Applied Information Technology And Computer Science Vol. 2 No. 2 (2021) 053-072

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Electronic Medical Record System using Ethereum Blockchain and Role-Based Access Control

Loh Chee Ming, Chuah Chai Wen*

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, 86400 MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2021.02.02.004
Received 14 June 2021; Accepted 09 September 2021; Available online 30 November 2021

Abstract: Medical record is a document that records the disease, diagnosis, and treatment history of patients. These records help the doctor to determine the diseases and provides patients drug prescriptions. However, there is a case about the falsification of medical records. The doctor falsified the medical record to evade legal responsibility. Also, according to 13abc Action news, some healthcare organizations found that employees unauthorized access to medical records. The employee accessed the medical records without work-related purpose but for their own benefit. Hence, this project intends to develop a web-based Electronic Medical Record system to solve the address the issues mentioned above. This system store the hash value of medical records in blockchain as a reference to prevent falsification of medical records. The system also implements access control to restrict unauthorized access to medical records. The methodology uses to develop the proposed system is Object-Oriented System Development (OOSD). Java is the main programming language uses to develop the system. The developed system protects the confidentiality of medical records and provides an integrity check for medical records. This system may prevent falsification of medical records and restricts unauthorized access to medical records.

Keywords: Electronic Medical Record, Ethereum Blockchain, Role-Based Access Control, PBKDF2, Java

1. Introduction

Medical record records the disease, diagnosis, and treatment history of patients. The medical record contains the privacy of patients as it recorded the personal information and disease of patients. Medical records are very important as they help the doctor to determine the diseases of the patient and prevent doctors get a mistake when prescribe. Therefore, it is very essential to manage and store medical records securely [1].

There was a case of falsified medical records in October 2017. The physician assistant altered the medical record to evade legal liability and tried to shirk the responsibility. The falsified records were only found after a long time. Finally, doctors are not legally responsible for falsifying medical records [2]. Therefore the integrity of medical records must be ensured to prevent the doctors from falsifying

*Corresponding author: cwchuah@uthm.edu.my

the records. The proposed system restricts the doctors to modify the medical records. Doctors are only allowed to create new medical records for patients. Also, the proposed system stores the medical record's hash value in the blockchain. By comparing the hash value of the medical records stored in the database and the hash value stored in the blockchain, it may detect any unauthorized modification of medical records.

Healthcare organizations face legal responsibility for unknown employee's unauthorized access, retrieve, and disclosing the patient's medical records. According to the 13abc Action News report, there was a former employee unauthorized access and modification of medical records in ProMedica [3]. An academic medical centre, Penn Medicine also found that a former medical assistant had improperly accessed patient records [4]. To overcome above mentioned problem, the proposed system records all activities of staff account in the audit log. It provides accountability in the system. The activities such as who accessed which patient's medical records are recorded in the audit log. Based on the audit log report, the healthcare administrator may deactivate or legal action towards the employee is suspended to access the medical records without work-related reason. The proposed system also implementing Role-Based Access Control (RBAC) to control the accessibility of medical records.

In the proposed system, there are four types of users: administrator, doctors, nurses and patients. The hospital administrator registers an admin account, the admin register the new staffs such as receptionist, doctor, and nurse. For the doctors, after consultation they add patient's medical record. The doctors may access or view the granted medical records from their patients. For nurse, they assist doctors to request for particular patients to grant access to their medical records. The patient may view their own medical records and control access their medical records. The proposed system stores the hash values of the medical records in the blockchain. The system refers to the hash values stored in the blockchain to check the integrity of medical records. The system also protects accountability by record the activities of the staff account in the audit log, and the admin is able to view this audit log.

2. Literature Review

This section presents the literature reviews that have been conducted for this project. The goal of this literature review is to understand the background and technology used for this system.

2.1 Hash Function

Hash function takes an arbitrary size of data as input to produce a fixed length output, called hash value [5]. It is impossible reverse the output of the hash function to reconstruct original data. Also, single bit change in the input of hash function may get a completely different output hash value. The same input always produces the same hash value [6]. Hash function consists of three main properties. There are Preimage Resistance, Second-Preimage Resistance, and Collision Resistance [5].

Table 1 describes each of the hash function properties.

Table 1: Properties of Hash function [5]

Properties	Description
Pre-Image Resistance	By given a hash value x, then using y as an input to hash function H, the output of $H(y)$ should not equal to x. It could be expressed as $H(y) \neq x$.
Second Pre-Image Resistance	By given input x, then using a different value y as input, the output of $H(y)$ should not equal to the output of $H(x)$. It could be expressed as $H(y) \neq H(x)$ if $y \neq x$.

Table 1: (cont.)

Properties	Description
Collision Resistance	By given values *x* and *y* as inputs, if *x* and *y* are different, then the output of $H(x)$ should not equal the output of $H(y)$. It could be expressed as $x \neq y, H(x) \neq H(y)$.

SHA-256 hash function is used as the internal hash function of HMAC in the proposed system. SHA-256 takes an arbitrary size of data as input. The block size of SHA-256 is 512 bits. The output of SHA-256 is 256 bits [7].

The SHA256 algorithm follows these steps [7]:

- Get the input message and ensure its length is a multiple of 512 bits. This is done by adding a padding.
- Take the pass message and parse it into N 512-bit blocks.
- Iterate over all blocks from step 2:
    - Initialize the message schedule, a sequence of 64 32-bit words.
    - Initialize eight working variables *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h* with the hash values H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7 from the previous iteration (for the first iteration, H_0 to H_7 are initialized with constants).
    - Perform 64 iterations where working variables a to h are rotated in a certain manner. In this step, the message is inserted into the hash using lots of bitwise mixing.
    - Compute the new intermediate hash values H_0 to H_7 as $H_0 = H_0 + a$, $H_1 = H_1 + b$ and so on.
- Concatenate H_0 to H_7 as the message digest and return it.

2.2 Hash Message Authentication Code (HMAC)

Hash Message Authentication Code (HMAC) secures the transmission of data over insecure channels, especially the integrity, authentication, and non-repudiation of data. HMAC uses a private key and cryptographic hash function to generate output hashes [5].

The HMAC algorithm express as [5]:

$$\text{HMAC}\,(K, M) \;=\; \text{H}((K^{+} \oplus \text{opad}) \,||\, \text{H}((K^{+} \oplus \text{ipad}) \,||\, M)) \qquad Eq.1$$

For the parameters in Eq.1, H is SHA256 hash function, K is the secret key, K^{+} is the block-sized key derived from secret key, M is the message, opad is outer padding, ipad is inner padding, || is concatenation, and $\oplus$ is exclusive OR [5].

First, the message is converted into bits and divided into blocks with block size b bits. The block size depends on the hash function used. The secret key, K is known by both sender and receiver. The secret key, K is padded with extra zero if the key size is less than the block size, b. If the key is longer than the block size, hash the key to getting the b size key. The key remains if the key size equal to the block size. The block-sized key, K^{+} is derived from the secret key, K [8].

