DIPLOMA AND TRANSCRIPT VERIFICATION
THROUGH BLOCKCHAIN TECHNOLOGY

Hachem El Alami

Supervised by:

Professor Nisar Naeem Sheikh

Spring 2022
DIPLOMA AND TRANSCRIPT VERIFICATION THROUGH BLOCKCHAIN TECHNOLOGY

Capstone Report

Student Statement

"I, Hachem El Alami, hereby affirm that I have applied ethics to the design process in the selection of the final proposed design. And that, I have held the safety of the public to be paramount and have addressed this in the presented design wherever may be applicable."

_____________________________________________________

Hachem El Alami

Approved by the Supervisor:

[Signature]

Prof. Nisar Naeem Sheikh
ACKNOWLEDGEMENTS

First and foremost, I would like to express my profound gratitude to my family, friends, and the AUI staff and faculty members, who have all supported me throughout these many years. I want to thank my mother, father, and brothers for supporting me through every moment of my life. Additionally, I would like to thank my parents and Al Akhawayn's staff for helping and supporting me during my transition from general engineering to computer science.

Secondly, I would like to thank my supervisor, Dr. Nisar Naeem Sheikh, for accepting to supervise my capstone project very late and, secondly, knowing that he was highly overwhelmed by excessive work and duty overload.

Last but not least, I would like to express my gratitude towards my friend, Amine Laghmich, who has always supported, helped me in times of need, and inspired me to undertake this project by proposing the project idea.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ............................................................................................................. 3

LIST OF FIGURES ...................................................................................................................... 7

ABSTRACT IN ENGLISH ............................................................................................................. 8

ABSTRACT IN FRENCH ................................................................................................................. 9

FEASIBILITY STUDY .................................................................................................................... 10

REQUIREMENTS AND SPECIFICATIONS ................................................................................... 12

  **FUNCTIONAL** ......................................................................................................................... 12

  **NON-FUNCTIONAL** .............................................................................................................. 12

CHAPTER 1: INTRODUCTION ...................................................................................................... 13

  **1.1. Context** ............................................................................................................................ 13

    1.1.1. Short History .................................................................................................................. 13

    1.1.2. The Problem ................................................................................................................... 13

    1.1.3. The Solution ................................................................................................................... 14

  **1.2. Background** .................................................................................................................... 14

    1.2.1. Existing Work .................................................................................................................. 14

    1.2.2. Blockchain ..................................................................................................................... 14

    1.2.3. Ethereum ....................................................................................................................... 16

    1.2.4. Smart Contracts ............................................................................................................. 16

    1.2.5. Programming Languages ............................................................................................ 17

    1.2.6. Bootstrap ...................................................................................................................... 18

    1.2.7. Node .............................................................................................................................. 18

    1.2.8. Truffle ............................................................................................................................ 19

    1.2.9. Ganache ........................................................................................................................ 19
1.2.10. Metamask ..............................................................20
1.2.11. Web3 ......................................................................21

1.3. Methodology ..................................................................22
  1.3.1. Approach ..................................................................22
  1.3.2. Benefits ....................................................................23
  1.3.3. Constraints ...............................................................23

CHAPTER 2: IMPLEMENTATION ..............................................25

2.1. Contract ........................................................................25
  2.1.1. Hashing Method ......................................................25
  2.1.2. Key Association Method .........................................26
  2.1.3. Security Restrictions ................................................27
  2.1.4. Payment Functionality .............................................27

2.2. Backend ........................................................................28
  2.2.1. Node Server ............................................................28
  2.2.2. Routes ....................................................................28
  2.2.3. Uploading Document ..............................................29
  2.2.4. Hashing .................................................................29
  2.2.5. Communication with Client-Side ..............................30

2.3. Frontend ........................................................................30
  2.3.1. Overview ...............................................................30
  2.3.2. Web Design ...........................................................31
  2.3.3. Contract Interaction ...............................................31

CHAPTER 3: STEEPELE ANALYSIS ........................................34

3.1. Social ...........................................................................34
3.2. Technological .............................................................34
3.3. Environmental ........................................................................................................34
3.4. Economical ........................................................................................................34
3.5. Political ...............................................................................................................34
3.6. Legal ..................................................................................................................34
3.7. Ethical ...............................................................................................................34

CHAPTER 4: CONCLUSION .........................................................................................36

References .................................................................................................................37

APPENDICES .............................................................................................................39

APPENDIX A: INITIAL SPECIFICATION ................................................................39
LIST OF FIGURES

Figure 1: How Does a Transaction Get into The Blockchain [4].................................15
Figure 2: Node Work Flow [7]..................................................................................19
Figure 3: Ganache .......................................................................................................20
Figure 4: Metamask......................................................................................................21
Figure 5: SHA-256 Hashes vs. Collision Probability [12]..............................................25
ABSTRACT IN ENGLISH

Blockchain technology and cryptocurrency have recently emerged as an unavoidable revolution in today's world. Today, everyone has heard about bitcoin, and more broadly, cryptocurrency. Blockchain technology has gained momentum due to its core idea of being decentralized. This capstone project aims at riding this new wave by taking advantage of the revolutionary possibilities offered by blockchain technology.

This project aims to use blockchain technology to verify diploma authenticity securely. This will be achieved by creating a smart contract that will be added to the Ethereum blockchain. Additionally, this project will incorporate the latest trends to develop an interface for recruiters to check diploma validity and Al Akhawayn to add their awarded diplomas to the blockchain.

