universities**. It breaks done to create a blockchain in which all the transactions represent the university exams passed by the students and their grades. Transactions must only be created by authorized personnel. Students read their career history, administrative staff and professors can record grades.

Regarding the described goal, at the first step, a distributed network should be created which would provide also the blockchain data structure and functionalities. After then that a distributed application should be developed and executed over the distributed network. in the end, all the universities functionalities would be added to the application.

3.2 Software Design and Software Architecture

In this section, the architecture of software and all the required software engineering diagrams would be explained.

The software is designed in three main services are participating with together to provide the main application by dividing the tasks between themselves. On the bottom layer, the blockchain and its file storage have existed then in the middle layer the platform is placed to handle the incoming and outgoing requests. The top layer is a user interface for interacting and exploring data.

- Blockchain micro service: At the bottom level of design, there is a network layer that connects different nodes to each other and they would be synched and manage the blockchain ledger by an API layer which is placed on top of the network layer.
- Platform and Security micro service: at the bottom is a database for general information and accounting then an API layer is existed in between to connect to security layer. the security level generates token for each account and approves the authentication for the main functionalities of the blockchain.
- UI/UX: a user interface for different sort of users to explore the blocks and eligible users could execute a couple of comments through the chain.

Networking and Application Architecture of blockchain 3.1.

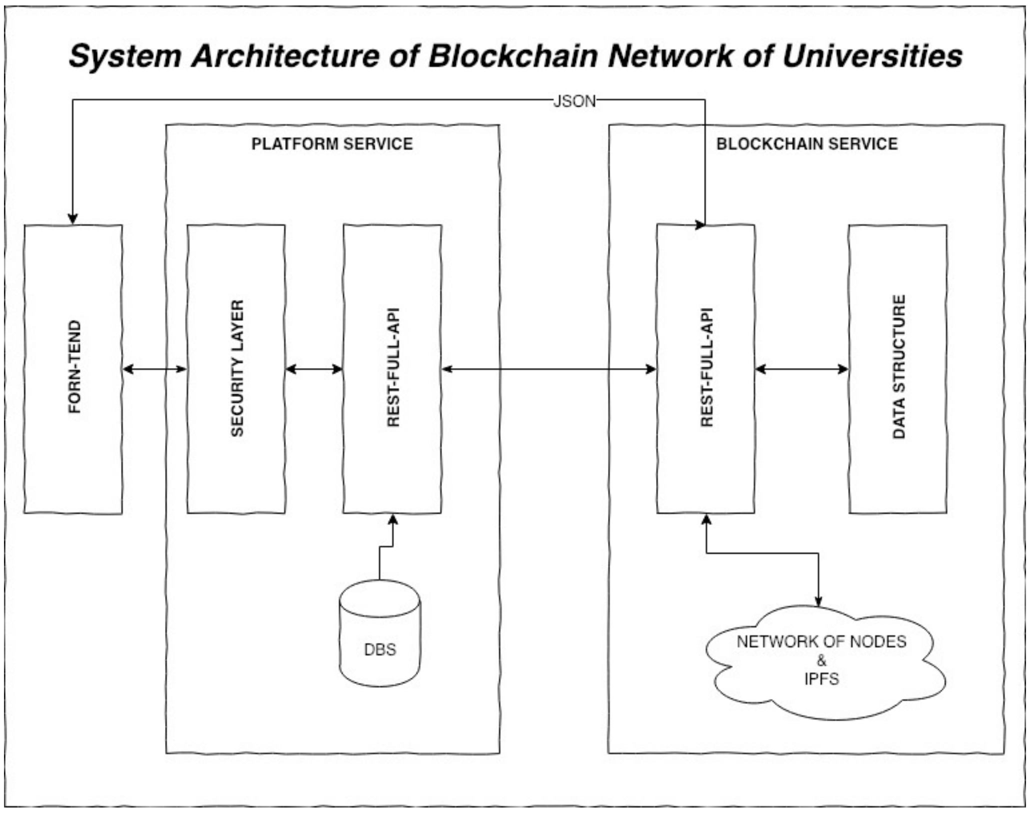

Figure 3.1: Software Architecture and API Layers.

3.3 Block of Data

The data block format mainly divided into two parts the first one that every different blockchain have is some basic information like index and nonce (are explained in chapter one). The second part is dedicated to using case of blockchain and customized by desire application.

Here is the proposed and applied data structure for student application such that every block of the chain has the same format and stored after mining by a node.

- **Transaction**

  A transaction is the tiniest entity in the blockchain and when it stores it would never change. This entity would carry some key information that depends on the desired application. In this thesis, each transaction has one recipient that who is the student, one sender who is the professor of a specific course. Course id and mark are assigned to the student.

  - **Student (Recipient):** It shows the student that who is involved.
  - **Professor (Sender):** It shows the professor that who has registered the mark for a student and couldn't change it.
  - **Course Id:** It shows the course that evaluated for a student.
  - **Mark:** It shows the final mark of student and would never change.

  All the transactions of implemented blockchain in this thesis have the above data format and other functionalities are implemented based on this basic data model.

- **PendingTransactions** [**Transaction**]

  List of all transactions which are arrived and established through the network since the last block was mined. This list of transactions is called pending transactions. When a node starts to mine a new block, it would fetch all the transactions one by one from the pending list then generate the hash value and broadcast the new block. At this point that all the partner confirms the new block then the pending list of all nodes of the network would become empty. After that, the chain is ready for new incoming transactions.

  Besides, in this thesis, all the pending list of all nodes are always the same and updated continuously to prevent the unwanted data errors.

- **Chain [Block]:**

  A chain is a list of blocks are linked as one-way backward link list such that every node just has a hash value of the previous block.

- **Block:**

  Block of data contains a couple of values are evaluated when the mining process done by a node.

  – **Index:** This attribute indicates the index of a block in the chain and it is just once used and a unique and ordered value, which is allocated locally by a node during mining of block. Since it is possible for different nodes to start and finish mining at the same time, there is a concurrency problem between them to allocate this unique and ordered value.

    The applied technique to deal with concurrent mining is like a FIFO queue, for example, when a node wants to start a mining process, it would use its own chain and get the length of chain and increase it by one. Then it would finish the mining process at this point it must broadcast the new mined block through the network to all other partner nodes. When a new block arrives, the node would start to sort the nodes based on its index and drop the older one and just accept the lower index. In meanwhile the consensus algorithm should confirm the arrival block.

    is accepting and dropping based on their second attribute that called timestamp. after broadcasting the block to all partners.