The ipad is 00110110 repeated $b/8$ times. While the opad is 01011100 repeated $b/8$ times. First, the block-sized key, K^{+} XOR with ipad to produce output, S_i with block size. Next, concatenate the S_i with the message and then hash it with n bits IV to generate output with n bits size. Then, pad the output to b bits size [8].

The key, K^+ XOR with opad to produce output with block size, S_0. Concatenate the S_0 and the output together. Lastly, hash it again with IV to produce n bits output. It is the final result of HMAC [8].

2.3 Password Based Key Derivation Function 2 (PBKDF2)

Password-hashing techniques are used to protect the password stored in the database [9]. The current standardization of password-hashing is PBKDF2 [10], while other widely used standards are Bcrypt and Scrypt [9].

PBKDF2 is defined by the choice of a Pseudorandom Function (*PRF*) and iteration count, *C.* The Pseudorandom Function (*PRF*) is HMAC-SHA256. PBKDF2 takes three inputs, which are password, salt, and length of derived key [10]. Table 2 describes the three inputs of PBKDF2.

Table 2: Inputs for PBKDF2 [10]

Inputs	Description
Password	User's password.
Salt	Random generated number that appended to the password.
Derived key Length	Length of the derived key in octet.

The output of PBKDF2 is the master key. The PBKDF2 is expressed as [10]:

$$mk = PBKDF2_{(PRF,C)}(P, S, kLen) \quad Eq.2$$

For the parameters in Eq.2, P is the password, S is the salt, $kLen$ is the derived key length, PRF is HMAC with SHA-256, C is iteration count, and mk is the master key [10].

The process of PBKDF2 [9]:

1. HMAC function takes the password and salt as inputs to produce the HMAC result.
2. Password and previously-computed HMAC result taken as inputs for the HMAC function to produce the HMAC result.
3. The HMAC result XOR with the previously-computed HMAC result in each iteration.

The final hash value is produced from the last round XOR operation.

2.4 Blockchain

In simple terms, blockchain is a chain of blocks. Each block stored multiple transactions. Every block is interconnected to the previous block through its hash value. Blockchain consist of a continuous sequence of blocks. The new block was added linearly to the "tail" of the blockchain [11].

After the block added to the blockchain, the content in the block is very difficult to be modified. Every block in the blockchain contains the hash value of current block data along with the hash value of the previous block. Any modification may affect the current hash and the previous hash stored in the next block become incorrect. Therefore the unauthorized modification is detected. [11].

Each block in blockchain consists of a block header and a block body. The block body contains a transaction counter and transactions. The size of a block and the size of each transaction determine the maximum number of transactions that a block can contain [11]. Block size varies from 1MB to 8MB and may over 8MB. Each block has a unique number known as Block ID generated cryptographically [6].

Block Header is composed of six fields, there are block version, Merkle Tree Root hash, timestamp, nBits, nonce, and parent block hash [11]. Table 3 shown the block header field and its description.

Table 3: Components of block header [11]

Components	Description
Block version	A 4-byte number specifies the version of the blockchain.
Merkle tree root hash	The hash value of all transactions in the current block body.
Timestamp	The current block added time in Unix time seconds.
nBits	The threshold of valid block hash in the current block.
Nonce	The generated number used to get target hash value.
Parent block hash	The hash value of previous block.

Blockchain has provided four security properties to secure the transactions stored in the blocks. There are decentralization, persistency, anonymity, and auditability [11][12].

- Decentralization
  Blockchain distributes the transaction information on the blockchain network. The transactions are stored by the blockchain nodes. The transactions on the blockchain network are verified by the nodes on the network through the consensus algorithm. The consensus algorithm maintains data consistency.

- Immutability
  Every block in the blockchain contains its hash value and also the previous block hash value. Therefore the block able to detect and restricts any unauthorized modification of data. Once the block is added to the blockchain, it is nearly impossible to remove or rollback the transaction. Any unauthorized modification and invalid transaction on the blockchain could be detected by the nodes on the blockchain network.

- Anonymity
  Blockchain provides an efficient way of hiding the identity of users and keeps the user's identity secret. Every user uses the generated address such as wallet address to interact with the blockchain. Therefore it does not disclose the real identity of users.

- Auditability
  Every transaction information is recorded in the block and authenticated by the digital signature of the users. Therefore the activities of the users when interacting with the blockchain become non-repudiation. All of the transactions that could be verified can be tracked easily.

2.4.1 Ethereum Blockchain Platform

Ethereum is an open-source blockchain platform. Ethereum blockchain was launched in 2015. Ethereum uses the cryptocurrency called Ether (ETH). The Ether is transferable between user accounts. Users must paid Ether when perform a transaction on the Ethereum blockchain [13].

Ethereum also called programmable blockchains since Ethereum supports smart contracts. A smart contract is a contract in digital form and it is a non-modifiable general purpose computer program that stored in Ethereum blockchain. Ethereum not only hosts the smart contracts but also executes it. Every user may deploy smart contracts to the Ethereum blockchain. After deployment successfully, the smart contract has a 40-digit hexadecimal ID which is often referred to as the address of the contract. User interacts with the smart contract through the address of the contract [13].

Users interact with the smart contract by sending transactions to its functions. The transaction burns a certain amount of gas units depending on the number of instructions executed during runtime. The

amount of gas units used in transaction is called gas usage. To send a transaction to the smart contract, it has to set two parameters: gas price and gas limit. Gas price is the amount of Ether to pay for one unit of gas. Gas limit is the maximum amount of gas units to pay for transaction. Therefore, the transaction fee that has to pay is gas price × gas usage, and the maximum transaction fee is gas price × gas limit [13].

2.5 Electronic Medical Record (EMR)

The medical records should be universally available, for example, access through the World Wide Web [14]. Universally available medical records are easily shared when patients transfer to different hospitals. However, there is potential of loss of information privacy via the remote access to the electronic medical record through the network. The attackers may hack the database to unauthorized access the medical records through the network [14].

Based on [15], there are seven requirements to secure the electronic medical record. Table 4 describes the requirements.

Table 4: Requirement to secure electronic medical record [15]

Requirements	Description
Authorized access	The system should be able to identify both healthcare providers and patients. Identification should be portable between different roles that access patient medical records.
Confidentiality	Medical records contain sensitive patient information, thus it is important to secure the confidentiality of records.
Patient's consent	Patients should able to control their medical records by allowing or deny others access to their medical records. Access to medical records should have the consent of the patient, especially share the medical records among different healthcare providers.
Information ownership	Healthcare providers are responsible for the patient information, but patients also have the right to access their medical records.
Relevance	Only allow the relevant entity to access the patient information.
Information consistency	The system should able to show the changes of information. Also, detect and restrict unauthorized modification of data.
Audits	The system should record all of the activities about access to the information and any modification of medical records. This mechanism allows monitor the activities on the system, thus provide accountability.