This project will use a range of technology enablers to achieve its goal. A non-exhaustive list includes Solidity, Truffle, and ganache for smart contract development, Node and its various modules for backend development, bootstrap, javascript, and Metamask for the frontend development.

Keywords: Blockchain, Cryptocurrency, Bitcoin, Decentralized, Ethereum, Smart Contracts, Solidity, Truffle, Ganache, Node, Bootstrap, Javascript, Metamask
ABSTRACT IN FRENCH

La technologie blockchain et les crypto-monnaies se sont récemment imposées comme étant une révolution incontournable du monde actuel. Aujourd'hui, tout le monde a entendu parler du bitcoin, et plus largement des crypto-monnaies. La technologie blockchain a pris de l'ampleur grâce à son idée maîtresse de décentralisation. Ce projet de capstone vise à surfer sur cette nouvelle vague en tirant parti des possibilités révolutionnaires offertes par la technologie blockchain.

L'objectif de ce projet est d'utiliser la technologie blockchain pour vérifier de manière sécurisée l'authenticité des diplômes. Pour ce faire, nous créerons un contrat intelligent qui sera ajouté à la blockchain Ethereum. En outre, ce projet intégrera les dernières tendances pour développer une interface permettant aux recruteurs de vérifier la validité des diplômes et à Al Akhawayn d'ajouter les diplômes qu'ils ont livré à la blockchain.

Ce projet fera appel à toute une série de facilitateurs technologiques pour atteindre son objectif. Une liste non exhaustive de ceux-ci comprend solidity, truffle et ganache pour le développement de contrats intelligents. Node et ses différents modules pour le développement du backend et bootstrap, javascript et metamask pour le développement du frontend.

Mots clés: Blockchain, Cryptocurrency, Bitcoin, Decentralized, Ethereum, Smart Contracts, Solidity, Truffle, Ganache, Node, Bootstrap, Javascript, Metamask
FEASIBILITY STUDY

This project aims to produce a system using blockchain technology to verify the authenticity of Al Akhawayn's diplomas and transcripts.

To do so, we will need to use smart contracts. A smart contract, in simple terms, is a program contained in a blockchain that executes when the conditions are met. The blockchain that is generally used for related purposes is Ethereum. However, it can be costly to create a smart contract. Furthermore, a "transaction" takes place for each document verification, which also costs money. Therefore, we will use an Ethereum network simulator to test our project. This will be incorporated using Truffle. Truffle uses the Ethereum Virtual Machine for blockchain application development.

We will have to use a programming language to create and use smart contracts. Even though it is possible to use Javascript, better options are available. Indeed, Javascript is a robust, widely-used programming language. However, it has not been designed for blockchain use. On the other hand, Solidity has explicitly been created as a smart contract programming language. It is the most popular solution. Furthermore, it shares a similar syntax with C++ and JavaScript. Therefore, it will be a familiar language, making it easier to grasp and use.

For the IDE, we will use Webstorm. This is an IDE made by Intellij, with a free license for students, for web development. Even though it is not made for contract development, it can still be used to write solidity code. It is handy for the whole contract development process because it can run a terminal from it directly. Additionally, it is convenient to use it since the project is JavaScript-heavy-based.

Finally, it can both run on the browser or locally. We can use an average computer since Truffle, Webstorm, and Solidity do not require heavy computing power. Finally, the time constraint should not be an issue. The estimation of one month to learn, understand, and test the implementation of smart contracts is more than realistic. Afterward, the implementation should
not take much more than one month. Therefore, there is about one spare month to accommodate any unseen event.
REQUIREMENTS AND SPECIFICATIONS

FUNCTIONAL
– Authorized users should be able to add diplomas to the blockchain
– Any user should be able to verify if a document is in the blockchain
– The system should have a payment feature to monetize its use

NON-FUNCTIONAL
– The system should have an easy and minimalistic user interface
– The system should be accessible through a web interface
– The system should enable the user to upload a document
– Documents can be converted to hash values
– Only PDF diplomas can be uploaded to be checked or added
– The webpage should not take time to execute
– The system should enable the user to search for a document through a student identifier
– The system should be able to handle multiple requests at a time
CHAPTER 1: INTRODUCTION

1.1. Context

1.1.1. Short History
Nowadays, everyone has heard about cryptocurrency. However, this has not always been the case. The most notorious cryptocurrency is Bitcoin. Its first appearance was made through a paper called Bitcoin – A Peer to Peer Electronic Cash System by a certain Satoshi Nakamoto in 2008. One year later, bitcoin became publicly available, and people were able to start mining it. It took another year before the first transaction of bitcoin was made. At that time, nobody could assign a value to that virtual money. Ten thousand Bitcoins were exchanged for two pizzas. Shortly after, bitcoin started gaining notoriety. This was due to one big assumption; bitcoin was a decentralized currency. [1]

The idea quickly spread to the mainstream, and other cryptocurrencies emerged. As the years passed, the cryptocurrency market rose. As of today, bitcoin is worth 40,000 dollars, while another exciting competitor, Ethereum, is worth 3,000 dollars. Since cryptocurrency is decentralized, it is tough for countries to regulate it. According to the World Economic Forum, regulators worldwide have found it challenging to regulate blockchain and cryptocurrency, and no international coordination has been achieved yet since it is a borderless technology. [2]