  – **Timestamp:** Each block contains a unique time timestamp. During the implementation of the application, this value would be generated by a miner node after finishing the mining process and it would be attached to rest of block data as root level value.

    Besides, since this unique value suggested to be used in the encryption of data as a source of variation for the block hash, which makes it more difficult for an adversary to manipulate the blockchain.

  – **Nonce:** A nonce is a number used just once for a single block that serves to secure the chain. this means that the same encrypted value can't be submitted to the server again. Even with the same data, the server from any source of the sender or target recipient will expect a distinct value due to the modification made by the nonce. In this case, if an attacker can get the nonce it still only works for a single block so it prevents replay attacks.

    The once used to be started with 0000 and randomly reproduced by a random generator. The way of using the nonce in this thesis more explained in proof of work.

  – **Hash:** it is the hashed value of the block. in another word, all the transactions are put in a single hashed string. In this thesis, the hash value is generated by a function that receives a block of data and converts to a fixed length hashed string and it looks like as a random string.

    The applied hashing function is SHA256, which takes a string and convert to another hashed string. It does two functionalities for each hash value. First, it gives a different hash string for different values and second, it gives a unique string for every input string.

  - **PreviousBlockHash** It shows the hash value of the previous block then current block knows where does it come from. Every block is linked to the previous one until reaching the genesis block. That a way to seek over the chain and make sure all the data are stored and not changed.

    As it is explained before for Hash property, the hashed value is generated by SHA256 and used in the next block to make a link between two blocks of the chain.

  - **Transactions [Transaction]:** List of all transactions which are mined be a miner node and established in this block. In this thesis transaction is a mark registration for a student that is clear who is the professor and who is the student and what is the mark. After mining a block there is no way to change on it.

- **NetworkNodes [Node IP Address]:** it is an array of node URLs which are participating in the network. In this thesis, it is a list of all universities are going to share their data together. The new node can be added to this list during working time of blockchain and it would automatically get all the information to have the same chain as others.

3.4 Genesis Block of Chain

Genesis Block is the first block in a chain. Every blockchain must have one and start off with this bloc. This specific Block has some initial values that make it different than others like 0 as block value, 0 as a previous link and etc.

Besides, this block is created by every node locally and it would never broadcast through the network the reason is all the node should use the same values to create a genesis block then they would have the same block. the only value that is different is the Timestamp of creation. while this block is just checked point to understand the starting point of the chain, the time difference does not matter.

Therefore, the following data format is used for the genesis block of the chain in this case, which contains arbitrary values:

- **Nonce: 100** , while for the other blocks would start with four zeros as prefix.
- **PreviousBlockHash: 0** , the reason is that there is no other block before than that. Also, this value helps to check that is the current block is the first block or not.

- **Hash: 0** , the hash value of genesis blocks it does not matter then just a fake value assigned to that.
- **Timestamp** , generating time, it is the time that the first block is generated and might be different from one node to another one.

3.5 Functionalities of University Blockchain

The university blockchain has a bunch of dedicated functionalities is developt only for it. while they would be consistent with the base model of the standard blockchain.

Proof of Work

Proof of works method is a very important and essential method for blockchain technology and because of a good implementation of that bitcoin and others are very secure. For making a blockchain to be sure that the data stored well and established transactions would never change, this aim will be achieved by the proof of work.

But what is it actually is used in this thesis? it takes the current block of data and previous block hash and generates a specific hashed string. The following method is used for proof of work for final implementation:

- Results want to have a hash which has **0000** at the beginning then for reaching that function would start with **nonce=0** and increment to reach the point
- Using current block data for hashing, even though the previous block hash is used
- Continuously changing the nonce value until it finds the correct hash
- Return to use the nonce value that creates the correct hash

Above rules consuming a lot of time and energy because it has a huge amount of computations. then if somebody wants to back to blockchain and tries to change a block or adding some data in that block, it needs to use a lot of calculations and energy to create the correct hash. trying to coming back and recreating some of the already existing blocks or trying to mind with its own fake data is not feasible.

Besides, since proof of work receives the previous block hash value and mixes with the current data block, all the block hashed values are linked with each other by their data. consider that after mining this block, all the next block must be recreated which need an incredible amount of energy. this is the reason that the blockchain technology is so secure.

Adding a New University node to network

In this thesis, a university has the same meaning like a node of network such that every node has a unique address. With respect to implementation, the assumption is that there is already a network of connected nodes and every node has a copy of the chain and would try to keep them always the same. Every node has its own Genesis Block which is created by a node locally. The Genesis Block has the same data structure and values as other blocks, with special values to be able to distinguishable. it can be used for some security checks and always should be careful about these parameters to prevent unwanted errors.

From data point of view, in the root of blockchain data structure, there is an array of network addresses that is the same for all nodes(universities). This array would be always updated, for example, when a new node joins to the network a new instance would be added to the array.

The sequence of adding a node would be started by the blockchain administrator, who is the only one that can add a new node to the network. The administrator could sign into the platform controller by passing an authentication process. After that, it would send the address of the new node with a secured request to the blockchain network API. The network-API endpoint would validate the incoming request by request header and checking the JSON web token. In the end, it would insert the new address an array of network addresses. As it is mentioned before, the array of nodes (universities) presents in the root of data structure and it is called networkNodes. More details about processes are in chapter four.

Broadcasting a Mark Transaction through Network

The main idea of the chain is to store the marks of students and would generate a certification for them automatically.

The process of mark registration would be started by professors. Every professor would be created by the administrator. Then they could sign into the platform and search for a student by name or student id. After that professor would select the desired student and fill up a form that contains the course and mark value.

When the professor submits the form an XHR request would be created and posted to network API. This request contains the form values and token authentication of professor, and it would be validated before broadcasting through the network.

At the end of the journey, when the request enters to the endpoint, it would create a transaction by form data and store it in its local chain. After that, it would broadcast through the network to all the other nodes and they would update their own chain when consensus algorithm confirms the transaction. More details about implementation are in chapter four.

Handling a Pending Transactions

All the transactions of the network before mining and storing in the chain, they would be collected in a list pending transaction. These transactions are valid until one node decides to do mining then it would clear the pending list.

Every pending transaction is the same data format as the mined transaction has. The only difference is that they are mined and stored and would not be missed.