2.6 Existing Electronic Medical Record System

This section explains two Electronic Medical Record System (EMR), Hospital Health Information Management System (HHIMS) [16] and MedBloc [17]. Section 2.6.1 presents Hospital Health Information Management System (HHIMS) [16]. Section 2.6.2 explains MedBloc [17]. Section 2.6.3 compares the two systems with the proposed system.

2.6.1 Hospital Health Information Management System (HHIMS)

Hospital Health Information Management System (HHIMS) developed by Dompe District Hospital in Sri Lanka in the year 2010. HHIMS was developed for the Out Patient Department (OPD). HHIMS is used to store clinical details of patients treated, aimed to replace paper medical records [18]. HHIMS has become an open-source medical record software now. Information Communication and Technology Agency (ICTA) implemented and managed this system currently [18].

HHIMS needs to install on a computer server in the hospital so that the Out Patient Department (OPD) and other departments able to use workstation computers to access the system via a local area

network. The server store all of the data such as patient information, no data stored in the workstation computer. Since the data stored centrally on the server, the data able to share with the workstation computers that are connected to the local area network [19].

2.6.2 MedBloc

MedBloc is a blockchain-based system. MedBloc enables healthcare organizations and patients to access and share health records in a secure channel. MedBloc uses encryption and smart contract-based access control to secure medical data [17].

MedBloc allows the patients to own their medical records by allowing or deny others access. The Immutability of blockchain allows patients to comfort their data are not unauthorized modified by others. The blockchain transparent property also enables data auditability and provenance. Patients are able to know who access their medical records and how their data are used [17].

2.6.3 Comparison of Existing System with Proposed System

Table 5 compares the Hospital Health Information Management System (HHIMS), MedBloc, and Proposed System with six properties. The six properties are confidentiality, integrity, accountability, patient's consent, accessibility, and audit log.

Table 5: Comparison of existing systems with proposed system

	HHIMS	MedBloc	Proposed System
Confidentiality	✓	✓	✓
Integrity	✗	✓	✓
Accountability	✗	✓	✓
Patient's consent	✗	✓	✓
Accessibility	✗	✓	✓
Audit log	✗	✗	✓

Based on Table 5, HHIMS only protects the confidentiality of medical records. While MedBloc and the proposed system implement confidentiality, integrity, accountability, patient's consent, and accessibility in the system. The proposed system also implements audit log property compares to MedBloc. The proposed system provides an audit log which records all of the staff's activities such as create and access to medical records event. Admin is able to monitor the staff's activities by view the audit log.

3. Methodology

Object-Oriented System Development (OOSD) is a methodology used to design, analyse, and develop a system or application. There are four phases in OOSD. There are object-oriented requirement analysis, object-oriented analysis, object-oriented design, and object-oriented implementation and testing [20].

In object-oriented requirement analysis phase, the requirements for the proposed system are analysed based on the academic paper related to Electronic Medical Record (EMR) and blockchain. The academic paper analysed include articles, journals, and conference papers published in IEEE, PubMed Central (PMC), and Research Gate.

In the object-oriented analysis phase, the information collected are analyzed. The modules for the proposed system are identified. There are seven modules, which are the register module, login module, staff account control module, audit log module, doctor consultation module, medical record module, and patient permission module. After identified the modules, functional requirements and non-

functional requirements are examined. The user roles for the system also identified. The roles are admin, receptionist, doctor, nurse, and patient.

In object-oriented design phase, the architecture of the proposed system is designed. The database table, the classes, and the user interface (UI) are designed for the proposed system.

In object-oriented implementation phase, the database tables, classes, and user interface (UI) are implemented. All of the tables and classes are linked to each other to ensure the proposed system function correctly. Also, the system is developed based on the design and requirements.

In the object-oriented testing phase, the testing process follows the designed test plan to ensure the proposed system functioning correctly. The debugging process undergoes to solve the problems when errors occur in the proposed system. The penetration test process is done on the system using two tools. These tools are Pentest-Tools.com and SSL Trust.

After deployment of the system, there may have some problems occur. For example, lack of functionality and failure to pass the expectation of users. There should have modifications to conform to the users. However, due to the limitation of time, the object-oriented maintenance phase is not going to implement in this project.

4. Analysis and Design

This section presents the analysis and design for the proposed system.

4.1 Functional Requirements

The proposed system has seven functional requirements as shown in Table 6. There are register module, login module, staff account control module, audit log module, doctor consultation module, medical record module, and patient permission module.

Table 6: Functional requirements for system

Modules	Descriptions
Register	• The hospital administrator registers an admin account by using username, password, and hospital details such as hospital name and address. • The patient registers a user account by using username, password, and personal information such as name, date of birth, email, and address.
Login	• Authenticate users such as admin, doctor, and patient log in to the system by using username and password.
Staff Account Control	• Admin adds, updates, and deletes staff accounts. • Admin deactivates the staff account.
Audit Log	• Admin views the access log that records all the activities of staff such as create new medical record, and access medical records.
Doctor Consultation	• Receptionist adds patients to the waiting list. • Receptionist assigns doctor for the patients. • Doctor view the current assigned patient waiting list.
Medical Record	• Doctor adds a new medical record to the patient. • Doctor and nurse view the medical records granted access by the patients. • Doctor views the medical records created by themselves. • Patient views own medical records.

Table 6: (cont.)

Modules	Descriptions
Patient Permission	• Doctor and nurse request and view the medical records of the patient. • Patient allows or denies the request to access their own medical records.

4.2 Non-functional Requirements

There are three categories of non-functional requirements, which are operational, performance, and security. Table 7 shows the non-functional requirements for the proposed system.

Table 7: Non-functional requirement for system

Requirement	Description
Operational Performance Security	• System only available when there is Internet connection. • System must locate to the correct session depend on who authorized. • User may access the system with correct username and password. • User's password should have minimum 10 characters, include at least one capital letter, at least one small letter, at least one number, and at least one symbol. • System automatic terminates the session and log out the user after inactivate for 30 minutes. • System hashes the user password with Password Based Key Derivation Function (PBKDF2).

4.3 General System Architecture

General system architecture defines the structure and behaviour of a system. Figure 1 illustrates the General system architecture for proposed system.