Nevertheless, some countries decided to legalize or adopt cryptocurrencies as their national currency. One such example is Salvador which made Bitcoin one of their national currencies. However, the same cannot be said about Morocco. As of today, Morocco has legal vagueness regarding cryptocurrency. Hence, we do not know Morocco's future concerning cryptocurrencies. [3]

1.1.2. The Problem
In the context of higher degree studies, freshly graduated students receive a diploma to certify the completion of their degree. First, this document is of personal value since it demonstrates the individual's accomplishment over years of studies. However, it is not its primary purpose. Diplomas are used and reviewed by recruiting entities to certify the recruit background studies.
The issue is that it is easy to falsify these kinds of documents. Indeed, basic Photoshop knowledge or online tools enables individuals to modify or even create fake diplomas. Furthermore, recruiters can indeed contact the delivering institute for verification. However, this can take time and becomes quickly overwhelming when having to do so for tens or hundreds of applicants.

1.1.3. The Solution
Many solutions can be adapted to enable recruiters to verify whether the applicant’s diploma or their claim of background studies is legit. For instance, the awarding institute could develop a platform to enable recruiters to verify if the applicant has indeed completed a degree at their institute. Traditionally, such a platform would be available through a web interface, and the applicant's information would be stored in a database. However, using a database opens the door to security issues. A few years ago, blockchain technology emerged to solve these issues. Indeed, blockchain can be used to store data and is almost bulletproof. Instead of having the interface communicate with the database, it can communicate with a smart contract stored in the blockchain. The added value is that blockchain offers mechanisms that incapacitate hackers from manipulating data maliciously.

1.2. Background

1.2.1. Existing Work
Work and research have already been made to develop systems for document validity using the blockchain. For instance, an article named “Proposing a reliable method of securing and verifying the credentials of graduates through blockchain” has been published in 2021. In this article, we can see an implementation using the same technologies as those used in this project. Additionally, the authors decided to use three “necessary information” for data retrieval. Student identifier, marks statement, and time stamp. Unfortunately, the concrete implementation isn’t further detailed. [4]

1.2.2. Blockchain
According to IBM, a blockchain is an immutable collection of records that eases recording transactions and following assets in a network. Here, assets could be either tangible or not. In other words, a blockchain is accessible by anyone with whom it is shared and enables transparency of transactions. In order to achieve that, each transaction is recorded in a block of data. These blocks of data can record anything, whether tangible or not. For instance, one could record his identity or information relative to a shipment transaction. These blocks are connected in a chain, one after the other, to form the overall blockchain. These are securely linked, preventing some malicious party from adding another block between them. Finally, these transactions are irreversible, and each added block further strengthens the identity of the previous blocks and the overall blockchain.

As figure [4, Fig. 1] demonstrates, when a transaction is requested, a block that represents that transaction is created. Afterward, it is sent to all of the participants in that blockchain. These participants are considered miners. Miners can be anywhere around the world, and they compete to be the first to validate the transaction. In exchange, they receive rewards commonly referred to as gas fees for their proof of work. Afterward, that block is added to the blockchain, and the network is updated and redistributed. This process ensures the security and validity of transactions by ensuring that those respect the initial "contract" used. [5]
1.2.3. Ethereum

Ethereum is the technology used for this project. Ethereum is widely known to be a cryptocurrency. However, it does not limit itself to that. Forbes describes, "Ethereum is intended to be much more than simply a medium of exchange or a store of value. Instead, Ethereum calls itself a decentralized computing network built on the blockchain technology." Ethereum is based on blockchain technology. First, it uses cryptography to ensure the security of every transaction. Next, it has a system where people worldwide use computers to confirm transactions by solving complex mathematical problems. This is referred to as mining. Once the miner solves a problem and verifies a transaction, they are rewarded for their proof of work by receiving gas fees in Ether. Ether is the cryptocurrency associated with Ethereum. This Ether can then be used to make payments to acquire or sell goods. Forbes adds that "The Ethereum network can also be used to store data and run decentralized applications. Rather than hosting software on a server owned and operated by Google or Amazon, where the one company controls the data, people can host applications on the Ethereum blockchain. This gives users control over their data, and they have unrestricted use of the app as there is no central authority managing everything." [6]

1.2.4. Smart Contracts

In the Ethereum blockchain, transactions can be made through a smart contract. A smart contract is a computer program made up of functions and control statements. A smart contract can be seen as a "contract" where both parties agree to the terms and conditions. As IBM further details, “Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily. To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the “if/when...then…” rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.” Furthermore, IBM notes some benefits of using smart contracts. First, it is fast, efficient, and accurate. When the conditions are met, the contract executes automatically and instantly. Second, it is trustworthy and transparent. Since there are not any third parties, and it is on the blockchain, it is entirely transparent, and nobody can alter it for their benefit. Finally, it is almost entirely secure. Since blockchain transactions are encrypted, it is extremely tough to hack. Additionally, since it is considered a block, and each block is connected to a chain, a malicious
party would have to hack the entire chain for them to change only one record. For all of these reasons, smart contracts deliver many undeniable benefits. [7]

The counterpart is that gas fees and proof of work have to be paid to change data within a contract. The more the contract is long (code) and complex, the more it is costly to add to the blockchain. Additionally, once using its functionalities, the gas fees increase as the task becomes more complex. In other words, the more the transaction is complex, the more its price to execute increases. For example, the gas fee should be meager if one wants to modify a data setting to a specific value. However, the gas fees would increase if the contract has to make calculations such as converting the input value through complex algorithms.

