ICIC Express Letters
Part B: Applications
Volume **14**, Number **11**, November **2023**

ICIC International ©2023 ISSN 2185-2766
pp. **1217–1224**

DEVELOPMENT OF WEB 3.0 INFORMATION SYSTEM WITH PERMISSIONED BLOCKCHAIN

Justin Susanto and Arya Wicaksana*

Department of Informatics
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang 15810, Indonesia
justin.susanto@student.umn.ac.id; *Corresponding author: arya.wicaksana@umn.ac.id

Received January 2023; accepted April 2023

Abstract. *Blockchain has the potential to securely store and manage information transparently in a distributed manner. It allows the development of decentralized applications in areas dependent on client/server technology, such as information systems. This study explores the design and implementation of a decentralized information system for real-world use cases. The EOSIO blockchain is chosen for this study's proof-of-concept, and a microservice is used to connect it with the rest of the information system. This study's chosen application of the information system is university student activity records and testing and evaluation using actual data from Universitas Multimedia Nusantara. The throughput of the decentralized information system delivers 7,492 requests per minute on average with a single validator node. This study showcases blockchain technology's design and implementation in decentralizing information systems with real use cases and data.*
Keywords: Blockchain, Downtime, EOSIO, Information system, Microservice, Throughput

1. **Introduction.** The discovery of blockchain and distributed ledger technology (DLT) allowed the development of decentralized applications that are transparent, immutable, and secure in comparison with the client/server technology [1-6]. It unveils the possibilities for developing a decentralized information system with the same or even higher quality of service in terms of throughput and uptime compared to the client-server architecture. Moreover, distributed applications do not suffer client/server technological drawbacks: single point of failure. Decentralized applications also bring back control to the users in the following three aspects: privacy, ownership, and dissemination [7]. Related works on decentralized applications demonstrate the potential and benefit of blockchain technology [8-11].

Guo and Liang in [8] pointed out that blockchain can establish a credit mechanism where parties lack mutual trust, reducing the non-technical costs required by centralization. Regarding information systems, bank credit information systems with blockchain technology could establish data ownership and promote data sharing. Sytnyk et al. in [9] designed and prototyped a decentralized information system for supply chain management using the Ethereum blockchain with Solidity smart contracts. It is concluded that the transparency of the tracking, trustworthiness and automation process in the supply chain has been improved. Beck et al. in [10] studied the implications of blockchain technology in business and information systems and encouraged information system researchers to start new research on the topic. The technology could facilitate cross-border money transfers and complex financial transactions, including tracking ownership of different and real-world assets in international shipping.

DOI: 10.24507/icicelb.14.11.1217

Blockchain technology adoption also benefits information systems in areas other than finance and business, i.e., academics and universities, as in [11]. Students' information is critical and sensitive, and blockchain technology provides a more protected and trusted archive of records. Ali et al. in [11] introduced three different blockchain models that can match different sizes of organizations and eliminate the need for miners. This is achieved by using trusted and well-identified nodes by the university to deal with the student information system. However, there are no produced results on the performance of the proposed decentralized information system. Performance is crucial in adopting new technology, such as blockchain, for established fields like information systems.

This study extends current state-of-the-art studies in decentralized information systems based on blockchain technologies by applying real-world use cases and scenarios. The main contribution of this study is the design and implementation of a blockchain-based information system for private-held institutions. The application chosen in this study for performance testing and evaluation is the data management of student activity credit units (SKKM) at Universitas Multimedia Nusantara (UMN). UMN is a private university in Indonesia with more than 10,000 active students in 2022. The use cases and data for testing and evaluation are obtained from UMN to develop the system. Private blockchain and semi-decentralized architecture are adopted for the decentralized information system. EOSIO blockchain technology is preferred to be cost-free and software developer-friendly [12-15].

The rest of this paper is organized as follows. Section 2 describes the research methods, and Section 3 presents the experimental results and discussion. Finally, Section 4 concludes this paper with suggestions for future work.

2. **Methods.**

This section describes the research methods used in this study: design, implementation, testing and evaluation.

