A Blockchain-Based Document Verification System for Employers

A PROJECT REPORT

for

INFORMATION SECURITY MANAGEMENT

(CSE3502)

F1 Slot

in

B. Tech (Information Technology)

by

KSHITIJ DHYANI

(18BIT0131)

SUBHRA PALADHI

(18BIT0191)

JAHNAVI MISHRA

(18BIT0243)

Winter semester, 2021

Under the Guidance of

Prof. SUMAIYA THASEEN I

Associate Professor Grade 1, SITE

Abstract

Academic Institutions often maintain records and documents of the students enrolled within the institute. These records are greatly regarded by employers and often verified before giving employment offers. While all the other sectors in academia are moving towards automation, this specific use case is still quite primitive in its functioning. Even when this is a regular activity undergone by academic institutes, methodologies with vulnerabilities and reasonable overhead are employed. There is a need for a tamper-proof, reliable, and faster mode of document verification for employers which can be used on the go with complete reassurance for the authenticity of the provided documents. This paper puts forward a blockchain-based platform that allows academic institutes to offer a secure, dependable, cost-effective, and scalable verification of documents via a web interface. The platform is built upon the logic written in solidity and utilizes the Ethereum blockchain to enforce immutability of the document, the backend is linked to the user interface using the web3.js library. Additionally, the platform offers a statistical comparison of these records to other students and assists the employer to assess the performance of the student being recruited.