1.2.5. Programming Languages

There are two main programming languages used for this project. Excluding HTML/CSS, this project uses JavaScript and Solidity. JavaScript is a functional programming language. Therefore it enables the construction of programs in an asynchronous manner. This means that long and complex functions can be executed, and instead of waiting for them to finish to move on to the following function, the program can continue and respond to other events during that time. This is extremely important in our case, where JavaScript is used to communicate with our smart contracts. Without this feature, the program would become unresponsive while waiting for the smart contract’s response. Instead, the contract promises a response, and the program continues running normally while waiting for it.

Solidity is the programming language used by Ethereum to create smart contracts. Solidity is an object-oriented programming language similar to Java. Solidity makes use of three different types of variables. State variables allow permanent data storage within the contract. Local variables are used to hold temporary data. Global variables reference relevant information about the blockchain and its properties. There are some important things to note. First, there is a particular type of variable called address. This is used to hold the address of a wallet or another contract. Secondly, functions can serve different roles, and keywords are used to define them. The payable keyword means that the function allows the transfer of Ether between two parties. Those can be between personal wallets and contracts, two contracts, or two wallets.
Next, the view keyword means that the function only reads data without modifying it. This is mandatory to view data from the front end. Additionally, reading data does not cost any gas fees compared to modifying data where a transaction has to be made. Hence a proof of work is required. The final important keyword is internal. This means that the function can only be accessed internally by other functions within the contract. Finally, Solidity allows the creation of modifiers. These are used to incorporate a certain logic and are then attached to the definitions of functions. Those functions can only be executed if the logic is met. For instance, one can create a modifier that checks that the owner is calling the function. If it is not the case, the called function does not execute.

1.2.6. Bootstrap
Bootstrap is one of the most popular frameworks used to design a webpage. It is open-source and enables the creation of responsive pages exceptionally easily.

1.2.7. Node
Node.js is an open-source, server-side environment written and used with javascript. Node has revolutionized javascript coding. This is because javascript was only used only for the client-side. When Node came along, it enabled programmers to use javascript to code backend servers. As geeks for geeks [8] explain, Node uses asynchronous and non-blocking I/O concepts. As they continue, "Non-blocking i/o means working with multiple requests without blocking the thread for a single request. I/O interacts with external systems such as files and databases. Node.js is not used for CPU-intensive work means for calculations, video processing because a single thread cannot handle the CPU works."
Additionally, Node has a big community creating handy libraries and frameworks, saving time.

1.2.8. Truffle
Truffle is a development environment used to test contract creation through the Ethereum Virtual Machine. Since deploying contracts to the Ethereum blockchain has a cost, it would be inconceivable to pay for each test. Therefore a virtual environment is essential. Truffle enables programmers to compile contracts and deploy them to a virtual blockchain. By deploying contracts to the virtual blockchain, developers can interact with them through various means. The simplest one is to use the built-in development console to test contract functions.

1.2.9. Ganache
Ganache is another tool that eases contract development and testing. It is a blockchain simulator that comes with a built-in user interface.
As the figure demonstrates, ganache simulates ten wallet addresses with 100 virtual ETH each. Ganache can then be used simultaneously with Truffle to create an environment for the whole process of contract development.

1.2.10. Metamask

Metamask is a crypto wallet software that can be accessed as a web plugin. It connects one's wallet to the web browser enabling easy interaction between dApps and the web client. Metamask enables connection with any Ethereum platform. Additionally, it can be set up to connect to a virtual blockchain running in a local environment such as ganache. It has a few key features.
As the figure demonstrates, users can see their current balance, record their transactions, and access their wallet addresses. Additionally, users can buy, send, or swap ETH. This is useful first to test and interact with smart contracts later on.

1.2.11. Web3

Web3 is the term used for technologies built to work with decentralized blockchains. It has gained exponential popularity over the last few years. Platforms developed using web3 technology are changing users' perspectives and use of the internet. As the New York Times [9] explains, “web3 platforms could give creators and users a way to monetize their activity and contributions so that today's mega-platforms do not.”

One famous example of a web3 application is pay-to-earn games. To achieve such applications, web3 technology can be used with javascript. For instance, to interact with Metamask, the website's frontend connects to the web plugin through web3 functions. In this case, the plugin is
considered a web3 provider. Web3 is still a vague word that could be used to reference various things. The only certainty is that it references technologies linked to blockchain technology.

1.3. Methodology

1.3.1. Approach

The approach undertaken for this project is a mix of research and testing out tutorials, continuously reiterating this process. The choice of using this approach is due to a few factors.

First, almost all of the technologies used had to be learned from scratch. Learning each one in a course-based approach would have taken too much time. For instance, a javascript course offered by Udemy [10] has 69 hours of video content without mentioning other course materials. Another javascript course offered by Codecademy [11] has an estimated 20 hours to complete. If we estimate an average of 10 hours per week of work put into learning Codeacademy's course, it would have taken two weeks to learn how to use javascript.

Second, blockchain technology is a concept that is hard to grasp. There are subtleties to this concept that are hard to catch without hard testing of the acquired knowledge. For instance, it is hard to understand how a contract works and is deployed without testing it out first. Hence, creating a plan or developing an overall project architecture from the start would be doomed to failure.