Consensus Algorithm

Every node in the network must ensure that the next block in a blockchain is truth version of the block, and it protects against unwanted changes and successfully forking the chain.

In this implementation of the blockchain, the longest chain method has been chosen to confirm the truth chain. the longest chain considers the longest chain as the right version and would replace the current when with longest one. this algorithm sends a request to all nodes in the network and asks for their copy of blockchain to compare them with the one in the current node. The theory is that if a group of miners is working simultaneously on the same chain, then only one will grow fastest or finish. Everyone that who is mining faster will be the longest and most trustable version of the chain. More details in chapter four are about the endpoint and how the communications are done.

Mining a New Node

The creation of a block called Mining block and all the pending transactions of a network should settle down. the point is that after creating(mining) a block all the pending transactions of blockchain would be removed to be prepared for the next block. Besides, all the nodes of the network should be pinged and synchronized with each other to be sure that they received the new block and update their ledgers.

At the first implementation this ability was allocated only to admin and then it extended to nodes (universities) to able to mine a new block. The node would call an endpoint via a secure and valid XHR request and network-API would authorize the request after that it starts to mine the pending transactions and encrypt them via SH256. When the mining process finished, the mined block would be broadcasted through the network and everybody would use the consensus algorithm to update its own chain by longest chain logic.

Exploring a Certification

The platform layer provides the user interface for the blockchain. In this thesis, this part called public area, which is a place that administrator, teachers and other users can communicate with the main chain over the network. For simplicity in environments management, different nodes forwarded to a specific port number.

All the public users can work with platform layer by a REACT interface. For exploring a certification of a user, the user could open the home page of platform application and there is a text box as a search keyword. This search keyword would post to platform layer to find the matched students. After selecting the matched student another request would contain by the student id and post them to a network layer. This request can be sent to any one of nodes (universities) because all of them have the same copy of chain. In response to the second request, if that student has any sort of transactions, there are marks of a student in different courses. It would receive all the transaction that recipient is that specific id. Then a layout is de4signed to show all of this information in certification format, which contains all the student information and his/her marks.

Security and Authorizations

In terms of security in this implementation, two main aspects are considered and tried to be covered.

First one is related to REST-APIs Authentications. Since there are two API layers one for a platform and one for networking, all the XHR requests are want to write some data over chain must contain two tokens in its header. One is a JSON web token for user authentication, which is generated during the sign-up process and stored in the platform layer database and is valid for 356 days. Another one is for networking takes, which is an encrypted version of the first one with a private key. this token is valid only for 1 minute after that would be expired.

With respect to the private key, the IP address of the source of the request considered to play the role of a private key and the user JWT is encrypted by private key by an encryption algorithm. All the nodes of networks are aware of that algorithm.

The second aspect that covered in this implementation is encrypting the previous hash block by SHA256 Algorithm. Every block must have a link to last mined block of chain this link must be secure to protect the chain against unwanted attacks. The hashing algorithm would receive the last block of chain and converting all data to a long-hashed string. This hash value is assigned to the link parameter of a new block which is called PreviousBlockHash. This algorithm is called during the mining process and the complexity of that would have a more secure chain.

Security and Authorizations

In terms of security in this implementation, two main aspects are considered and tried to be covered.

First one is related to REST-APIs Authentications. Since there are two API layers one for a platform and one for networking, all the XHR requests are want to write some data over chain must contain two tokens in its header. One is a JSON web token for user authentication, which is generated during the sign-up process and stored in the platform layer database and is valid for 356 days. Another one is for networking takes, which is an encrypted version of the first one with a private key. this token is valid only for 1 minute after that would be expired.

With respect to the private key, the IP address of the source of the request considered to play the role of a private key and the user JWT is encrypted by private key by an encryption algorithm. All the nodes of networks are aware of that algorithm.

The second aspect that covered in this implementation is encrypting the previous hash block by SHA256 Algorithm. Every block must have a link to last mined block of chain this link must be secure to protect the chain against unwanted attacks. The hashing algorithm would receive the last block of chain and converting all data to a long-hashed string. This hash value is assigned to the link parameter of a new block which is called PreviousBlockHash. This algorithm is called during the mining process and the complexity of that would have a more secure chain.

Summary

In this chapter, the main goals of the thesis and the journey to achieve them are explained. As it is described, the objective is to Design a Blockchain Network of Universities such that all the transactions represent the university exams passed by the students and their grades. Transactions must only be created by authorized personnel. Students read their career history, administrative staff and professors can record and change grades. Besides, different aspects of development regarding networking, sending and receiving the transactions of students or professors, or how to extract data from blockchain and etc, all are discussed.

Chapter 4

Chapter Four: Application Development

The aim of this to describe the implementation of a full blockchain prototype, which is hosted on a decentralized BC network. Explaining how to implement a whole network for a blockchain such that a couple of nodes are distributed through a network and they would be able to synchronize themselves and manage their ledgers.

4.1 Development Strategy

Implementation and coding of the final application divided into three main steps are done almost simultaneously. Two steps are related to backend tasks and one for frontend tasks.

The plan is to keep only the mark information in blockchain and keep the rest of the information of students of the blockchain data structure. The reason is that this policy would reduce the amount of data is going to be carried by the chain and reduce the size of the block. Also, it would have a significant effect on computation time of mining. There was another solution like keep other information in another chain and keep a mapping between these two chains or using hieratical chains. Since the rest of the information is not critical and possible store in a simple database and just reveal them when they are required, then the second layer of backend added to manage these pieces of stuff. This cover the risk of losing data and missing some portion of them.

Based on the above strategy, the backend part has two independent APIs one for networking and communications between nodes of blockchain and another one are dedicated to the blockchain-off call to actions like tokens generation. More details would come later.

Frontend part mainly provides a block explorer for users to communicate with chain and a user interface for administrator and professors to do their actions. In next sections would see that some actions like submitting a mark are available only for administrator and some actions are available for administrators like adding a new node to the network.

Development steps are as following:

- Defining a data structure for BC
- Build a REST-Full API for BC
  - Building a decentralized BC network
  - Defining and Implementing a network consensus algorithm
- Building REST-Full API for platform layer
  - Database communications
  - Authentications
  - Blockchain synchronizations
- Building a block explorer application (UI/UX)

4.2 Stack of Technology

The stack of technology in frontend includes REACT-JS which is a JavaScript extension and ES6 version are applied. In backend side, the development language is Node-JS which is a JavaScript language for server side. Node-JS used for both layers (Platform and Networking functionalities via HTTP). As a data storage in the platform layer, MongoDB is used for blockchain-off data and Inter Planetary File System (IPFS) used for blockchain data management.