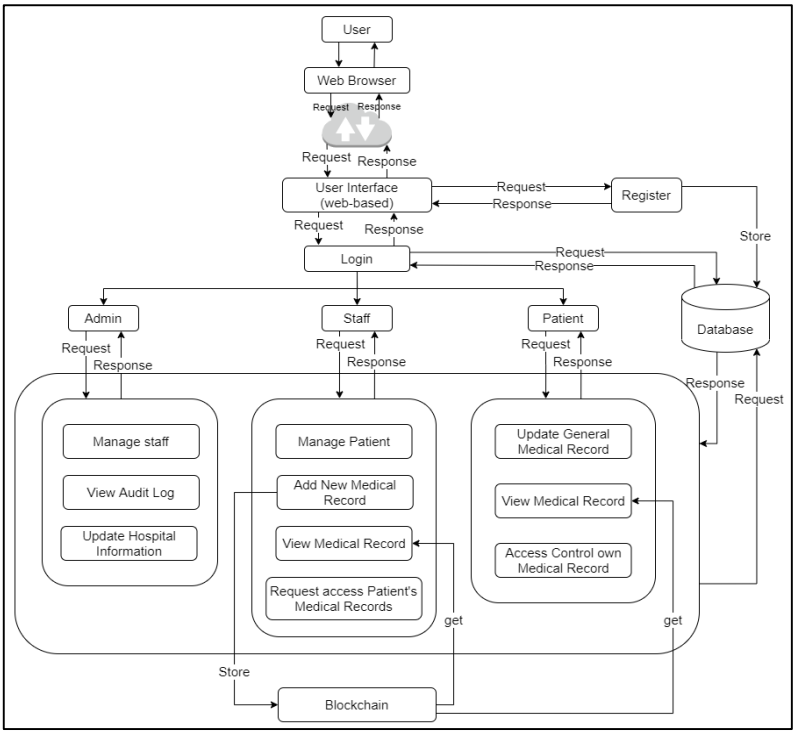

Figure 1: General system architecture for proposed system

As shown in Figure 1, the proposed system is web-based. User uses a web browser to access the system through Internet. The system stores the data in the database. The system retrieves the data from the database when user requests, then displays it to the user. The system stores the hash value of medical records in the blockchain.

There are three types of users in the proposed system, which are admin, staff, and patient. Every user able to login through the login user interface. Only admin and patient are allowed to register new account. Admin performs three functions, there are manage staff, view audit log, and update hospital information. There are three types of staff, which are receptionist, doctor, and nurse. Receptionist manages patients by adding them to the waiting list and assign doctor for patients. Doctor able to perform three actions, add medical records, view medical records, and request access to patient's medical records. Nurse may view medical records and request access to patient's medical records. For the patient, there have four actions can perform, which are update general medical record, view medical record, and access control own medical record.

4.4 Classes Diagram

This section illustrates the relationship between classes implement in the proposed system. Figure 2 shows the class diagram for the proposed system.

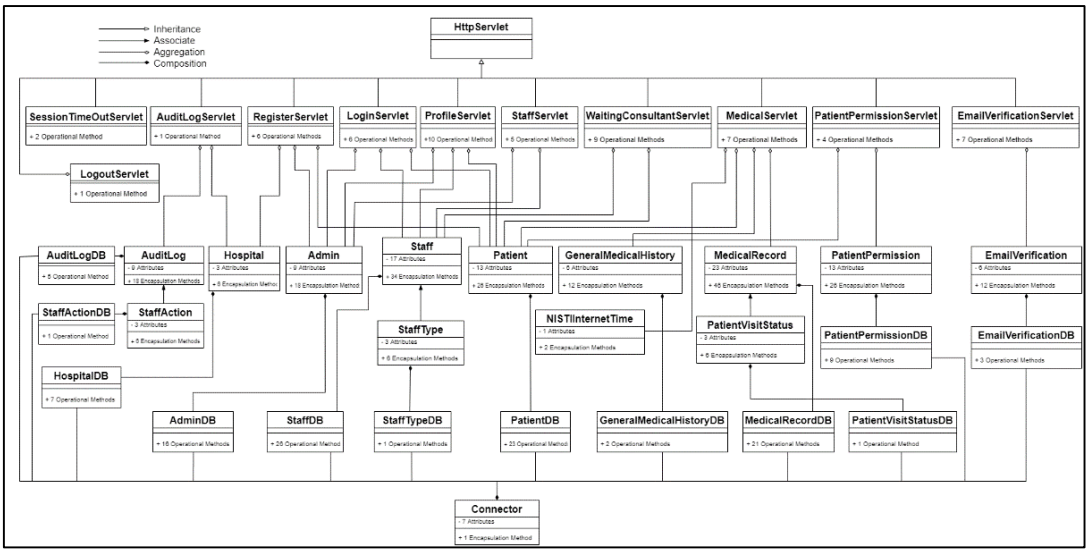

Figure 2: Class Diagram for proposed system

The class diagram contains 11 servlets and 26 classes. The servlets are HttpServlet, SessionTimeOutServlet, AuditLogServlet, RegisterServlet, LoginServlet, ProfileServlet, StaffServlet, WaitingConsultantServlet, MedicalServlet, PatientPermissionServlet, EmailVerificationServlet, and LogoutServlet. The servelets handle request from user and response to the user. Servlets contain two methods which are doGet() and doPost().

The classes are Connector, AuditLog, StaffAction, Hospital, Admin, Staff, StaffType, Patient, GeneralMedicalHistory, MedicalRecord, PatientVisitStatus, PatientPermission, EmailVerification, and NISTInternetTime. Each of the class has getter and setter method for every attributes, except the Connector class which is used to connect the database.

The DB classes are used to interact with the database to store and retrieve data. The DB classes are AuditLogDB, StaffActionDB, HospitalDB, AdminDB, StaffDB, StaffTypeDB, PatientDB, GeneralMedicalHistoryDB, MedicalRecordDB, PatientVisitStatusDB, PatientPermissionDB, and EmailVerificationDB.

4.5 Entity Relationship Diagram

The ERD of the proposed system has 13 entities. These entities are EMAIL_VERIFICATION, HOSPITAL, ADMIN, STAFF, STAFF_TYPE, PATIENT, GENERAL_MEDICAL_HISTORY, MEDICAL_RECORD, PATIENT_VISIT_STATUS, PATIENT_PERMISSION, AUDIT_LOG, STAFF_ACTION, and MEDICAL_CONTRACT. Each entity has its own primary key, ID. The HOSPITAL, ADMIN, STAFF, PATIENT, GENERAL_MEDICAL_HISTORY, MEDICAL_RECORD, PATIENT_PERMISSION, and AUDIT_LOG entities have foreign key related to others. The MEDICAL_CONTRACT entity is stored in the Ethereum blockchain. The ERD for the proposed system is shown in Figure 3.