2.1. **Design and implementation.**

The design and implementation part consists of front-end (website) and back-end (blockchain and database) with microservice as the connecting bridge. The microservice is tested under several scenarios to validate it in real-world environment.

2.1.1. *Blockchain subsystem.*

The main idea is to represent the credit units as EOSIO tokens (fungible tokens) in the blockchain. Transactions are created and pushed into the block that is later added to the blockchain. These transactions are made of tokens that keep the credit units given to the students. Transactions on the EOSIO service system, including token transfers, are added with a memo. In this case, the memo helps provide additional information on the transaction for the credit unit type. The design will allow the generation of ten billion tokens, and all tokens will be kept in the administrator account.

Each user account created on the website has its designated EOSIO wallet. The wallet design is a standardized design using keosd. Upon the events' validation, students who request credit units through the website will receive a token in their wallet. When the EOSIO wallet creation is successful, user data will be entered into the database. The user will receive an email for account verification using Google's SMTP (Simple Mail Transfer Protocol).

Each token transaction on the NodeOS system will generate an id of the transaction. The id is valid for tracing the transaction and is stored in the database. This means that every token transaction between users is carried out. The transaction will be held in a database containing transaction history data. Transaction history data cannot be manipulated simply because transaction history data can be compared with data on the EOSIO blockchain.

2.1.2. *Interface subsystem.* Students must register to access the website using the university's official email domain. The registration process takes users' data such as email, password, username, and id to create an EOSIO blockchain account. The website home page provides the user with all necessary information, including current credit units and transaction history. The website facilitates the business process needed to submit and validate a student's activity. Credit units are given upon successful validation in an EOSIO token. The student service department accesses the website with various roles from the students, and the staff can review and validate submitted activities.

All the business processes required are designed together with the blockchain. Smart contracts are used in the blockchain to provide the business process's functionalities on the website. The blockchain network is integrated into the website by using microservice. The microservice backend design will follow the web service model built using the spring boot framework. The controller will have a dependency on the service. The service can depend on other inbound/outbound services or the service repository to retrieve data from the database.

2.1.3. *Database subsystem.* The database system used is NoSQL-based (or non-relational database) MongoDB. Unlike relational databases, NoSQL databases are collection-based, containing documents in JSON or Javascript Object Notation. NoSQL database is chosen due to its high adaptability to dynamic data changes, which is suitable for the target application. Besides, the web service system is designed to be implemented with systems currently running because the system has no known structure, so the NoSQL database is suitable for the conditions of this system requirement. The details of all the collections are shown in Table 1.

Table 1. Database design with NoSQL

User	SKKMQueue	SKKMTransactionHistory
id	id	id
userEmail	activity	transactionId
eosUsername	photos	skkmPoints
password	skkmPoints	email
NIM	email	activity
token	NIM	eosUsername
isVerified	createdDate	createdDate
createdDate	version	version
version	isAccepted	

User collection has documents consisting of fields id (_id), user email (userEmail), EOSIO username (eosUsername), password, student's identity number (NIM), token, isVerified, createdDate, and version. The token is a field that functions for user verification. Every time a user is registered, a verification email is sent, and the user is given a token for verification. The isVerified field indicates whether or not the user has verified the account. All collections share the database's id, createdDate, and version.

The SKKMQueue collection has documents consisting of fields id (_id), activity, photos, skkmPoints, email, student's identity number (NIM), createdDate, version, and isAccepted. The activity field is used to store activities carried out by students to claim credit units. The photo field stores photos as evidence of the student activity, and the field uses Base64 encoding. The isAccepted field is used to determine submissions' status based on the admin's decision to accept or reject the submission.

The SKKMTransactionHistory collection has fields consisting of id (_id), transactionId, student activity credit units (skkmPoints), email, activity, EOSIO username (eosUsername), createdDate, and version. The transactionId is used to store the transaction id from the EOSIO blockchain.

2.2. **Testing and evaluation.** The throughput parameter used as a benchmark for the microservice capability used in testing in this study is the microservice throughput, namely RPM (Requests per Minute). This test uses Apache JMeter to test the microservice throughput. The tests are carried out using two computers. The first computer acts as a server that runs the spring boot server instance, ExpressJS, keosd, NodeOS, and EOSIO blockchain service instance. The other computer is used as a client, and it runs the Apache JMeter.