With respect to version control, the GIT is used with Bitbucket which provides a practical tool for communications and controls different versions of the software. The number of branches from the master branch is 30 (still might be increased).

4.3 Blockchain and Network Functionalities

The blockchain data structure is explained in details in the previous chapter and here mostly focused on implementations and functionalities. Each Block carries some values and all of them are linked to each other. All the required information for a block covered by creation process of a block. also, they are categorized into three different:

- Block Related Values like the index of a block, link to previous block and creation timestamp.

- Pending Transactions are all transactions since the last block is mined. in other word, start to over the new transactions for the new upcoming block (stores only new transactions for the next block)
- Hash Value which refers to the security information like hash value and nonce.

Creation of a block and communicating with chain and rest of blocks are provided by a bunch of functions like below, which are covering from creation and mining and other actions.

- **function Blockchain():** Defining the blockchain data strucutre.
- **createNewBlock = function (nonce, previousBlockHash, hash):** creating a block for Blockchain by using a prototype function that receives three parameters nonce, previousBlockHash, hash. All the required information for every block should be provided here and preparing for the next/previous link through the chain.
- **getLastBlock = function ():** Returns the last block of chain
- **createNewTransaction = function (mark, sender, recipient):** Creating a new transaction for blockchain has three parameters mark: how much would be sent by this transaction, sender's and recipient's address then all the transactions in this blockchain would have this information
- **addTransactionToPendingTransactions = function (transactionObj):** all the people in blockchain would have transactions to send and receive a mark. all the new transactions are pushed into a list of new transactions of blockchain but they are not really recorded, they would be recorded when the new block mined. besides, flash out the array to be ready for the next block.
- **hashBlock = function (previousBlockHash, currentBlockData, nonce):** a function that receives a block of data and converts to a fixed length hashed string and it looks like as a random string. hashing function sha256, which takes a string and hash it to a string. it does two things, first it gives a different hash string for different values and second, it gives a unique string for every input
- **proofOfWork = function (previousBlockHash, currentBlockData):** the implemention of proof of work that explained in data architucture chapter.
- **chainIsValid = function (blockchain):** Since each block has hash value of itself and previous block, to make sure that the chain is valid, this function would iterate through all blocks of chain.

- **getBlock = function (blockHash):** returns a block by its hash value.
- **getTransaction = function (transactionId):** returns a transaction by its transaction id.
- **getAddressData = function (address):** returns the content of a block with all information by its address.

4.4 Backend Functionalities

The backend part is in charge of all functionalities and communications are required to happen. These functionalities are splatted into two main sectors to reduce the complexity and amount of works. The first part is platform API which has to provides the user authentications and all sorts of interactions between the blockchain and different users. The second one is mainly dealing with the blockchain of universities.

Implementation notes:

- All the requests and responses are sent/receive by request-promise package.
- The body of requests is passed in the JSON data format and parsed by the body-parser package.
- **Database Schema,** Regarding the database storage in **platform layer**, there are three (it might be increased later) different database schemas are implement to cover the main database interactions. These are related to users, professors (called teachers), and students. more details about the attributes and inline functionalities are available in code.

In the next section a bunch of important endpoints are defined and implemented for each API as it is mentioned.

4.4.1 Decentralized Blockchain Networking API

The networking API provides a bunch of endpoints with three main purposes such as block creation, networking communications, and data retrieval. The assumption is that there is a network of connected nodes and every node has a copy of the chain and would try to keep them always the same.

Since each node creates its instance locally then its Genesis Block has the same data structure and values as other, except timestamp then always should be careful about this parameter to prevent unwanted errors.

Intra Node communication endpoints

The first step is that a node should be able to communicate with its own local blockchain. The following endpoints are proposed for primitive tasks:

- **get(/blockchain):** It would send back the entire blockchain with whole information about all blocks, block transactions and chain information like pending transactions and network address.

- **post(/transaction):** It would post or create a new transaction. the body of the request must carry transaction information of professor id(sender), student Id (receiver), course Id and mark. when the new transaction is created. it should be inserted into the list of pending transactions of its local chain by calling addTransactionToPendingTransactions(newTransaction).

- **get(/mine):** It would mine a block for chain and give a reward to the node.

For mining of a block, first of all, the current node has to get the last mined block of chain to know that who is the previous block and it would make a link to it. This functionality is provided by getLastBlock() , after that a new block should be created with all the currently pending requests. The important point is related to proof of work which guarantees the security of blockchain and makes sure that nobody can touch it. When a block is created the current node should prepare a request to aware all the other nodes of the network to update their local chain with a new block.

In this section all the details about Building a Decentralized BC Network API endpoints are categorized in four different types and explained how they are communicating with each other. therefore, to reach to this aim a set of nodes in different places are connected to each other through a network and all of them has the same copy of a blockchain and locally doing some activities then synchronizing together to be exactly the same. While there is no centralized point to manage the network and all the nodes have the same capabilities. API's endpoints for creating a decentralized network divided into node and block synchronization.

Node synchronization endpoints

- **post(/register-and-broadcast-node):** every new node should be registered and then broadcasted through the entire network then all nodes would be aware that a new node is added and should upgrade their network routing address (node URLs) by hiring /register-node and /register-nodes-bulk endpoints.

- **post(/register-node):** all the already existed nodes should register in the local chain of the recently broadcasted node. In the figure 4.1 it is shown how to add a new node and broadcast it for building a Decentralized Network of universities.