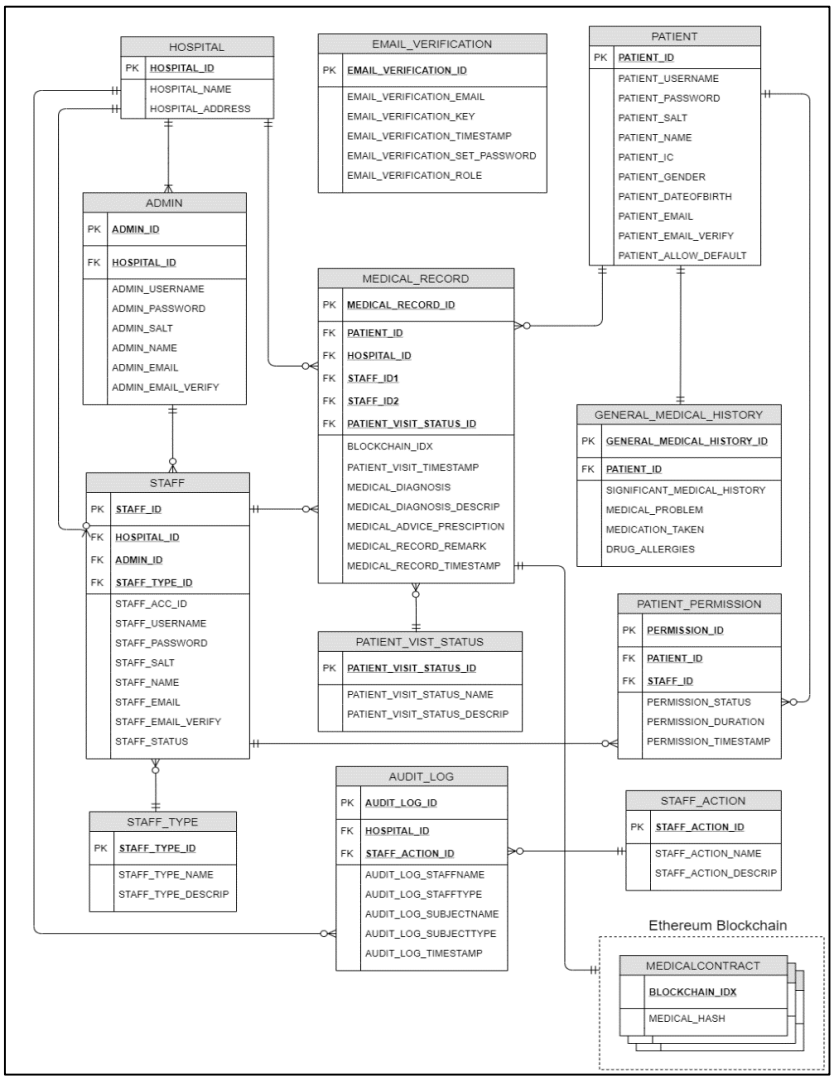

Figure 3: ERD for proposed system

5. Implementation

This section examines the cost of cryptocurrency required to interact with Ethereum blockchain smart contract and implementation of the proposed system. The system is implemented by following the system security requirements and electronic medical record security properties.

5.1 Cost of Cryptocurrency to Interact with Ethereum Blockchain Smart Contract

The developed system stores the hash value of medical records on Ethereum blockchain by interact with smart contract. To send a transaction to the smart contract, it has to set two parameters: gas price

and gas limit. Gas price is the amount of Ether to pay for one unit of gas. Gas limit is the maximum amount of gas units to pay for transaction. The transaction fee that has to pay is gas price × gas usage, and the maximum transaction fee is gas price × gas limit. Gas prices are denoted in Gwei, each Gwei is equal to 0.000000001 ETH (10^-9 ETH). This system sets the gas price as 20 Gwei, equal to 0.00000002 ETH. This system also sets the gas limit as 3000000 unit. Therefore, the maximum transaction fee needed to pay for each transaction is 0.06 ETH. Most of the time, each transaction fee is cost lower than 0.06 ETH.

5.2 Implementation of System Security Module

This section presents the security module implemented in the proposed system.

5.2.1 Implementation of Strong Password

Figure 4 shows the Java code for implementing strong password in the system. The password is not allow to be empty. The minimum length and maximum length are set for the password. The minimum length of password is 10 characters and maximum length of password is 20 characters. The pattern of password must consists of at least one capital letter, one small letter, one number, and one symbol. The system will reject the registration and print error message if fail to meet these requirements.

```
public static String checkPassword(String password) {
    String error = null;

    if(checkEmpty(password)) {
        error = "Password cannot be empty.";
    }
    else if(password.length() < 10) {
        error = "Minimum of 10 characters is required for password.";
    }
    else if(password.length() > 20) {
        error = "Maxminmum of 20 characters is allowed for password.";
    }
    else {
        String passwordRegex = "^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#\\$%\\^&\\*])[0-9a-zA-Z!@#\\$%\\^&\\*]{10,}$";
        Pattern passwordPattern = Pattern.compile(passwordRegex);
        Matcher passwordMatcher = passwordPattern.matcher(password);
        if(!passwordMatcher.matches()) {
            error = "Password must consist of minimum 10 characters, include at least one capital letter, small letter, number, and symbol.";
        }
    }

    return error;
}
```

Figure 4: Strong Password

5.2.2 Implementation of Secure HTTP Response Header

The Secure HTTP Response Header enhance the security of the web page by restrict the browser to load data from unknown source. Figure 5 shows the Java function to set HTTP response header for every page send to the users.

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
    if(request.isSecure() && response instanceof HttpServletResponse) {
        HttpServletResponse resp = (HttpServletResponse) response;
        resp.addHeader("Content-Security-Policy",
                       "default-src 'self'; "
                       + "script-src 'self' 'unsafe-inline' 'unsafe-eval' 'unsafe-hashes' https://unpkg.com/web3@latest/; "
                       + "connect-src 'self' https://ropsten.infura.io/v3/29c0676ab1154efeaa0db2563f92959c; "
                       + "font-src *; "
                       + "img-src 'self' data: 'unsafe-inline'; "
                       + "style-src * 'self' 'unsafe-inline'; "
                       + "base-uri 'self'; "
                       + "form-action 'self';");
        resp.addHeader("Strict-Transport-Security", "max-age=31622400; includeSubDomains");
        resp.addHeader("X-Frame-Options", "DENY");
        resp.addHeader("X-XSS-Protection", "1; mode=block");
        resp.addHeader("X-Content-Type-Options", "nosniff");
        resp.addHeader("Referrer-Policy", "no-referrer");
    }
    chain.doFilter(request, response);
}
```

Figure 5: Set HTTP Response Header

5.3 Implementation of Electronic Medical Record Security Module

This section presents the electronic medical record security properties implemented in the proposed system.

5.3.1 Implementation of Authorized Access Property

Authorized access property indicates the system should identify both hospital staff users and patient users. System authenticates and authorizes users with their username, password, and roles. Figure 6 shows the Java code for staff authentication. Staff is able to access the system if the username, password, and staff type are correct.