Architecture

★ **High Level Design:**

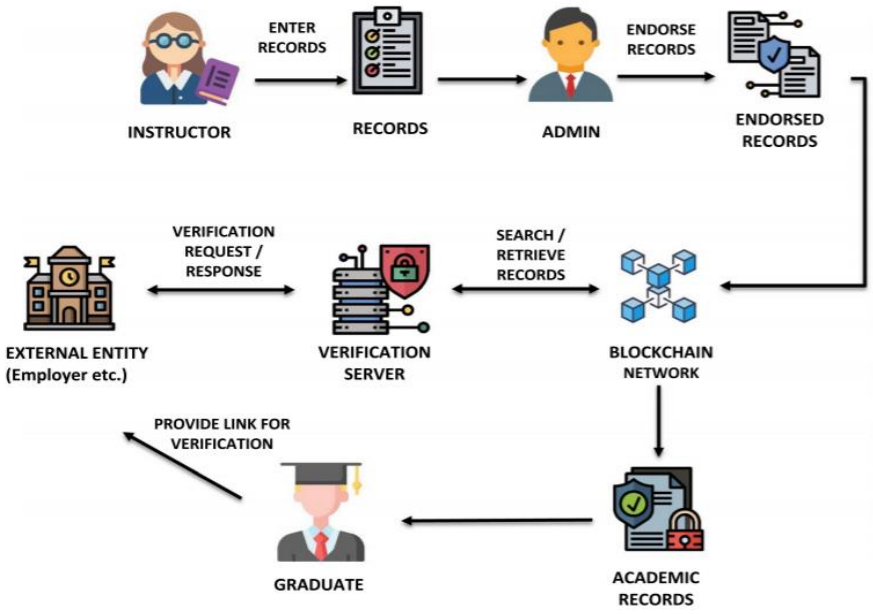

Figure 1: Flow of the proposed methodology

★ **Low Level Implementation**

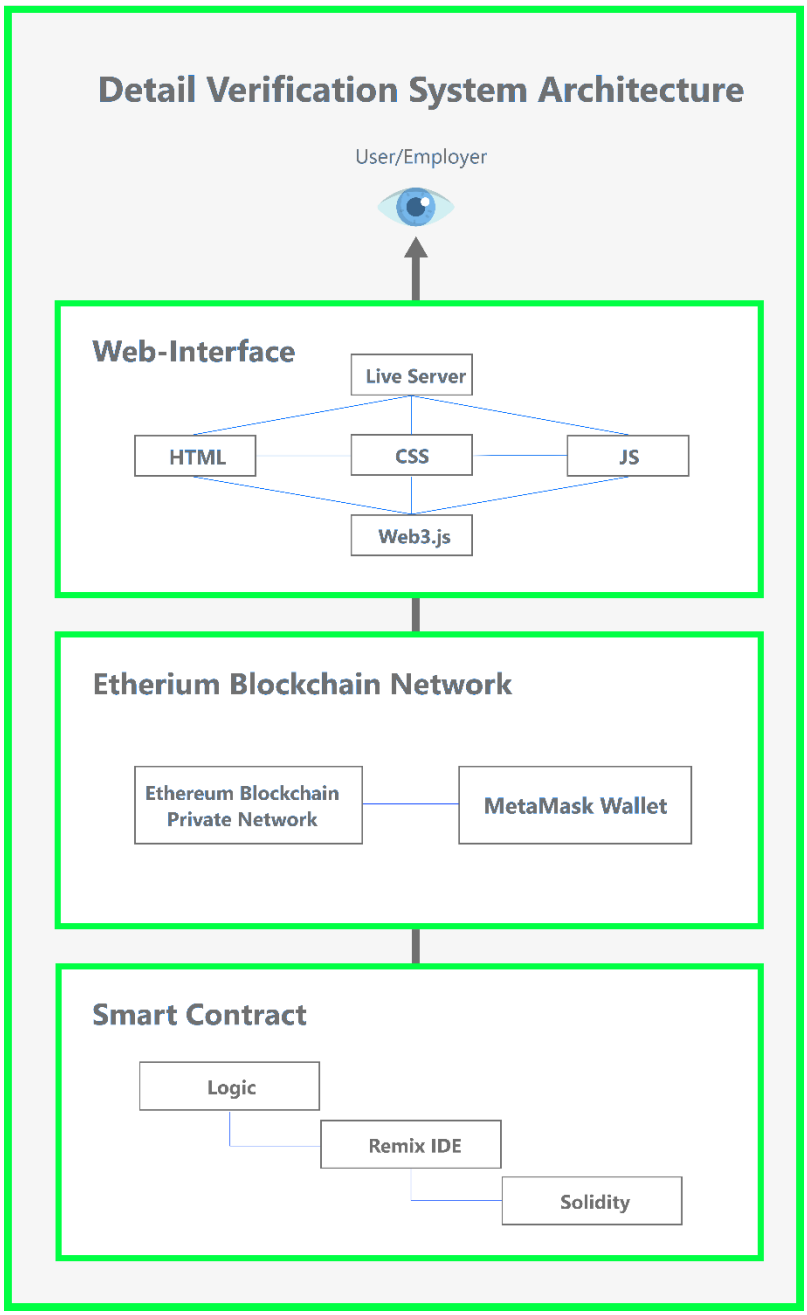

Figure 2: System architecture for the proposed technology.

The system architecture can be broken down into three main modules. Namely the (1), smart contract, the actual (2) ethereum private block-chain network, and finally the (3) web interface module.

Ethereum Smart Contracts are a methodology developed and provided by the ethereum blockchain community, which allows any blockchain idea to become highly scalable. This is achieved by introducing a programming language called solidity [19], where the logic to be followed by the blockchain is specified. With this algorithmic set of rules written in solidity the associated ethereum blockchain network knows exactly how to function. For our proposed technology, the logic for the document verification system is written in solidity, where the struct data type defines the attributes to be stored for the student, all within the Remix IDE.

Ethereum blockchain network is the second layer of the proposed architecture and it is essentially a network of nodes/participating devices working together to validate transactions and maintain a ledger based on the norms of the ethereum community. This blockchain network is where the data of the students will be stored once the smart contract is deployed. Once the contract has been deployed and the data has been pushed and validated by the network, it fundamentally becomes tamper-proof. The deployment is done using the Remix IDE.

The last and the topmost layer of the proposed architecture is the web interface, where the employer or academic authorizes can come to verify information and documents. This web interface is connected to the blockchain using a Javascript framework- Web3.js. This web interface abstracts and encapsulates the other two layers, and makes them invisible to the user. The user simply searches the desired candidate's information and verifies it.

Modules implemented

The proposed technology was implemented through three modules.