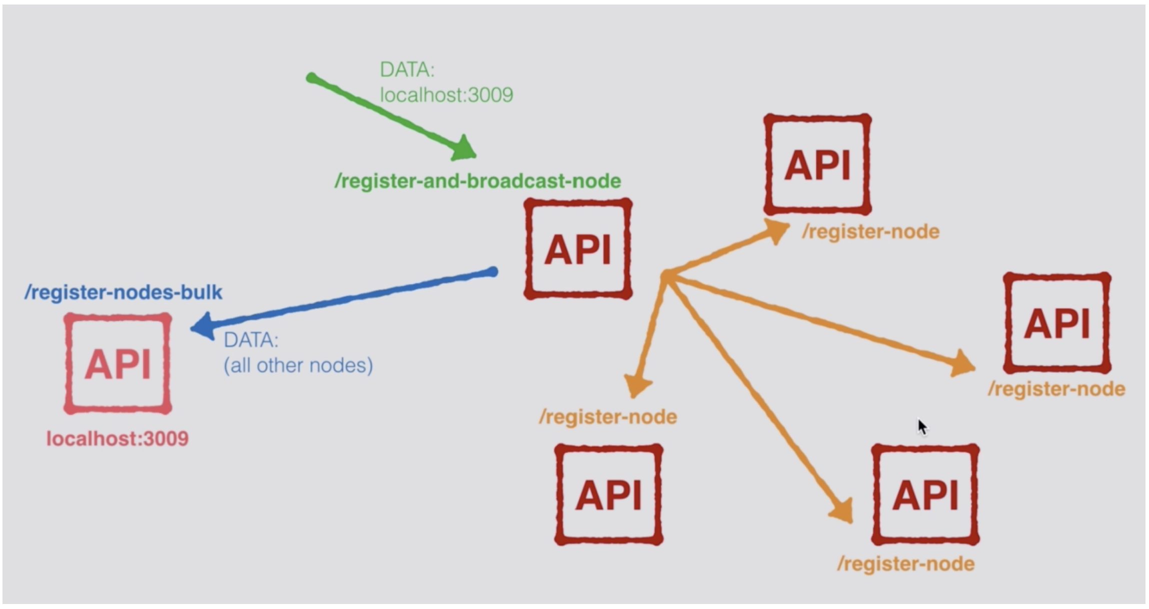

Figure 4.1: Registering and Broadcasting a new node to a decentralized network of universities.

- **post(/register-nodes-bulk):** It would register multiple nodes all at the same time in the network if a node does not already exist and it is not the current node.

There might be a question like "how to register a new local node in the decentralized network". The local node should hit a node at the network (any node of the network) to register-and-broadcast-node. Then the targeted node should send the URL of the new node as the body then all the other nodes of the network will receive a broadcast message on their /register-node endpoint and they should register the new node. also, the receiver nodes don't need to broadcast it again. in the end, the node who did broadcast should send the URLs of other nodes to the new node and hit the /register-nodes-bulk.

(broadcast-a-node.4-1.png)

Block and transactions synchronization endpoints

- **post(/transaction/broadcast):** It would be hired whenever, it supposed to create a new transaction. First, it creates a new transaction with specifications and then will broadcast the transaction through the network for all the other nodes

- **post(/receive-new-block):** it would save the new broadcast block through the entire network and would check that the block is fine.

the image 4.2 shows how broadcast a new transaction through the network to make sure that all the nodes have the same list of transactions.

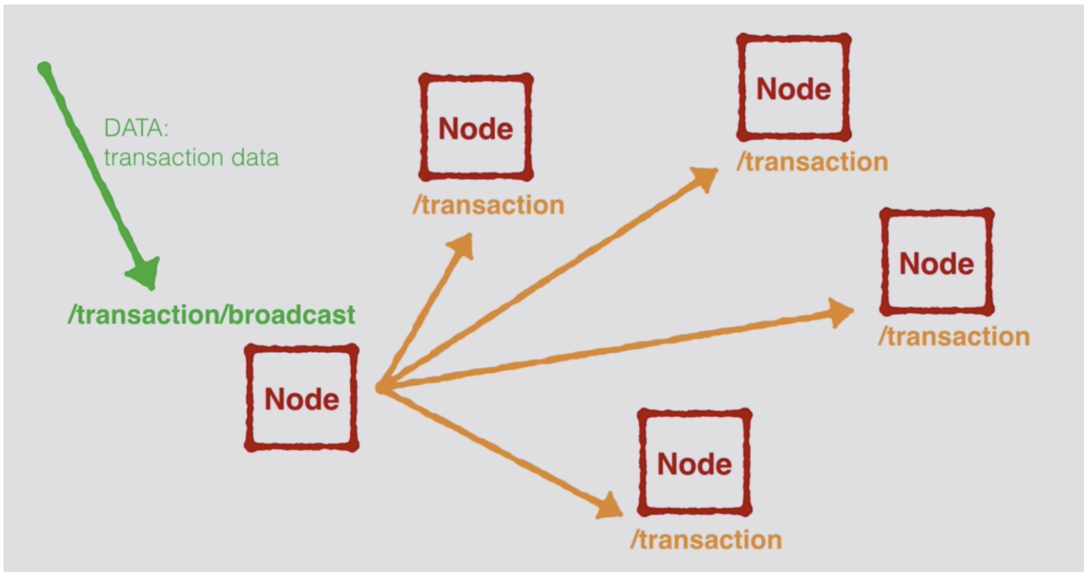

Figure 4.2: Broadcasting a new Mark Transaction through the entire Network.

Defining and Implementing a network Consensus Algorithm

Every node in the network must ensure that the next block in a blockchain is a truth version of the block, and it protects against unwanted changes and successfully forking the chain.

In this implementation the longest chain method has been chosen to confirm the truth chain. the longest chain considers the longest chain as right version and would replace the current when with longest one. this algorithm sends a request to all nodes in the network and asks for their copy of blockchain to compare them with the one in the current node. When it found the longest one can replace. More details are in previous chapter and implementation verion of consensus algorithm is available through an endpoint:

- **get(/consensus):** It asks a copy of the block from all nodes of the network and compare them with itself and clone the longest one.

Building a block explorer application (UI/UX)

The block explorer is a user interface just to display the entire network to make it possible to retrieve the data and check the blockchain.

- **get(/block/:blockHash):** It sends a block hash to retrieve the entire block.
- **get(/transaction/:transactionId):** It sends a transaction id and expects to receive the correspond transaction id.
- **get(/address/:address):** It sends an address and expect to receive the correspond transactions of that address and some more data.
- **get(/block-explorer):** It sends back the html file to render the interface.

4.4.2 Platform and Authentication API

The endpoints of the platform are designed to support three different types of requests and these requests are working with blockchain-off data that doesn't need to be stored in blockchain and would reduce the complexity of implementation and reduce the amount of data over the network. To serve the user sign up and sign in and sign up the following endpoints are used.

- **post('/users/signup', function (req, res)):** receives the sign up information and store in MongoDB.
- **post('/users/signin', function (req, res)):** receives a request that carries the email as username and a password. Also, and simultaneously creates the token and send back to source.

To serve the activities related to students interactions interface the following endpoints are used.