Third, blockchain technology and development are relatively recent. Tools and documentation are reasonably scarce and hard to grasp. For instance, Truffle's official website [12] has a section named guides. There are guides explaining how to use Truffle with various case studies on this page. However, the page offers only eight guides. Two were released in 2016, three in 2017, one in 2018, one in 2019, zero in 2020, and one in 2021.

Additionally, the website also has a tutorial section. This was extremely useful. However, it only explained the fundamentals from creating the project to the deployment, testing, and connection to Metamask. The tutorial's whole project consisted of creating a basic website that allowed the adoption of dogs. At the end of the tutorial, a note says, "stay tuned for our future tutorial," which has not been released yet.
1.3.2. Benefits
There are a few benefits to adopting this methodology. First of all, it has saved enormous time concerning the material learned. Indeed, by researching and using tutorials appropriate for the desired functionality to be adopted, the non-essential parts were omitted. For instance, forums recommended using bootstrap when looking for methods to create a web page's front end. Since bootstrap removes the CSS (styling) part of frontend development, it was futile to learn how to use CSS. Another example, when looking for ways to create a payment system, tutorials came up on how to do so with examples. To achieve that, we have to code it within the contract and in the frontend part, where web3 technology becomes handy. Learning every aspect of web3 technology would have been pointless. Instead, by using the various tutorials, we could understand and use only the parts that were needed to develop the payment system. As the last example, when learning how to write contracts with Solidity, it was more appropriate to research examples and then learn by reproducing them.

All of these examples serve to demonstrate that the adopted methodology was sound and reasonable to use.

1.3.3. Constraints
Even though the methodology had many benefits, it also had constraints. The most notable constraint was due to misinterpretation or misunderstanding, especially regarding how contracts work and interact with web3 technologies. For instance, one of the first misunderstandings was how Metamask communicates with deployed contracts. The mistake at the time was trying to code the Metamask interaction with the contract on the server-side. After about a week of research and successive failures, the solution became obvious. For Metamask to interact with the contract, it had to be coded on the frontend side. This seems obvious today. However, it was not apparent and caused a lot of time loss and frustration.

Another example, previously, it was mentioned that a payment feature was incorporated through research and tutorial testing. However, it was prone to the same mistake of misunderstanding. After trying to implement every single available tutorial, over 13 hours of work, without success,
it became apparent that some misinterpretation was made. Indeed, the mistake was from a misunderstanding of how the value of the payment had to be implemented. Instead of coding the value in the frontend, it was written directly in the contract. However, contracts are not allowed to specify how much money is sent. They can only specify how much money "must" be sent. This slight misinterpretation made a world of difference.

One last example, when trying to read data from one function within the contract to the other, it is as simple as writing a return value. However, if one wants to read data from the contract through the front, returning the value is not enough. Since reading data does not modify the contract, the view keyword must be added to the function definition. Without that, the contract believes that the function is modifying its state, and therefore, a transaction object is returned instead of the desired value.
CHAPTER 2: IMPLEMENTATION

2.1. Contract

2.1.1. Hashing Method
Hashing is a technique used to encrypt data. One of the ways to store a document on the blockchain is to encrypt it through hashing and save that value within the contract’s data.

![Graph showing SHA-256 Hashes vs. Collision Probability](image)

*Figure 5: SHA-256 Hashes vs. Collision Probability [13]*

The probability of having two hashed documents colliding cannot be excluded. Indeed, it is possible to have the same hashed value for two different documents. However, this is extremely unlikely. According to the figure [12, Fig. 5], there is a 0.1 probability to have a hash collision after $1 \times 10^{44}$ hashes using the SHA-256 method. This means that we would have a 10% probability of hash collision after adding $1 \times 10^{44}$ diplomas to our contract. Knowing that Al Akhawayn University has about 11000 alumni, currently enrolled students, dropped students, and professors combined. There is an infinitely small probability that two diplomas’ hashes collide. Therefore, it is a relatively safe method to explore.
The main incentive to use hashing over other methods is due to price constraints. Indeed, adding data to the blockchain costs money. As we already know, gas fees increase based on the complexity of the task to be performed. Therefore, adding a hashed value is a straightforward task to perform. Hence, the hashing method is the least costly approach to use.

Incorporating the hashing method within the contract has been achieved through three steps. First, creating a mapping, named diplomas, between string and integer values. By default, every string value is set to zero since it is empty. The second step was to create a function to modify the diplomas mapping. The function is pretty simple, taking one parameter, hash, and one statement that links the hashed value to the integer value one through the use of our mapping. The third step was to create another function that takes the same parameter, hash. However, instead of writing data, it returns the integer value linked to that hashed value through the use of our mapping. This enables us to quickly check if the hashed value has been added to our mapping. As explained previously, the first function is considered a transaction since it changes the state of our contract. Therefore, the one adding a diploma to the contract has to pay gas fees. However, the second function only reads data. Therefore, the one checking if a diploma exists within the contract does not pay any fees.

2.1.2. Key Association Method
The key association method is another way to check if a diploma has indeed been awarded. The idea is to link a unique identifier to relevant data such as degree, name, and graduation date. This method offers a few advantages compared to the hashing method. First of all, it removes the hashing collision issue since it uses a unique identifier. Second, it enables retrieving additional information not available within diploma documents, such as GPA. The drawback is the cost. The gas fees also increase since it writes more data in the contract than the hashing method.