1. Smart Contract Module

The smart contract is written in the solidity programming language, which is considered to be a mild representation of C++ and Javascript. The smart contract is essentially the logic behind the functioning of the blockchain. For our particular use case, our smart contracts implement the logic behind document and detail verification. The solidity program a..k.a the smart contract defines functions within the code to perform functions such as creating a student, and reading student details with functions *createStudent()* and *readStudent()* respectively. The details within each student's block are determined by defining a *struct*, which is essentially a user-defined data type that helps represent complex real-world entities in the code easily. This contract is written within the Remix integrated development environment which facilitates

solidity development. The smart contract for our proposed application is written in the solidity version *0.5.0* and is further augmented with *abicoder v2* which allows us to use structs and dynamic variables along with functions. The code for the smart contract can be found and downloaded at https://github.com/wimpywarlord/ISM-PROJECT.git

2. Ethereum Blockchain Network

The ethereum blockchain network used for the proposed technology is *Rinkby* private network [20]. This network is a typical ethereum network that utilizes the Proof of Work (PoW) algorithm to verify transactions to and hence limits the user from tampering with the data. In order to deploy the smart contract to the rinkby network, the *metamask* cryptocurrency wallet is used to pay the gas money [21] as shown in figure 2.

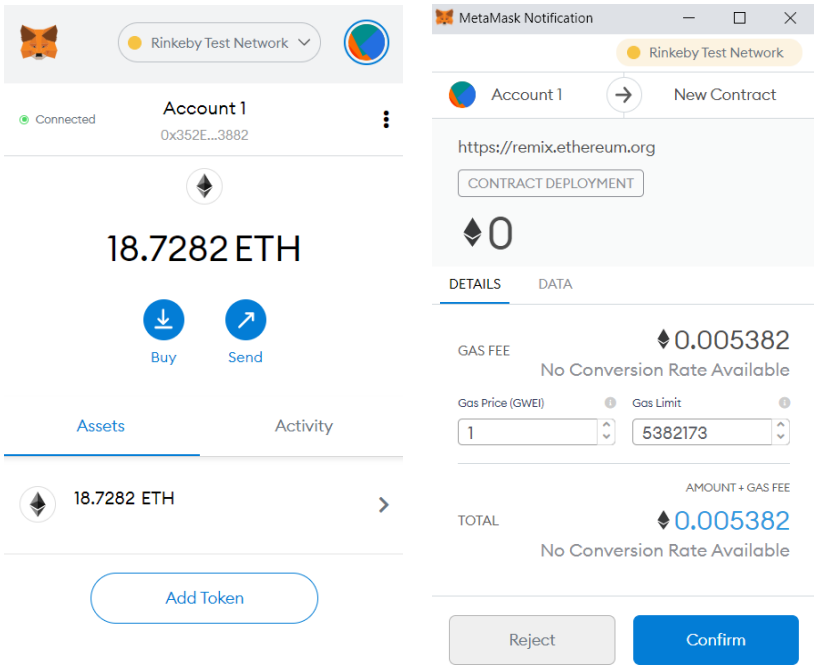

Figure 3: Metamask cryptocurrency wallet being used for payment of gas fee while the deployment of smart contracts on the Ethereum Rinkby test network.

To attain ether in the rinkby network wallet, public faucets were used. Upon linking the metamask wallet with the Remix IDE, the contract is deployed on the network. With this, the blockchain is set up and additionally equipped with the smart contract which is the logic of functioning.

3. Web Interface

The web interface is the topmost layer of the whole architecture. This is the only layer that is visible to the user, other layers are of no significance for the user. The web interface needs to be connected with the blockchain. This is done using the *Javascript* framework *Web3.js* [22], which is developed to enable interactions between ethereum networks and nodes over *HTTP/IPC/WebSocket* protocols. Additionally, the *Application Binary Interface* (ABI) for the smart contract [23], along with the deployed contract *to address* is fed to the web3.js function, to perfectly establish the connection. Once connected the interface is structured using HTML, styled using CSS, and manipulated using Javascript.

➔ Scalability

The platform is built upon the ethereum blockchain and hence scaling is a very straightforward process. As per the current standard the chain can store up to 13TB of data. Additionally, if the data requirements increase, a simple enhancement of memory of the full nodes will vastly increase the ledger capacity. Moreover, the simple web interface for the users, allows them to use the service without being connected to the ethereum blockchain directly. Therefore, even in the state of storage issues, the platform is fully functional.