- **post('/student/create', passport.authenticate('jwt', session: false), function (req, res):** receive the information of a student and would create it. The point is the request must carry a valid JWT-Token to pass the authentications.
- **get('/student/get-list', passport.authenticate('jwt', session: false), function (req, res)** returns the list of all students. it is mandatory to carry the JWT.
- **post('/student/search', passport.authenticate('jwt', session: false), function (req, res):** receive a keyword that can be name or an Id then sea4ch through the database and returns it.. it is mandatory to carry the JWT.

- **post('/student/search-public', function (req, res):** same as previous one without authentication. It would return limited amount of information.

To serve the professors interactions interface the following endpoints are used.

- **post('/teacher/create', passport.authenticate('jwt', session: false), function (req, res):** creates a professor with required parameters and authentication is mandatory.
- **get('/teacher/get-list', passport.authenticate('jwt', session: false), function (req, res):** returns the list of professors and authentication is mandatory.

As a quick note, the data validation is implemented in MongoDB and mainly supported by that.

4.4.3 Testing Endpoints

In this thesis, it has been tried to develop the blockchain and platform features based on the test and delivery. the testing framework is Mocha Chai which is quite famous and suite for test developments.

The following tests are developed for blockchain and its networking functions:

- **Test-chain-validation:** testing the chain validation to have a right data structure by sending a mock data of a sample chain to chainIsValid(BC1.chain) function. it expects to receive an error in case of wrong data or confirmation in case of right data.
- **Test-creating-a-block:** testing the creation of a block by passing an object of mock data with all the block requirements to createNewBlock(obj) function. It expected to receive a built block if all the required data are sent correctly otherwise it would receive an error.
- **Test-creating-a-transaction(test-adding-mark):** testing the creation of a transaction. it creates a block first then would call the createNewTransaction function for a couple of times and checking to have a bunch of correct transactions there.
- **Test-genesis-block:** testing the genesis block by tacking an instance of blockchain that would create a genesis block by itself.
- **Test-hash-function-sha256:** testing the hash function of sha256 by creating a block with a bunch of transactions and then testing the hashBlock function.

- **Test-proof-of-works:** testing the proof of works by creating a new block and calling the proofOfWork function and passing the current block and hash value of the previous block.

The following tests are designed for the platform layer:

- **Test-sign-up:** it does test the signup of administrator by passing email, password and telephone number.
- **Test-sign-in:** it does test the sign in bypassing the email and password and in response would receive a token if the sign in was successful or would get an error in case of failure.
- **Test-reload-blockchain:** it is one of two-step XHR tests that first call the platfor4m layer then it would call the blockchain layer to get the whole block.
- **Test-adding-node:** this test tries to add a new node to the network by sending an IP address of new node if the new address does exist and respond then it would be successfully added to the network otherwise it would get an error. This test, also, is done to step because it required to connect to the blockchain layer as well.
- **Test-adding-student:** testing the platform endpoint to add a new student.
- **Test-adding-teacher:** testing the adding teacher endpoint.
- **Test-search-student:** testing the student search by passing a keyword to backend and rec receiving list of matched students.
- **Test-geting-certifaction:** testing the platform and blockchain with two steps request and expected to receive the marks of student from blockchain and build certification from platform endpoint.

4.5 Frontend Functionalities

The frontend of application is designed by REACT-JS. Every feature has been developed based on the atomic and container component design approach, which depends on its complexity and functionalities. In total around 46 components are implemented to cover almost all around the project. Also, Based on the tasks that each component does the frontend components can be divided into three categories first is Layout components which are mainly providing the layout of pages for public pages and logged in page. the second one is related to Error handling components which are handling the server page errors and some exceptional errors. the last category is Block explorer components which are a list of components to interact with blockchain and providing the user interface for it.

4.5.1 Developing the Layout Components

The application is designed to have three different layouts which are based on the users and their actions such as Administrator layout, Professors layout, and Public layout. All of them and their most important functionalities are explained in the following section.

Administrator layout components: the platform and chain administrator can sign in with their email and password by passing a valid token to the sign in endpoint. the following actions are provided for administration control panel:

- **Adding a new node to network,** only the administration has access to add a new now node. This form has one text field to write the IP address of new node then it would submit the request via HTTP with a valid token of an administrator. This request would be validated at the platform layer. After than that, it would send a request to networking API to add and broadcast the node. The point is to test on a local machine with simulating the network by NGNIX every node should be mapped into different ports and this port is part of IP address and must be posted to the endpoint.

- **Adding a new professor,** is another main functionality that the administrator can do it. It has a form that the administrator has to fill it up with the information of the professor and then send it to the endpoint. As a quick note, at the begging the software designed to have one more use as a university employee. It is not applied because for too many users and high amount of complexity and fewer functionalities then the employee is removed and this function, adding a new professor, assigned to the administrator.

- **Adding a new student,** is a post form that gets the information of a student and then sends them to endpoint in the platform layer. If the incoming data pass the validation it would be added to the list of already existed students and available for the professor to search and add a new mark or in block explorer to search for a certification. The point here is, the student id is a hash value which is generated automatically when the student created and it is a unique value.

- **Mining a block,** the administrator can execute a mining process over a specific node. By sending the request to network and firing and mining endpoint. The IP address of the source of the request is mentioned in the body of request and mining would be done by that node. In local just mapping over the NGINX is enough while in the real network the request from desire source node would be broadcasted.

Professors layout components: the professor who is the only one can register a mark for a student and all the other parts of the platform would be able just to use them. The point is there is no chance to modify the number after registering the mark then the professor must be so sure when he/she is going to submit a mark for a student.

- **Search for a student,** which is the basic function for a professor that wants to register a mark. The professor can search by the name or student id. The list of matched students with search criteria would be displayed in a table.
- **Adding a mark for a student,** when the list of students displayed on the table, there is a call to action link for each student that the professor can click over it and a modal of mark registration would be open. After than that, the professor id, student id is already filled up and the professor can select the course and enter the mark value. In the end, it can submit the form to save the transaction over the blockchain.

Public layout components: the public part of the platform is an interface for visitors to come over there and use the block explorer for querying. in terms of development approach all the public components are developed exactly the same as other parts with one big difference which is there are no token and authentications on its requests.