There are ten out of twelve endpoints in total that are testable. The Register endpoint could not be tested because sending an email using the Simple Mail Transfer Protocol (SMTP) server is free. Thus, the testing procedure would cause the SMTP to get blocked instead. The TokenVerification endpoint cannot be tested because it is part of account verification, which will return an error upon verification if it is already verified before.

The tests are carried out using a one-second timeout parameter except for the endpoints with uploading features; the timeout is ten seconds. The number of users per minute that accesses the endpoint simultaneously is 1,000, 5,000, 10,000, and 15,000. The tested endpoints are Login, GetAccountBalance, ListSKKMStatus, SKKMTransactionHistory, EditUserSKKM, RequestSKKM, DetailSKKM, SendSKKM, GetSKKMQueue, and AcceptOrRejectSKKMRequest.

3. **Results and Analysis.** In this section, the test results for all tested endpoints are presented. The test results of endpoints Login, GetAccountBalance, ListSKKMStatus, and SKKMTransactionHistory are shown in Tables 2-5.

Table 2. Endpoint Login

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	5,000	0%
3	10,000	1s	10,000	0%
4	15,000	1s	10,741	28.39%

Table 3. Endpoint GetAccountBalance

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	5,000	0%
3	10,000	1s	10,000	0%
4	15,000	1s	12,155	18.97%

Table 4. Endpoint ListSKKMStatus

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	5,000	0%
3	10,000	1s	10,000	0%
4	15,000	1s	14,011	6.59%

Table 5. Endpoint SKKMTransactionHistory

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	5,000	0%
3	10,000	1s	8,588	14.12%
4	15,000	1s	8,333	44.45%

The results shown in Tables 2-5 are from the four of the most accessed endpoint by the users. This demonstrates the system's reliability in serving users on the four endpoints with 0% downtime for 5,000 users per minute within 1s timeout. Downtime for larger users per minute would be reduced given the larger timeout parameter.

The EditUserSKKM endpoint is tested using a ten-second timeout parameter. The results are shown in Table 6. The resulting 1,000-user throughput is 960 requests per minute with 4% downtime. Testing with 5,000, 10,000, and 15,000 users cannot be tested because the microservice experienced a downtime when tested with 5,000, 10,000, and 15,000 users. This endpoint has a reasonably low throughput because the image upload feature uses the Base64String format, slowing the performance. This user edit endpoint could not be tested because, with 1,000 users per minute, the results made the microservice down. When the I/O-thread programming is implemented, this endpoint could be tested with 1,000 users per minute.

Table 6. Endpoint EditUserSKKM

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	10s	960	4%
2	5,000	10s	–	100%
3	10,000	10s	–	100%
4	15,000	10s	–	100%

The RequestSKKM endpoint also has parameters in the form of photos. Thus, the timeout used for this endpoint is ten seconds. The results are shown in Table 7 – the test conducted with 1,000 users obtained a throughput of 580 requests per minute. The microservice fails when tested with 5,000, 10,000, and 15,000 users in one minute. Table 8 shows the test results for endpoint DetailSKKM. The throughputs are close to the maximum, with 0% downtime.

Table 7. Endpoint RequestSKKM

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	10s	580	42%
2	5,000	10s	–	100%
3	10,000	10s	–	100%
4	15,000	10s	–	100%

Table 8. Endpoint DetailSKKM

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	4,898	2.04%
3	10,000	1s	9,998	0.02%
4	15,000	1s	14,996	0.03%

Tables 9 and 10 present the test results of endpoints SendSKKM and GetSKKMQueue, respectively. The SendSKKM endpoint test yields poor results due to the email server's limited performance. The test with 1,000 users generates 14.3% downtime. However, the throughput obtained in this test does not reflect the throughput generated by the NodeOS in the blockchain. These differences are caused by the high failure rate when sending emails due to timeout. Increasing the timeout would give the system enough time to serve all requests.