➔ Challenges and Solutions

Currently, the ethereum blockchain can support only up to 30 transactions per second which acts as a bottleneck in some edge cases where rapid update queries in data are issued for multiple participating organizations at the same time. However, the *Ethereum 2.0* is expected to be released by the end of this year, which will push the threshold to around 100000 transactions per second.

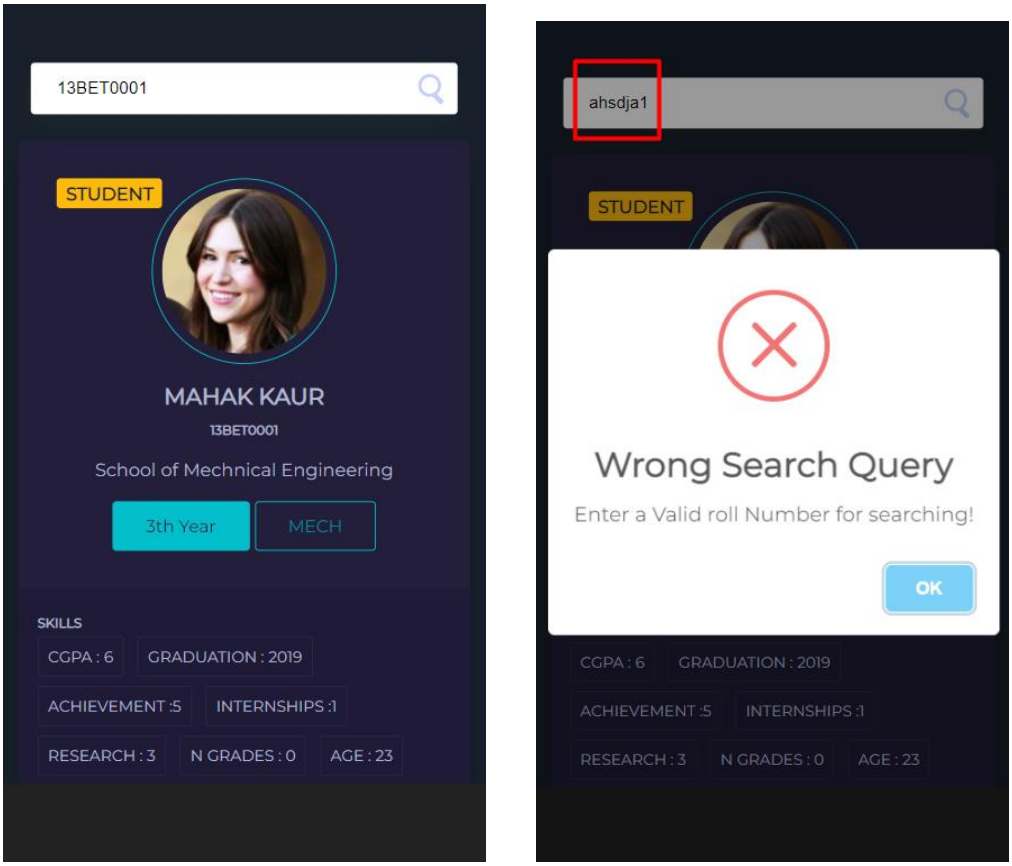

Figure 4: The web interface displaying the student's information and detecting faulty queries.

The web interface gives a thorough overview of a student's profile by giving information about the academic record, extracurricular record, industrial experience, research experience, and much more as shown in figure 4.