```
staff = StaffDB.getStaffByUsernameAndStaffType(staffUsername, staffType);
if (staffUsername.equals(staff.getStaffUsername()) && staffPassword.equals(staff.getStaffPassword())) {
    if (staff.isStaffEmailVerify()) {
        if (staff.isStaffStatus()) {
            AuditLogDB.addAuditLog(setAuditLog(staff));// store login action into audit log

            StaffSession staffSession = new StaffSession();
            staffSession.setStaffID(staff.getStaffID());
            staffSession.setStaffType(staff.getStaffTypeName());
            staffSession.setStaffAccID(staff.getStaffAccID());
            staffSession.setStaffUsername(staff.getStaffUsername());
            staffSession.setStaffName(staff.getStaffName());
            staffSession.setStaffEmail(staff.getStaffEmail());
            staffSession.setHospitalID(staff.getHospitalID());

            session.setAttribute("staffSession", staffSession);
            session.setMaxInactiveInterval(30 * 60);

            return true;
        }
        else {
            request.setAttribute("staffStatus", "inactive");
            return false;
        }
    }
```

Figure 6: Authentication of Staff

5.3.2 Implementation of Confidentiality Property

Confidentiality property indicates that the system should protect the confidentiality of data. The proposed system protects the confidentiality of password stored in the database by PBKDF2 algorithm. Password is hashed with salt before stored into database. Figure 7 shows the Java function for PBKDF2 which hash the password and salt together to produce 512 bytes of the hash value. Figure 8 shows the system stores the password and salt together in the database, the password is stored as hex string which is not readable by human to protect the confidentiality of password.

```
protected String hashPassword() {
    final char[] passwordChars = getPassword().toCharArray();
    final byte[] saltBytes = getSalt().getBytes();

    try {
        SecretKeyFactory skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
        PBEKeySpec spec = new PBEKeySpec(passwordChars, saltBytes, 10000, 64 * 8);
        SecretKey key = skf.generateSecret(spec);
        byte[] hashedBytes = key.getEncoded();
        return Hex.encodeHexString(hashedBytes);
    } catch (NoSuchAlgorithmException | InvalidKeySpecException e) {
        throw new RuntimeException(e);
    }
}
```

Figure 7: PBKDF2 function

STAFF_PASSWORD	STAFF_SALT
50118272b373354c56e93bcac3e77f37ad628af...	5z3ukm4RVxM9516Me8V5k9vXEug
b72c4792d13f9d808a7d36652d9528490ae969...	K-OVy3obsDxduT3xfLRcsmwWbGI
5d43a8416ef6e31590b1397157663a98e4ca28f...	5Gswrc2QFpGxTVpovxvJ6GjLnt8

Figure 8: Password and Salt stored in database

5.3.3 Implementation of Patient's Consent Property

Patient's consent property indicates that hospital staff (doctor or nurse) needs to request patients to access their medical records. Patients can control their medical records by allow or deny others access to their medical records. Figure 9 shows hospital staff (doctor or nurse) interface to search patients by patient's name or IC to request access to their medical records. Figure 10 shows the patient interface to views the access request list. Patient may accept or reject staff (doctor or nurse) access to their medical records.

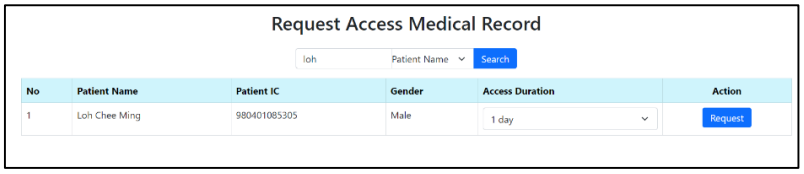

Figure 9: Request Access Medical Record (Hospital Staff)

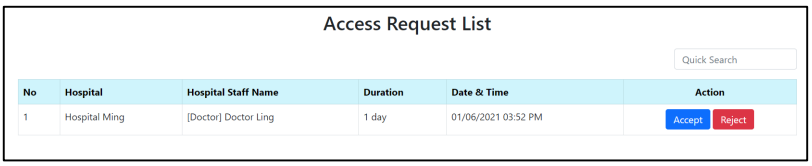

Figure 10: Access Request List (Patient)

5.3.4 Implementation of Patient's Consent Property

Information ownership property indicates that the patients have the right to access their own medical records. Figure 11 shows the system display patient's medical records in table form. Patients may click the "View" button to view the detail of the medical record.

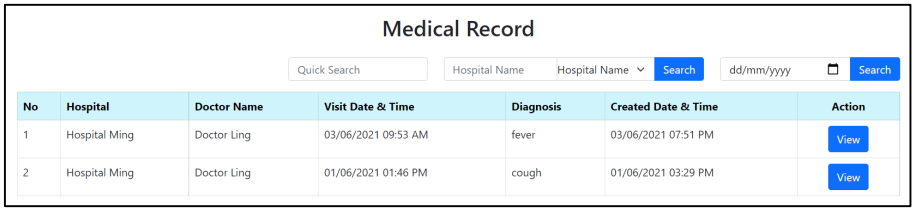

Figure 11: Patient View Medical Record

5.3.5 Implementation of Relevance Property

Relevance states that only the relevant entity allow to access the patient's medical records. System implements relevance property by using Role-Based Access Control (RBAC). System ensures only relevance staff users access to certain page to perform actions. Figure 12 shows the Java code for redirect page based on role. The system redirects the staff to the page based on their staff type after login successfully. The staff only can perform actions authorized for their staff type.

```
if (session.getAttribute("adminSession") != null && staffType.equals("admin")) {//admin login successful
    response.sendRedirect("AdminPanel");// after login successfully, redirect to AdminPanel
    return;
}
else if (session.getAttribute("staffSession") != null) {//staff login successful
    if (staffType.equals("receptionist")) {
        response.sendRedirect("ReceptionistPanel");
        return;
    }
    else if (staffType.equals("doctor")) {
        response.sendRedirect("DoctorPanel");
        return;
    }
    else if (staffType.equals("nurse")) {
        response.sendRedirect("NursePanel");
        return;
    }
}
```

Figure 12: Redirect staff to page based on role

5.3.6 Implementation of Information Consistency Property

Information consistency property indicates that the system should able to ensure the consistency of data and detect any unauthorized modification of data. The proposed system displays error message to warn the users when detect any unauthorized modification of medical record. System stores the hash

value of medical record in the Ethereum blockchain for reference to check the integrity of the medical records. Figure 13 shows the Javascript function that stores hash value into the Ethereum blockchain. This function contract with the smart contract on the Ethereum blockchain to store the medical record's hash value on the blockchain. Figure 14 shows the system print error message when the hash values in database and blockchain are different.