Incorporating the key association method within the contract has been achieved through four steps. First, a diploma structure has been created. This structure holds the relevant data-name, CIN, GPA, degree, and graduation date. Second, like in the hashing method, a mapping has been created. However, instead of linking a hash value to an integer value, we link a student identifier, represented as an integer, to the diploma structure. Afterward, we have two functions. The first
function takes as parameters student identifier, name, CIN, GPA, degree, and graduation date. Then, it instantiates a diploma as defined by the previous structure. Finally, the function links the student identifier to the instantiated diploma through the use of the mapping. The second function takes one argument, student identifier, and returns the relevant information associated with that value.

2.1.3. Security Restrictions
We need security restrictions to ensure that nobody else other than the authorized entity is allowed to add diplomas to the contract. Contracts enable developers to add that logic within them. The idea is to allow only specific wallet addresses to use functions defined to add diplomas. To do so, we first have to create variables holding authorized wallet addresses. In the current implementation, two addresses are authorized. The first one, "contractOwner," is the one that is supposed to be used to add contracts by Akhawayn’s designated responsible. The second one, "contractAdmin," serves as a backup in the eventuality that the access to the contract owner's wallet has been lost. If both wallet accesses are lost, nobody will be able to add diplomas to the contract anymore. Next, we must create a modifier that checks if the wallet interacting with the contract is either the admin or the owner. Another function is also created to allow the passing of ownership. To restrict function access, we must add the modifier's name to those functions. In the current implementation, the restriction has been added to functions’ adding diplomas and modifying ownership.

2.1.4. Payment Functionality
Payment is the last functionality added to the contract. This functionality is different from paying gas fees. Indeed, gas fees are mandatory and go directly to miners. These are not coded by the contract developer. Instead, they are incorporated within the blockchain logic. Payment functionality, in this case, is used to make some functions usable only if a payment has been made. This can be used to require payment by third-party users who want to check if a diploma is indeed authentic to Al Akhawayn University. The current implementation requires a one-time payment of an arbitrary value, one eth, to use the function that checks if a diploma is valid. This is implemented in three steps. First, we have a mapping between wallet addresses and an integer value. Then, a function enables the payment, requiring one eth to be sent by the third-party’s
wallet address. This function first checks if that wallet address has already been paid. If that is the case, it does not further execute. If that is not the case, it checks that 1eth or more has been sent and then transfers that value to the contract owner's wallet address, which is Al Akhawayn's responsibility. Afterward, it links the third-party’s wallet address to the integer value one, which indicates that the payment has been made. Finally, to make functions useable only if a payment has been made, we add a required statement checking if the wallet address calling the function is in the mapping.

2.2. Backend

2.2.1. Node Server

The node server is used for the backend that communicates with the web interface. It serves a few roles. First, it links the bootstrap, jquery, and web3 modules to the frontend. Second, it is used to hash diplomas in pdf format and sends that value back to the frontend. Finally, it also creates various routes for the web pages.

To set up the node server, we first have to create the project directory. This can be achieved by using the command "npm init ." The terminal recognizes this command once node.js is installed. The project directory has a vital document named package.json that has a few roles, mainly creating basic scripts to launch our server and linking the modules used by our server. Then, we have to instantiate the express module. According to express.js [14] official website, “Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.” Once the module is instantiated, we have to set it up to listen to an arbitrary port. In our case, it is set to 8082. Finally, we use the command “npm start,” which calls the defined script in the package.json file to launch the server. The script launches the server in nodemon mode, meaning that whenever a change to the project is made, the server is automatically relaunched.

2.2.2. Routes

As mentioned previously, in the framework of Node, routes are used to link URL links to web pages. Additionally, they can be used to link URL links to function calls. These functions are
written within the node server file. To use them, we have to instantiate the router module. This module is then used to create paths and links them by sending, as a response, the desired page. For example, to link the URL “http://localhost:8082/dataAdd” to the desired page, we call the function router.get and give it the parameters “/dataAdd” and our handlers. Handlers are used to handle our request and response. Then, within the handler’s function, we call the method to send our file by joining the directory name to the directory of our desired file. The directory name is the first part of our URL up to the port. The directory of our desired file is the one present in our project’s directory. This process has to be repeated for each URL link. In this project’s implementation, routes are defined in a separate file and exported as a module to the main server file.

2.2.3. Uploading Document
To upload our document to the server, we use the multer module. This module is helpful because it is easy to use and serves as a middleware to automate the process of uploading files. To use multer, we have first to instantiate it and then create what is called disk storage. The disk storage indicates the destination, where the file has to be stored, and the filename we want to give it. In this implementation, the destination is within our source directory. The filename is the actual date in number format followed by the original filename. Keeping the diploma file on the server-side could become handy for future use cases. Afterward, the server has to be set up to listen when the frontend sends a file and respond accordingly by hashing that file. This can be done by listening to a specific URL request, similar to how routes are set up. In our case, when that URL request is made, we first check if the file object sent is empty. If that is the case, we simply redirect the user to the page they were in. If that is not the case, then we perform the hashing.

2.2.4. Hashing
Hashing is done using the crypto module. This module comes with multiple hashing methods. We are interested in the SHA-256 because it is one of the most complex and least likely to be prone to hashing collision. To do so, we first have to convert our file into a binary sequence using another module named fs. This module basically reads the file in binary form by giving it, as an input argument, the desired document’s path. This is also the reason why we had to upload the diploma file to the server. As a side note, we had to keep a record of the file path of this
document by saving it when the document was initially uploaded. Then we have to instantiate the SHA-256 algorithm and feed it with the binary version of our document file. Finally, we must check whether the user requested to check or add the diploma and redirect to the appropriate URL. This is further explained in the next section.