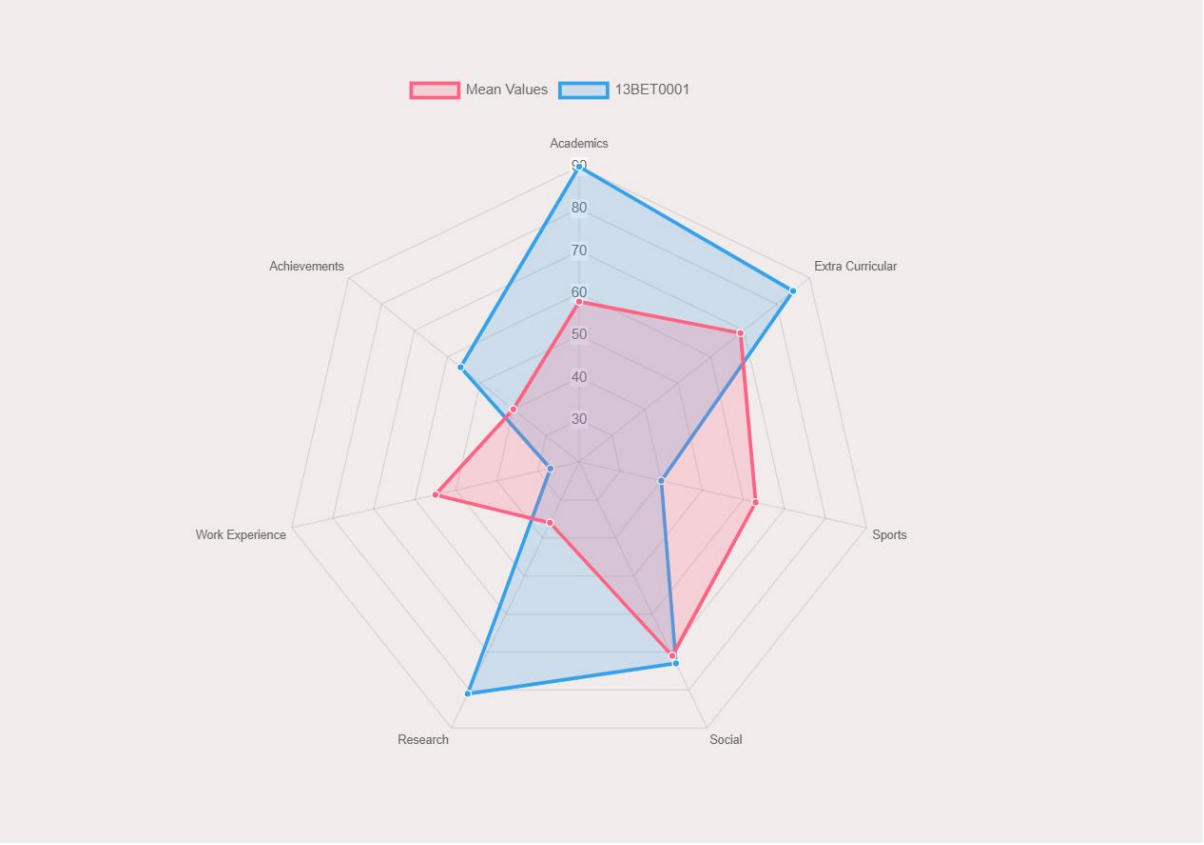

Figure 5: Analytical overview of the student's performance.

Additionally, the web interface offers, searching candidates based on their skill, which helps employers to shortlist desirable students for their job role and moreover provides an analytical overview of the student's abilities while comparing it to the average of all the students in the university/organization as shown in figure 5.

Implemented Code link

https://drive.google.com/drive/folders/1Qj57rdxIbQNzi3HdZHhzOPLlA0HJvdJ3?usp=sharing

Plans for review 3

1. **Storing the pdf (more than one):**

   We need to add the feature of storing multiple PDFs and also the transcript or the grade history can be uploaded in a PDF

2. **Having a unique identifier for student with multiple qualification from same college, (solve by storing a unique identifier like aadhar)**

   This basically means for example, if the student has done both UG and PG from the same university, then there should be an identifier for the student to identify that the same student is mentioned in two different fields. So aadhar number is unique for every individual and it can be used as an identifier.

3. **Find students with specific skills ()**

   This will help the recruiter to filter the students according to their specific requirements. For example, if they need a student who is good in Java specifically or may be a student who is good in web development. So like this the students with required skills could be found.

4. **Better user interface.**

   A better interface for user interaction by adding new features.

5. **Optimised Skill filtering**

   This is for optimising the skill filtering techniques that is to find the students with specific skills as mentioned above.

6. **Improve graphs and charts**

   The graph used is the spider chart which is not very easily understandable by someone who is naive. So we will try to use pie chart, bar chart or any other chart which is easily understandable to depict the above data.