- **Search for student** which is generally the same as explained in the professor's layout while it does not send the token in it has access to request the read-only endpoints, for example, it is possible to fetch a certification of a student. also, it is impossible to register a mark via search in the public area.
- **Viewing the list of marks of a student** when the search request processed and responded by the server, this response contains a list of marks for a student who is matched. this list of marks is displayed in a table of course as a part of certifications. besides, each course is displayed by a name and mark for each row.
- **Viewing the certification of a student** which is a mock-up template that shows that the searched student is graduated or not and to which university does he/she belongs. also, by using the previous section, it is possible to figure out what are the list of courses which are passed or not passed. the point is this certification is generated based on the block information of a blockchain which is not possible to change them.
- **Sing In** the Administrator and professors can sign into the platform. this process is provided by sign in component which takes the user name and

password ans send them to the server. then it waits for a response from server to verify the authentication request or to reject it.

Every one of the layouts has a bunch of other small functionalities which are not mentioned above while they are available in the final code and design.

4.5.2 Block Explorer Components

The most of the components are mentioned in public layout that they work with blockchain to retrieve data. While any of them can be categorized as block explorer components. The reason is that these components fetching data from blockchain and displaying on the screen. also, there is a group of components that they work in between to serve other requests as follows:

- **Reloading the chain,** It is a function that reloads the chain to get the last updated version of data and display in a proper way on the screen. This function would call an endpoint without any authentication and that endpoint replies by a JSON message that contains all the blocks of the chain, list of pending transactions, and chain configuration.
- **Transaction explorer,** Since every transaction carries information of a registered mark for a student, there should be a small function that retrieves only this small data. this job is done by Transaction Explorer that get an Id, or name of student or course name then returns the transaction information.

Besides, a bunch of other functionalities is developed that is not mentioned here and for more details about them, you can see in the code.

4.6 Error Handling

The error handling is managed in fronted and backend in a similar way with a bit of difference. for example, the validation error is provided by data models while in meanwhile the frontend tried to prevent sending the wrong format of data as much as possible. therefore, errors are divided into three main categories such as

- **Throwing Exceptions:** an error (predefined or user-defined) can be raised using the throw statement. it is used when the error is happening in a critical section.
- **Exception Handling:** which is accomplished with a try...catch statement. it is applied in most of places to avoid the crash of application.
- **Customised Errors:** showing the error message based on the runtime like 404, 504 and others. In this case, the error message is customized based on the targeted event by using throw a new error.

Regarding the HTTP requests, the promises package is used which gives us a cleaner way to handle errors. Instead of having to handle errors for every single operation, it is possible to clean up this code by doing it at the end of multiple operations. it also provides the HTTP error handler for many promise-based routes. when there are two HTTP routes that are just basic CRUD operations over our database. because instead of handling all errors independently, the application wants to handle all errors in one single function. in the end, this error handler used in the routes to have a single global error handler, so all error handling logic will live in the same place.

Chapter 5

Conclusions and Future works

The blockchain is used in different sectors and mainly in the finance market and all sorts of the applications which are developed by blockchain are allowing people to secure their digital relationships more than it was before. Data is being disclosed differently, secured differently and recorded differently. This concept is changing digital relationships, creating the ability for them to be automated in code via smart contracts and etc.

At the end of this implementation of thesis we got the following results:

- **It is possible to authorize the certification of graduated stu-dents by blockchain technology**

  it means for different universities to share their public data together by creating a decentralized blockchain network of universities and sharing some services and databases together that would be also possible to make them available for the public.

- **It is possible to expand this case study for other sectors**

  it means not only the student certification even though all the other sorts of a document can be stored and verified by using blockchain technology. Some services such as declaration letters or any other documents that somebody needs to verify a piece of information could be a fitted candidate service to publish via a block explorer of a blockchain.

- **Do not store all the data in chain**

  it means by storing some data off the chain, mainly low and second-order priority information. it would help to reduce the complexity of the application and it would make the development process to be faster and easier, meanwhile, it would not have a significant effect in security and losing data. it is like a trade-off but a good option for productivity enhancements.

Future Works

As future tasks, implementing the same functionalities or services in other sections would help people to improve the performance of their workflows in universities or any other organizations. It would right task to work more on expanding the functionalities and enhancing the capabilities of developed application mainly in terms of data security and network security management. Also, the recalling the network or application is challenging transitions period especially in blockchain because it would produce a huge amount of data and not easy to maintain then working in this area would be a good candidate for future works.

Appendix A

Coding Style and Review

Execution Scripts

A bunch of scripts is provided that by executing them in a proper place, it would be possible to run the application and use it in the right way. In the backend side, two scripts are provided one for executing the application test and another one is for launching a new node. Blockchain Network and Application Execution Script A.1.

```
"scripts": {
 "test": "mocha dev/test",
 "node_platform": "nodemon --watch dev -e js dev/app.js 9000 http://
     localhost:9000",
 "node_1": "nodemon --watch dev -e js dev/app.js 3001 http://localhost
     :3001",
 "node_2": "nodemon --watch dev -e js dev/app.js 3002 http://localhost
     :3002",
 "node_3": "nodemon --watch dev -e js dev/app.js 3003 http://localhost
     :3003",
 "node_4": "nodemon --watch dev -e js dev/app.js 3004 http://localhost
     :3004",
 "node_5": "nodemon --watch dev -e js dev/app.js 3005 http://localhost
     :3005"
},
```

Figure A.1: Application and Network Execution Scripts.

In frontend side just one script for running the frontend of application is provided.

- run: *"npm start"*

Styling of components

The styling of components can be done in different ways like inline CSS, developing by SASS language or pure CSS in a separate file. At this thesis implementation, a combination of pure CSS and building a set of styled components are used to reduce the complexity of the project in the frontend side. The following criteria are considered for implementation:

- **BEM (Block, Element, and Modifier) mythology** is applied wherever a sort of style was required to have a style for a component. This methodology helps to have a clear name convention and understanding of component layout.
- **Building Styled Components** a couple of components are atomic and completely independent are developed as styled components.
- **Styling File Managements** In this implementation all the CSS selectors are defined in a separate file and then imported in component. The advantage is that few amounts of global CSS are attached to page and error handling is easier and faster.

Context API and state management

Context is a new React feature that allows you to share state between components. It provides the access to the context regardless of whether the component is a direct child of the context. While Context is used for global state [19].

Why is Context API used in this thesis?

First of all, it was a trade-off for considering whether to use Context or Redux. Based on the following resaons, the Context API which provides a lot of the same functionality as Redux with some additional benefits, the context API is choosen to be used for state management of application.