Table 9. Endpoint SendSKKM

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	857	14.3%
2	5,000	1s	–	100%
3	10,000	1s	–	100%
4	15,000	1s	–	100%

The endpoint GetSKKMQueue is used by the system administrator to retrieve all the requests of the student activity credit units from the queue. When this test was performed, the SKKM user queue collection data had reached 6,000 data. When testing was performed with 1,000 users per minute, the throughput was 998 requests per minute. The number of users per minute parameter is increased to 5,000, 10,000, and 15,000 which causes the test to fail to complete due to timeout. The results are shown in Table 10.

Table 10. Endpoint GetSKKMQueue

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	998	0.2%
2	5,000	1s	–	100%
3	10,000	1s	–	100%
4	15,000	1s	–	100%

The last endpoint is AcceptOrRejectSKKMRequest. The system administrator solely accesses this endpoint to accept or reject the request. The test results are given in Table 11.

Table 11. Endpoint AcceptOrRejectSKKMRequest

No	User per Minute	Timeout	Throughput (RPM)	Downtime
1	1,000	1s	1,000	0%
2	5,000	1s	5,000	0%
3	10,000	1s	10,000	0%
4	15,000	1s	12,222	18.52%

The highest throughput is delivered at the DetailSKKM endpoint by almost 15,000 requests per minute. The lowest throughput endpoint is at the RequestSKKM endpoint of 580 requests per minute. All endpoints successfully passed the 1,000 users per minute stress test. It is reasonable for some endpoints to suffer from the stress test due to limitations from supporting services such as the email server. Various operations carried out within the blockchain also affect performance. Queries with several parameters could also be quite time-consuming, which could cause specific endpoints to fail the test. Endpoint RequestSKKM and endpoint EditUserSKKM have a reasonably low throughput due to the image upload process.

This study contributes to designing and implementing a semi-decentralized information system using the EOSIO blockchain for real-world applications. The discussion focuses

on the design and implementation considerations of the decentralization approach for the information system. Since the organization uses the information system privately, the blockchain architecture regarding permission and access is closely controlled and managed by the owner and operator of the information system. Such an information system suits a private blockchain system, which enforces access control and permission to read and write transactions. This is the primary and first consideration in choosing the EOSIO blockchain in this study and is the foundation of other design and development considerations.

In this study, two smart contracts are developed and run in the blockchain. The first smart contract is for user wallet creation, and the second is for SKKM transactions. The decentralization level of the information system and the application use cases require only two smart contracts for the information system to operate thoroughly. This study shows that the degree of decentralized software application determines the complexity and number of smart contracts required. Decentralized applications rely entirely on smart contracts validating student activities and still require human intervention to analyze and decide. It is only feasible to implement some of the functionalities in smart contracts.

Adding more smart contracts increases transaction fees in the blockchain; thus, the cost to run this information system increases unnecessarily. Identify essential operations in the information system to be carried out in the blockchain network. The client-server solution best implements the information system's everyday operations and functions.

In this study, the information system has successfully used blockchain as its primary technology to store and manage the SKKM (student activity credit units) records. The records are distributed across the blockchain network. However, the application still has to rely on a central authority, the university, regarding the policy and software rules, including the operational service for the students. The decentralization approach allows the university to own and run the validator nodes in this study wholly. It is feasible and possible to upgrade the system architecture to be fully decentralized when the business process allows the services to be charged for fees, incentivizing validators.

4. **Conclusions.**

EOSIO blockchain provides the essential vitals to create a decentralized application. In this study, the blockchain is implemented and integrated with the web-based information system microservices. The information system consists of a blockchain, website, and database. The website provides easy access to the system functionalities, and users can also access the blockchain directly through the command line. The database only stores limited helpful information to synchronize the information displayed on the website and the blockchain. The system's core business process information is safely stored in the EOSIO blockchain. Microservice is designed and created to connect the website with the EOSIO blockchain.

Testing and evaluation show that the microservice's highest throughput is 14,996 RPM, with an overall throughput of 7,492 RPM. The results presented in the previous section are obtained from the stress test conducted using only one microservice instance. Naturally, adding more instances would deliver higher throughput (requests per minute). In this study, the test shows that the system performance is adequate to serve the organization's needs. Optimization could be done by adjusting the timeout parameter for specific endpoints, such as the RequestSKKM endpoint, to allow the photo uploading process. High throughput is essential for the system's quality of service (QoS). Adding more blockchain nodes would increase the throughput and improve the overall QoS.