References

[1] Garain, U., & Halder, B. (2008, December). On automatic authenticity verification of printed security documents. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing* (pp. 706-713). IEEE.

[2] Brdesee, H. S. (2019). An Online Verification System of Students and Graduates Documents and Certificates: A Developed Strategy That Prevents Fraud Qualifications. *International Journal of Smart Education and Urban Society (IJSEUS), 10*(2), 1-18.

[3] Lakmal, C., Dangalla, S., Herath, C., Wickramarathna, C., Dias, G., & Fernando, S. (2017, September). IDStack—The common protocol for document verification built on digital signatures. In *2017 National Information Technology Conference (NITC)* (pp. 96-99). IEEE.

[4] Putro, P. A. W., & Luthfi, M. (2019, October). An authentic and secure printed document from forgery attack by combining perceptual hash and optical character recognition. In *2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)* (pp. 157-162). IEEE.

[5] MIT Media Lab Learning Initiative and Learning Machine, "Digital Certificates Project," [Online]. Available: http://certificates.media.mit.edu/

[6] "ShoCard | Identity for a Mobile World", *Shocard.com*, 2017, [online] Available: https://shocard.com/.

[7] Turkanović, M., Hölbl, M., Košič, K., Heričko, M., & Kamišalić, A. (2018). EduCTX: A blockchain-based higher education credit platform. *IEEE access*, *6*, 5112-5127.

[8] Badr, A., Rafferty, L., Mahmoud, Q. H., Elgazzar, K., & Hung, P. C. (2019, June). A permissioned blockchain-based system for verification of academic records. In *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)* (pp. 1-5). IEEE.

[9] Nakamoto, S. (2019). *Bitcoin: A peer-to-peer electronic cash system*. Manubot.

[10] Chowdhury, M. J. M., Colman, A., Kabir, M. A., Han, J., & Sarda, P. (2018, August). Blockchain versus database: a critical analysis. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (pp. 1348-1353). IEEE.

[11] Buterin, V. (2017). Ethereum: a next generation smart contract and decentralized application platform (2013). *URL {http://ethereum. org/ethereum. html}*.

[12] Metcalfe, W. (2020). Ethereum, Smart Contracts, DApps. In *Blockchain and Crypt Currency* (pp. 77-93). Springer, Singapore.

[13] Wang, S., Ouyang, L., Yuan, Y., Ni, X., Han, X., & Wang, F. Y. (2019). Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(11), 2266-2277.

[14] ConsenSys. (n.d.). *Metamask Wallet*. MetaMask Docs. https://docs.metamask.io/guide/.

[15] *Rinkeby Ethereum Test Network*. AirSwap Support. (n.d.). https://support.airswap.io/en/articles/2831385-what-is-rinkeby.

[16] Remix. (n.d.). *Remix - Ethereum IDE*. Remix. https://remix-ide.readthedocs.io/en/latest/.

[17] Wood, G. (n.d.). Solidity. https://docs.soliditylang.org/en/v0.8.3/.

[18] *web3.js - Ethereum JavaScript API*. web3.js - Ethereum JavaScript API - web3.js 1.0.0 documentation. (n.d.). https://web3js.readthedocs.io/en/v1.3.4/.

[19] Dannen, C. (2017). *Introducing Ethereum and solidity* (Vol. 318). Berkeley: Apress.

[20] Iyer, K., & Dannen, C. (2018). The ethereum development environment. In *Building games with ethereum smart contracts* (pp. 19-36). Apress, Berkeley, CA.

[21] Lee, W. M. (2019). Using the metamask chrome extension. In *Beginning Ethereum Smart Contracts Programming* (pp. 93-126). Apress, Berkeley, CA.

[22] Lee, W. M. (2019). Using the web3. js APIs. In *Beginning Ethereum Smart Contracts Programming* (pp. 169-198). Apress, Berkeley, CA.

[23] Zheng, G., Gao, L., Huang, L., & Guan, J. (2021). Application Binary Interface (ABI). In *Ethereum Smart Contract Development in Solidity* (pp. 139-158). Springer, Singapore.