```
function pushHashToBlockchain(medicalRecordHash) {
    var pushHash = contract.methods.pushHash(medicalRecordHash);

    //encodeABI pushHash Solidity method
    var encodedABI = pushHash.encodeABI();

    //prepare transaction data
    var tx = {
        from: ownerAddress,
        to: contractAddress,
        gas: 3000000,
        gasPrice: 20000000000,
        data: encodedABI
    };

    //sign and send transaction
    web3.eth.accounts.signTransaction(tx, privateKey).then(signed => {
        var tran = web3.eth.sendSignedTransaction(signed.rawTransaction);
        tran.on('receipt', receipt => {
            storeBlockchainIdx(web3.eth.abi.decodeParameter('uint256', receipt.logs[0].data));
    });
        tran.on('error', console.error);
    });
}
```

Figure 13: Store Hash value to Ethereum Blockchain

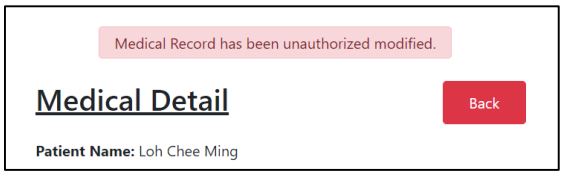

Figure 14: Error message for unauthorized modified medical record

5.3.7 Implementation of Audits Property

Audits property indicates that the system should record the activities of the staff in the audit log. This mechanism allows the admin to monitor the staff's activities on the system, thus providing accountability. Figure 15 shows the system display audit log in table form for admin. Admin may search the audit log by staff name, subject name, or date.

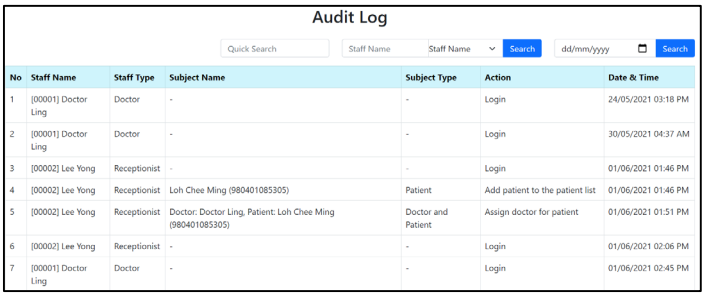

Figure 15: Audit Log

6. Result and Discussion

This section presents the security test plan result, penetration testing result, and user acceptance form result.

6.1 Result Test Plan

Table 8 shows the result of security test plan for the proposed system. There are 12 security test plans. The developed system had passed all these test plans.

Table 8: Security Test Plan Result

No	Check List	Actual Result
1	Ensure the error message not direct indicate which part of the authentication data incorrect. For example, error message should not show "incorrect username" or "incorrect password".	**Pass**
2	Enforce the password complexity inside the policy. For example, require password with minimum 10 characters and maximum 20 characters, at least one capital letter, at least one small letter, at least one number, at least one symbol.	**Pass**
3	Password should be obscured in the text box.	**Pass**
4	Ensure users only can perform actions based on the role.	**Pass**
5	Auto Logout user after inactivity for 30 minutes.	**Pass**
6	Ensure user is not allowed to reset password using expired email link or email link that already been used.	**Pass**
7	Minimum and maximum length, and invalid format input in input field are denied.	**Pass**
8	User must input correct old password and input new password satisfied the strong password requirement to change password.	**Pass**
9	Password in the database is hashed with PBKDF2 algorithm.	**Pass**
10	Salt is added to the password before hashed with PBKDF2 algorithm.	**Pass**
11	Medical record only allows to update when diagnosis, diagnosis description, medical advice and prescription, remark, and medical record timestamp columns are empty.	**Pass**
12	Display warning message when the medical record was unauthorized modified.	**Pass**

6.2 Penetration Testing Result

Pentest-Tools.com and SSL Trust are two pentest tools that used to scan the developed system. Based on the Pentest-Tools, the proposed system has the secure response HTTP header include Strict-Transport-Security, Content-Security-Policy, X-Frame-Options, X-XSS-Protection, X-Content-Type-Options, and Referrer-Policy. These security headers restrict the browser from download malicious content when load the web page and prevent Cross-Site Scripting (XSS) attack. Figure 16 shows the pentest result of Pentest-Tools.com.

Figure 16: Pentest-Tools.com Result

For SSL Trust, there are two types of scan performed. The first type of scan is for malware and virus detection. The second type of scan is for vulnerability examination. For the malware and virus section, the result shows that the developed system did not compromised by any malware and virus. SSL Trust report shows the proposed system has a secure connection through TLS protocol with a valid certificate. For the vulnerability scan report, the proposed system is secure from attacks such as

Heartbleed, CCS, ROBOT, DROWN, and LOGJAM. Figure 17 shows the certificate check and protocol check result.

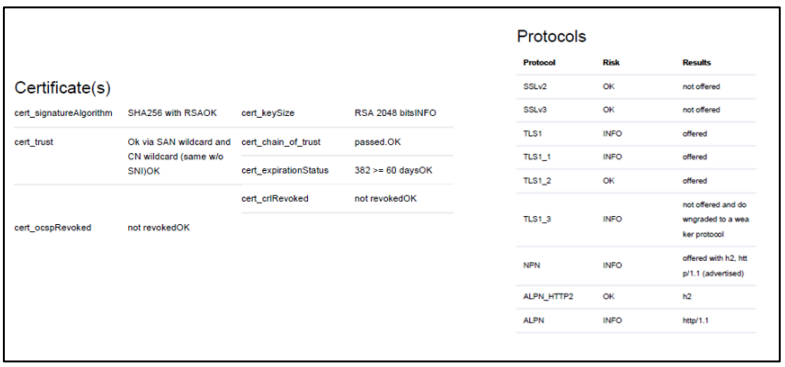

Figure 17: SSL Trust Certificate Check and Protocol Check Result

6.3 User Acceptance Result

The user acceptance form collects data from users by using Google Form. There are two user acceptance forms which are for doctor and patient. The form consists of three sections. Section A collects respondent information, Section B collects testing result, and Section C collects feedback.

6.3.1 User Acceptance (Doctor) Result

For the doctor acceptance form, there are three question in Section A which collect information of the responders include name, email, and job scope. There are four questions in Section B which collects testing result from responders as shown in Table 9. Section C collects feedback from the responders. There is one question ask for suggestions and comments of responder regarding the system.

Table 9: User Acceptance Form for Doctor (Section B) Questions

No	Section B Questions	Test Result	
		Yes	No
1	Doctor may add new medical record for patients.		
2	Doctor cannot modify saved medical records.		
3	Doctor only can view the medical records created by him.		
4	Doctor requests patients to access patient's medical records.		

There are 2 responders for the user acceptance form for doctor. Both 2 responders are clinic doctors from AMC Clinic Seremban 2. Figure 18 shows their responses for the four questions in Section B. Based on the chart, two of the responders choose “Yes” for all of the four questions. The result shows that the system passes the test from the doctor users.

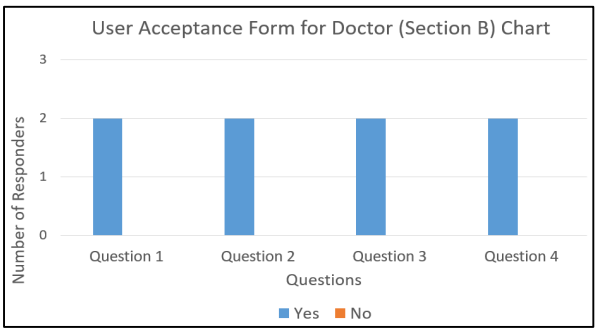

Figure 18: User Acceptance Form for Doctor (Section B) Chart

For the feedback in Section C, two of the responders give the same comment which is add medical record process is slow. They suggest to speed up the process of add medical record.

6.3.2 User Acceptance (Patient) Result