- **Complexity** Redux is a powerful tool, but here is only needed a simple shared state such as application alerts or a logged-in user, context is a simpler place to start. The application set state uses a simple setState function rather than a web of actions and reducers.
- **Testability** Since context is built in to the React API, it is easy to test. Also, it will render components using the real ReactDOM API rather than attempting to mock out context.

- **Readability** Because context uses render props instead of a higher-order function, it is easier to see which props are getting passed in to our components, and where they are coming from.

Codde Linter Configuration

ESLint is an open source JavaScript linting utility originally created by Nicholas C. Zakas in June 2013 [20]. Code linting is a type of static analysis that is frequently used to find problematic patterns or code that doesn't adhere to certain style guidelines.

The primary reason ESLint was used in this thesis was to allow us to apply our own linting rules completely pluggable and following the same pattern during the development process. It used for both frontend and backend parts of the code. Coding Linter Configurations and reasons to use them A.2.

```
{
  "extends": "eslint:recommended",
  "rules": {
    "eqeqeq": "error", //Require === and !== (eqeqeq)
    "camelcase": ["error", { "properties": "always" } ], //enforces camelcase style for property names (variableName)
    "no-var": "error",
    "semi": [2, "always"], // requires semicolons at the end of statements
    "no-undef": "error",
    "no-undefined": "error", // Disallow Use of undefined Variable
    "no-unused-vars": "error",
    "no-console": "error", // Aviod console.log/error/warn to be in code
    "no-constant-condition": ["error", { "checkLoops": false }], // Avoids the loops becasue of fixed condition like if(true)...
    "no-dupe-args": "error", // input args of function
    "no-dupe-keys": "error", // keys inside an object
    "use-isnan": "error", // use isNaN(foo) to check the value
    "no-eq-null": "error",
    "no-extra-semi": "error",
    "no-empty": "error", // No empty block
    "no-unreachable": "error",  // detect the unreachable code
    "array-callback-return": "error", // Enforces return statements in callbacks of arrays methods
    "block-scoped-var": "error",  // This rule aims to reduce the usage of variables outside of their binding context and emulate traditional block scope from other languages.
    "init-declarations": ["error", "always"],
    "callback-return": "error", //Enforce Return After Callback
    "handle-callback-err": "error",
    "no-unused-expressions": "error",
    "no-useless-constructor": "error",
    "no-useless-concat": "error", //Disallow unnecessary concatenation of strings
    "max-statements-per-line": ["error", { "max": 1 }],
    "no-this-before-super": "error",
    "no-redeclare": "error", //disallow variable re-declaration
    "no-duplicate-imports": "error",
    "comma-dangle": ["error", "never"], // require or disallow trailing commas
    "indent": ["error", 2], // Space distance of each line from left = 2 spaces a tab
    "semi-spacing": ["error", {"before": false, "after": true}],
    "no-extra-parens": "error",  // unnecessary parentheses
    "arrow-spacing": ["error", { "before": true, "after": true }],
    "arrow-parens": ["error", "as-needed"],
    "arrow-body-style": ["error", "as-needed"],
    "eol-last": ["error", "always"], // require or disallow newline at the end of files
    "space-before-function-paren": ["error", {"anonymous": "always", "named": "never", "asyncArrow": "never"}],  //Require a space before function parenthesis
    "comma-spacing": ["error", { "before": false, "after": true }]
  }
}
```

Figure A.2: High Priorities JS Functionalities and Coding Style Rules.

Bibliography

[1] I. Bashir, *Mastering Blockchain*, second edition, 2018.

[2] B. Brakeville, Sloane; Perepa, Bhargav, *Blockchain basics introduction to distributed ledgers*, IBM Developer. IBM. Retrieved 25 Sep 2018.

[3] D. David Mills, K. Kathy Wang, B. Brendan Malone, A. Anjana Ravi, *Distributed Ledger Technology in Payments*, Clearing, and Settlement, FEDS Working Paper No. 2016-095.

[4] Y. Guo, C. Liang, *Blockchain application and outlook in the banking industry*, Financial Innovation, Springer, 2016.

[5] B. BlockchainHub, *Blockchain Explained - Intro for Beginners Guide to Blockchain*, blockchainhub.net, 2019.

[6] S. Satoshi Nakamoto, *Bitcoin white paper: A Peer-to-Peer Electronic Cash System*, www.bitcoin.org, 2008.

[7] M. Piekarska, *Hyperledger Blockchain Technology for Business* , Hyperledger Wiki, wiki.hyperledger.org, Feb, 2019.

[8] B. Carson, M. Higginson., *Blockchain explained: What it is and isn't, and why it matters*, McKinsey Insights, 2019.

[9] X. Olleros, M. Zhegu, *Research Handbook on Digital Transformations* , Cheltenham 2016.

[10] G. Gabsion, *Policy Considerations for the Blockchain Technology Public and Private Applications* , 19 SMU Sci. Tech. L. Rev. 327, 2016.

[11] S. Tabrizi , *A Next-Generation Smart Contract and Decentralized Application Platform* , Ethereum white paper, ethereum.org, 2016.

[12] B. BigchainDB GmbH , *BigchainDB 2.0 , The Blockchain Database*, BigchainDB white paper, Berlin, Germany, Paper version 1.0, May 2018.

[13] C. Chain Inc., *Chain Protocol Whitepaper*, version 1.2, 2019.

[14] R. Gendal Brown , *The Corda Platform: An Introduction*, Corda white paper, May, 2018.

[15] R. Creighton , *Domus Tower Blockchain (DRAFT)*, Domus Tower white paper, Domus Tower Inc. San Francisco CA, USA, March 28, 2016.

[16] Blockstream , *Elements Blockchain or Sidechain solution*, Elements Project website, 2019.

[17] C. Kuhlman, B. Bollen, *Hyperledger Burrow (formerly eris-db)*, HIP identifier Burrow, the permissionable smart contract machine, 2017.

[18] Coinprism, Inc., *Openchain Documentation*, Revision 49adcb2a, 2015.

[19] Facebook Inc., *REACTJS Context API Management*, A JavaScript library for building user interfaces, 2019.

[20] N. C. Zakas, JS Foundation, *ESLinter Documentation*, A pluggable linting utility for JavaScript, 2013.

[21] B. Yokubov, *Blockchain based storage of students career*, Blockchain based storage of students career, 2019.