2.2.5. Communication with Client-Side
Communication with the client-side is essential to send our hashed value back to the front end. After exploring various methods, the most appropriate one was to attach the hashed value to the URL link. To do so, we can redirect the user to the page they are in, and at the end, a question mark followed by the hashed value. The question mark indicates that parameters are given and clearly indicates that anything following should not alter the destination file retrieval. In other words, it does not change the URL destination. If the user wants to check a diploma in the current implementation, they are redirected to the page they are in, followed by a question mark, the tag "checkDiploma," and finally, the hashed value. Otherwise, they are redirected to the same link. However, the tag indicates "addDiploma" instead.

2.3. Frontend

2.3.1. Overview
The frontend has two main pages and a navigation bar. The navigation bar includes Al Akhawayn’s logo and the navigation for the two pages, hash and data. Additionally, data has a dropdown menu to access two subsections. The first web page is for the hash method. It allows the user to upload a document. Additionally, it enables the user to pay and check or add a diploma to the blockchain. The second page has two subsections. The first subsection checks if a document is in the blockchain by entering a student identifier. The second subsection is used to add a diploma by entering the student’s identifier and relevant information. The idea was to have two different approaches. Either having all of the features on the same page, which was incorporated on the first web page—or dividing the adding and checking features into two different subpages, which were incorporated on the second web page. Additionally, through the front end, the Metamask plugin
interacts with the contract. Therefore, the first time the user reaches one of the web pages, Metamask launches and asks the user to log in.

2.3.2. Web Design

The design of the web pages is minimalistic. It is heavily dependent on bootstrap. The first page includes three buttons. The first button is to call the pay method. The second button is to add the document to the blockchain. The third button is used to check if the document is in the blockchain. When a diploma is successfully added to the blockchain, the webpage displays a popup saying that it has been successfully added. Similarly, a popup is displayed, either saying that the diploma is valid or not when a diploma is checked. Additionally, if the user has not clicked on the pay button and completed the transaction, the Metamask interface will display a red message indicating that the transaction will not succeed.

As said previously, the second page is divided into two subpages. The first one has a field to enter the user identifier and a button to check whether that identifier exists in the blockchain. Additionally, the webpage also has blank fields that the user cannot modify. These fields are used to display the name, CIN, GPA, degree, and graduation date. Next to them, there is the text indicating which is which. When the user enters a student identifier and clicks on the check button, the previous fields are either filled with the student’s information, if found, or a popup is displayed indicating that the diploma does not exist. The second subpage is to add the diploma by entering the student identifier and relevant information. It is similar to the first subpage. However, there are a few differences. The first one is that the button reads add instead of check. The second one is that the user can write in the fields. The last one is that the degree field is a dropdown button indicating a limited number of choices. All of the pages have as main title, "Diploma Verification."

2.3.3. Contract Interaction

The contract interaction is written in the frontend using javascript. Each page has an associated javascript file. These files all share some standard functionalities.

First of all, whenever a page is loaded, an app class is called. This app class is where all of the code is written. The app class has two main variables, web3Provider, and contracts. These are
used to hold the data of the web provider and our contract. Then the web provider is instantiated. The web provider could either be our Metamask or ganache. Therefore, we first check if the webpage has an injected Metamask provider. If that is the case, we request the frontend user to log in to their Metamask account. If the request is successful, we move on. Otherwise, we return an error saying that the user was denied access. We connect to the ganache provider if the webpage does not have a Metamask provider. Afterward, we instantiate the web3 provider. Finally, we move on to the following function.

The next function serves to initiate our contract. To achieve that, we have to get our contract's data. A build is created with a JSON file when a contract is deployed. This file contains the contract's relevant data. Therefore, we have to call that file and pass the data to the truffle contract module. This module facilitates the linkage between our frontend and our deployed contract. Once the link is instantiated, we have to give the provider we are using as a parameter. In our case, we give the parameter of the previously instantiated web3 provider, Metamask.

Then, we have to bind events. This can be done in multiple ways. In our case, we read the URL and check for relevant keywords. For instance, if the URL has a question mark followed by the keyword "checkData," then the function to check data is called.

Afterward, functions differ with regard to the task to be accomplished. However, they all share the same first and last two procedures. The first one is to get the wallet's account. The second one is to get an instance of the contract. The first part is needed to communicate to the contract which account is calling it. The second part is essential for the frontend to call the contract's functions. The last two procedures are needed to process what happens after the contract's function call. The first one is executed if the function call is successful, and the second one is executed if the function call fails.

From this point, each javascript file linked to the webpages differs in functionalities. The javascript file linked to the hashing page can call three different contract functions. It first determines which function to use by checking the URL parameters. The three possible parameters are check, add or pay. In the case of check or add, the server-side attached a hash
value after those keywords. Therefore, the function splits the URL to store the hash value. Then, the appropriate function is called. The two first functions, add or check, take in two parameters. The first one is the hash value. The second one is the account that corresponds to the user’s wallet. The third possible function is the payment method. It takes in two parameters as well. The first one is the wallet address, and the second one is how much is to be paid. Afterward, based on which function has been previously called, an alert is sent to the front page with the appropriate message. This is displayed as a popup.