For the patient acceptance form, there are three question in Section A which collect information of the responders include name, email, and job scope. There are four questions in Section B which collects testing result from responders as shown in Table 10. Section C collects feedback from the responders. There is one question ask for suggestions and comments of responder regarding the system.

Table 10: User Acceptance Form for Patient (Section B) Questions

No	Section B Questions	Test Result	
		Yes	No
1	Patient can update own general medical history.		
2	Patient can view own medical records.		
3	Patient can accept or reject doctor or nurse access to their medical records.		
4	Patient gets error message for incorrect pattern of password (minimum 10 characters, at least one uppercase, one lowercase, one number, and one symbol) input when registration.		

There are 5 responders for the user acceptance form for patient. Both 5 responders are students. Figure 19 shows their responses for the four questions in Section B. Based on the chart, five of the responders choose "Yes" for all of the four questions. The result shows that the system passes the test from the doctor users.

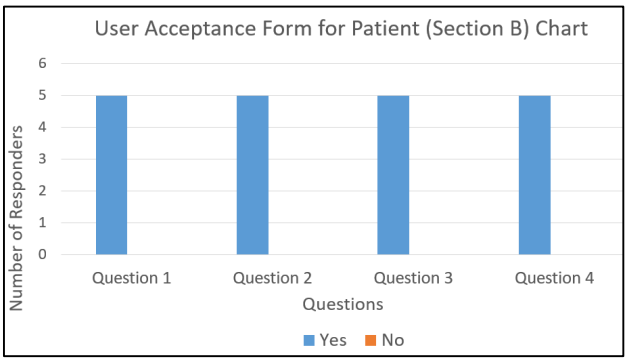

Figure 19: User Acceptance Form for Patient (Section B) Chart

For the feedback in Section C question, two of the responders give the same suggestion that is to add booking consultation function for the system. One of the responders gives a comment which is nice and simple design website. Two of the responders did not give any comments and suggestions.

7. Conclusion

The proposed system, Electronic Medical Record System using Ethereum Blockchain and Role-Based Access Control has been developed successfully and achieved all of the objectives and fulfil the security requirements of electronic medical record. The proposed system stores the medical record's hash value in the blockchain. By comparing the hash value of the medical records stored in the database and the hash value stored in the blockchain, it may detect any unauthorized modification of medical records.

The proposed system protects the confidentiality of medical records by implement role-based access control. The system only allows specific role such as doctor to access created medical records.

Also, the doctos and nurses only allowed access or view the granted medical records from their patients. Patients are able to grant or revoke doctors and nurse to access their medical records.

For the future works of the proposed system, improvement can be done by enhancing the security of the system. The future works for electronic medical record system included:

- Add a function to ensure every user creates a private key during registration for interaction with the Ethereum blockchain. It may provide security when the user uses the system to interact with the blockchain.
- Redirects the user to the root page when their session is destroyed after inactivate for 30 minutes.
- Add a notification function to notify patients when there are new access medical records requests from hospital staff.

Acknowledgement

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support and encouragement throughout the process of conducting this study.

References

[1] J. Tsai and G. Bond, "A comparison of electronic records to paper records in mental health centers," Int. J. Qual. Heal. Care, vol. 20, no. 2, pp. 136–143, 2007.

[2] R. Painter, "Doctor And Physician Assistant Alter Patient's Medical Record—You Won't Believe What Happened Next," Painter Law Firm, 2017.

[3] S. Hegarty, "Former ProMedica employee suspected of stealing identities," 13abc Action News, 2019.

[4] A. Whelan, "Former Penn medical assistant accused of improperly accessing patient records," The Philadelphia Inquirer, 2019.

[5] A. A. Alkandari, I. F. Al-Shaikhli, and M. A. Alahmad, "Cryptographic Hash Function: A High Level View," in 2013 International Conference on Informatics and Creative Multimedia, 2013, pp. 128–134.

[6] D. K.N. and R. Bhakthavatchalu, "Parameterizable FPGA Implementation of SHA-256 using Blockchain Concept," in 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 370–374.

[7] S. Sanadhya and P. Sarkar, Attacking Reduced Round SHA-256, vol. 2008. 2008.

[8] D. Ravilla and C. S. R. Putta, "Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs," in 2015 International Conference on Electronic Design, Computer Networks Automated Verification (EDCAV), 2015, pp. 154–159.

[9] G. Hatzivasilis, "Password-Hashing Status," 2017.

[10] M. S. Turan, E. Barker, W. Burr, and L. Chen, "Recommendation for password-based key derivation," NIST Spec. Publ., vol. 800, p. 132, 2010.

[11] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in 2017 IEEE International Congress on Big Data (BigData Congress), 2017, pp. 557–564.

[12] H. Atlam, A. Alenezi, M. Alassafi, and G. Wills, "Blockchain with Internet of Things: Benefits, Challenges and Future Directions," Int. J. Intell. Syst. Appl., vol. 10, Jun. 2018.

[13] G. A. Oliva, A. E. Hassan, and Z. M. J. Jiang, “An exploratory study of smart contracts in the Ethereum blockchain platform,” Empir. Softw. Eng., pp. 1–41, 2020.

[14] R. C. Barrows Jr. and P. D. Clayton, “Privacy, Confidentiality, and Electronic Medical Records,” J. Am. Med. Informatics Assoc., vol. 3, no. 2, pp. 139–148, Mar. 1996.

[15] J. J. P. C. Rodrigues, I. de la Torre, G. Fernández, and M. López-Coronado, “Analysis of the security and privacy requirements of cloud-based electronic health records systems,” J. Med. Internet Res., vol. 15, no. 8, pp. e186–e186, Aug. 2013.

[16] S. Kulathilaka, ““eHospital-Dompe “project--the story of the transformation of a district hospital in Sri Lanka,” Sri Lanka J. Bio-Medical Informatics, vol. 4, no. 2, 2013.

[17] J. Huang, Y. W. Qi, M. R. Asghar, A. Meads, and Y. Tu, “MedBloc: A Blockchain-Based Secure EHR System for Sharing and Accessing Medical Data,” in 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), 2019, pp. 594–601.

[18] K. Mendis et al., “Cloud-Based Open Source Primary Care Electronic Patient Record System for Sri Lankan Citizens,” in 2019 National Information Technology Conference (NITC), 2019, pp. 41–46.

[19] hhims.org, “Hospital Health Information Management System User Handbook.” 2011.

[20] A. R. Hevner, “Object-Oriented System Development Methods,” vol. 35, M. C. B. T.-A. in C. Yovits, Ed. Elsevier, 1992, pp. 135–198.