Acknowledgment. The authors would like to thank Universitas Multimedia Nusantara for the support of this research work.

REFERENCES

[1] R. Azzi, R. K. Chamoun and M. Sokhn, The power of a blockchain-based supply chain, *Computers and Industrial Engineering*, vol.135, pp.582-592, DOI: 10.1016/j.cie.2019.06.042, 2019.

[2] R. Casado-Vara, J. Prieto, F. De La Prieta and J. M. Corchado, How blockchain improves the supply chain: Case study alimentary supply chain, *Procedia Computer Science*, vol.134, pp.393-398, DOI: 10.1016/j.procs.2018.07.193, 2018.

[3] B. B. A. Christyono, M. Widjaja and A. Wicaksana, Go-Ethereum for electronic voting system using clique as proof-of-authority, *Telkomnika (Telecommunication Computing Electronics and Control)*, vol.19, no.5, 1565, 2021.

[4] L. Mark, V. Ponnusamy, A. Wicaksana, B. B. Christyono and M. Widjaja, A secured online voting system by using blockchain as the medium, in *The Smart Cyber Ecosystem for Sustainable Development*, P. Kumar, V. Jain and V. Ponnusamy (eds.), Wiley, 2021.

[5] A. Wicaksana and J. C. Wira, Security analysis of private blockchain implementation for digital diploma, *International Journal of Innovative Computing, Information and Control*, vol.18, no.5, pp.1601-1615, DOI: 10.24507/ijicic.18.05.1601, 2022.

[6] W. Philips and A. Wicaksana, Hybrid approach of quick response code and non-fungible token in private permissioned blockchain for anti-counterfeiting, *International Journal of Innovative Computing, Information and Control*, vol.18, no.5, pp.1617-1632, DOI: 10.24507/ijicic.18.05.1617, 2022.

[7] C. A. Yeung, I. Liccardi, K. Lu, O. Seneviratne and T. Berners-Lee, *Decentralization: The Future of Online Social Networking*, https://www.w3.org/2008/09/msnws/papers/decentralization.pdf, 2008.

[8] Y. Guo and C. Liang, Blockchain application and outlook in the banking industry, *Financial Innovation*, 24, DOI: 10.1186/s40854-016-0034-9, 2016.

[9] R. Sytnyk, V. Hnatushenko and V. Hnatushenko, *Decentralized Information System for Supply Chain Management Using Blockchain*, http://ceur-ws.org/Vol-3156/paper45.pdf, 2022.

[10] R. Beck, M. Avital, M. Rossi and J. B. Thatcher, Blockchain technology in business and information systems research, *Business and Information Systems Engineering*, vol.59, pp.381-384, DOI: 10.1007/s12599-017-0505-1, 2017.

[11] S. I. M. Ali, H. Farouk and H. Sharaf, A blockchain-based models for student information systems, *Egyptian Informatics Journal*, vol.23, no.2, pp.187-196, DOI: 10.1016/j.eij.2021.12.002, 2022.

[12] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng and V. C. M. Leung, Decentralized applications: The blockchain-empowered software system, *IEEE Access*, vol.6, pp.53019-53033, DOI: 10.1109/ACCESS.2018.2870644, 2018.

[13] Y. Huang et al., Characterizing EOSIO blockchain, *arXiv.org*, arXiv: 2002.05369, 2020.

[14] W. Zheng, Z. Zheng, H. N. Dai, X. Chen and P. Zheng, XBlock-EOS: Extracting and exploring blockchain data from EOSIO, *Information Processing and Management*, vol.58, no.3, 102477, DOI: 10.1016/j.ipm.2020.102477, 2021.

[15] Y. Huang, B. Jiang and W. K. Chan, EOSFuzzer: Fuzzing EOSIO smart contracts for vulnerability detection, *arXiv.org*, arXiv: 2007.14903, 2020.