The two other javascript files linked to the pages responsible for handling the key association method work in the same way, with the difference in the contract function’s call. The page responsible for checking the student identifier calls the contract's function associated with it. This function takes two arguments: the student identifier and the wallet address. The web page's fields are updated with the corresponding information if the identifier exists. Otherwise, a popup is displayed, alerting us that the diploma does not exist. The page responsible for adding the diploma calls the function to add a diploma with the relevant data. It takes in as an argument the student identifier, name, CIN, GPA, degree, graduation date, and wallet address. Finally, there are two essential things to note. First, Solidity does not handle floating numbers. Therefore, the GPA has to be converted to an integer value by being multiplied by one hundred when added to the chain and divided by one hundred when retrieved. Secondly, as previously mentioned, when a function with a restriction is called, Metamask displays a red message saying that the transaction will not succeed if the logic is not met.
CHAPTER 3: STEEPLE ANALYSIS

3.1. Social
The social implication of this project will hopefully limit people's tendency to lie about their background information. Having a diploma and transcript verification system will add another level of difficulty for people trying to forge this information.

3.2. Technological
All of the frameworks used in this project are recent and up to date with the latest trends. All of these technologies were previously explained.

3.3. Environmental
Environmentally, this project could limit the use of printable paper by digitalizing files. Nevertheless, the downside of using blockchain is that it consumes a lot of energy to mine and use, polluting the world.

3.4. Economical
This project's scope can reach two different clients, either universities such as Al Akhawayn or, more broadly, any entity that needs an effective way to check the validity of documents.

3.5. Political
Blockchain has caused many controversies this last decade. Countries have banned or adopted cryptocurrency. Nevertheless, in this project's scope, no political implications should affect it.

3.6. Legal
Cryptocurrency has legal vagueness in Morocco. Additionally, Morocco has banned cryptocurrency use since 2017. However, as more and more countries are moving towards the legalization and regulation of cryptocurrency, there is hope for Morocco to follow the trend.

3.7. Ethical
This project will remove people’s ability to forge documents. Its goal is to guarantee document validity and authenticity. In the specific case of Al Akhawayn, this project will enable anyone to check whether the said diploma is valid. In a broader context, this project could be applied to all Moroccan sectors to help fight fraud and ease document relate procedures.
CHAPTER 4: CONCLUSION

In conclusion, the primary purpose of this project was to use blockchain technology to verify diploma and transcript authenticity. The reason behind the use of blockchain technology is that it is a highly secure and new way to store and use data.

There were three core aspects to the implementation of the project. The first was to create a smart contract that held the logic and rules for diploma manipulation. Once smart contracts are deployed to the blockchain, their logic and rules become immutable. Therefore, with exemplary implementation and architecture, hacking is almost impossible. The developed smart contract had four big implementations. First, it had the functionality to restrict access to specific methods by allowing only authorized wallet addresses to call and use them. Second, it had a payment method that allowed the monetizing of document verification. Third, it incorporated a hashing method to add documents to the blockchain and verify their existence. Finally, it had a key association method to add and retrieve diploma-relevant data by using a student identifier as a parameter.

The second aspect was the backend development. This was achieved by using JavaScript and Node. The primary purpose of the backend was to hash documents and send that information back to the frontend. Additionally, the backend was responsible for the routing of webpages. Furthermore, even though it was not implemented, the backend could be used in the future to create a login interface by communicating with a database. This could, for example, create a more sophisticated user experience.

Finally, the third aspect was the frontend or user interface. This was developed by heavily using JavaScript and some bootstrap, and HTML. JavaScript was mainly in charge of creating the communication between the web interface and the smart contract deployed to the blockchain. This had to be done through the frontend to use the Metamask web plugin.
References


This capstone aims to use blockchain technology to fight fraud related to official documents. More specifically, create a blockchain-based system for Al Akhawayn to deliver transcripts and diplomas.

The first part of the analysis will understand how Al Akhawayn manages the creation and delivery of transcripts and diplomas. Then, research and find solutions to incorporate blockchain into those. Afterward, investigate the optimal solution of delivery. For instance, would it be better to deliver those documents electronically or add a QR code to the physical documents to check their validity? Finally, investigate solutions on how to check those. There are many ways to check achieve that. For instance, would it be viable to create an app that reads QR codes and returns whether the document is valid or not? Another idea could be to create a website where one can upload the document to check its validity.

Once the analysis is done, having all the necessary information, the design can begin. The overall idea is to create a system that will manage the authenticity of diplomas and transcripts. Therefore, the design will focus on how to achieve that the most efficiently possible. As of now, the idea is to incorporate blockchain with electronic versions of those documents. However, this is subject to change. The design will incorporate agile principles and methodology, more specifically SCRUM. At the end of the design phase, all user stories will be created and organized in a series of weekly sprints. This will ensure that the project is advancing effectively and regularly.
Since this project will use agile methodology to produce a minimum viable product (MVP), the implementation and testing will go hand in hand. As of now, an MVP is the most realistic final deliverable. Nevertheless, this is subject to change according to the analysis and design phase. Once a sprint is over, all new features will be tested to ensure that everything is working correctly.

This project aims to produce a system that will guarantee document validity and authenticity. This will help fight fraud concerning Al Akhawayn's diplomas and transcript in the short term. Nevertheless, in the long term, this could be applied within all sectors of Morocco to fight fraud and ease document-related procedures.