EQUIVA JOURNAL
Journal of Mathematics & Information Technology
Volume 01 – Nomor 01 – Year 2023 – Pages: 49-67

IMPLEMENTATION OF BLOCKCHAIN AND NON FUNGIBLE TOKEN (NFT) TO VALIDATE THE AUTHENTICITY AND OWNERSHIP OF OFFICIAL DOCUMENTS

Nur Fajri Azhar[1, a)], Muhammad Nasa'i Kairupan[2, b)], Ahmad Rusdianto Andarina Syakbani[3, c)], Ansar Fadillah[4, d)]

[1,2,3,4]*Department of Mathematics and Information Technology, Informatics, Kalimantan Institute of Technology, Balikpapan, Indonesia*

[a)]Penulis korespondensi: fajri@lecturer.itk.ac.id
[b)]11201065@student.itk.ac.id
[c)]11211005@student.itk.ac.id
[d)]11201012@student.itk.ac.id

Abstract

Data security and authenticity of ownership of private documents such as land certificates and identity cards or KTP are critical aspects of Indonesia's development process. Database system that has been imple- mented by the government to secure citizen's personal information is still susceptible to 6 major risk factors, namely, data breaches and theft, damaged data due to virus attacks, limited data storage space, data loss due to system damage, data can be accessed without certain access rights, as well as a lack of system certainty when security threats occur. The rampant cases of data counterfeiting and selling of KTP and land certificates have been a problem for the Indonesian government and have claimed lives for a long time. Based on the problem of public service management, this proposal suggests the development of a public service application for KTP and land certificates based on blockchain titled NFT Document Inspector to help mitigate risks con- tained in the Indonesian government's database system as well as ensuring the authenticity of KTP and land certificates. The results of this research will provide insights for policy makers, practitioners and researchers regarding the adoption of blockchain mechanisms to secure citizen's private data in order to accelerate the digital transformation of public services for Indonesia to recover faster and rise stronger, which in this case is KTP and land certificate. It is possible to create system information application of KTP and land certificates as public service with blockchain as a database system. Blockchain provides semi-immutable data storage, hashed data, and log activities that improves data transparency, data security, and data reliability, still it is a daunting task to implement it in real-life due to blockchain's scalability and infrastructureproblems.

INTRODUCTION

Data security and authenticity of ownership of private documents such as land certificates and identity cards or KTP are critical aspects of Indonesia's development process [1]. The Indonesian government is trying to pro- vide the best service for Indonesian citizens in ensuring the security of their personal data and the authenticity of their personal documents [2] in protecting their property rights [3]. Database system that has been implemented by the government to secure citizen's personal information is still susceptible to 6 major risk factors, namely, data breaches and theft, damaged data due to virus attacks, limited data storage space, data loss due to system damage, data can be accessed without certain access rights, as well as a lack

of system certainty when security threats occur [4]. The rampant cases of data counterfeiting and selling of KTP [5] and land certificates [6] have been a problem for the Indonesian government and have claimed lives for a long time. Based on the problem of citizen's private documents in Indonesia, a solution is needed to ensure the security and authenticity of KTP data and land certificates stored in the government database system, in which case we propose the concept of a information system management application using blockchain technology.

Blockchain is a distributed ledger that uses public-key encryption and consensus protocol verifying the authenticity to record data on nodes called blocks in a secure, transparent, decentralized, cost-effective, and time-efficient manner [7]. Every node in the blockchain network that participates in computing has a copy of the data and all parties validate every new block added to the blockchain through a consensus validation [8]. It can be perceived as a group of people sharing data using no intermediary agents, where trust between all the involved parties can still be built since all parties can see all the occurring transactions in the blockchain network [9]. Blockchain also has a high level of security because of its own nature such as the use of an eternal ledger, a chain of blocks that are interrelated with the hash value of each previous data block so that if you change the data of a random block it will affect the entire blockchain making it have highly sensitive data and information, and decentralized in nature so that every node in the network must have the same data ledger according to consensus or the data block will be rejected. This makes the blockchain protected from fraudulent transactions from hackers [10].

Previous studies [11] [12] [13] argue that blockchain has enormous potential in public services, because it can overcome problems such as human error, data privacy, and security. Studies on the implementation of blockchain technology in public services model [4] and citizen's private data security systems [14] along with land certificates [15] have been carried out, however the implementation of a public service management in- formation system as an application that uses blockchain technology still does not exist. Based on the problem of public service management, this proposal suggests the development of a public service application for KTP and land certificates based on blockchain titled NFT Document Inspector to help mitigate risks contained in the Indonesian government's database system as well as ensuring the authenticity of KTP and land certificates. The results of this research will provide insights for policy makers, practitioners and researchers regarding the adop- tion of blockchain mechanisms to secure citizen's private data in order to accelerate the digital transformation of public services for Indonesia to recover faster and rise stronger, which in this case is KTP and land certificate.

RESEARCH METHOD

Research method of NFT Document Inspector application development consists of requirements analysis, system design, application development, system testing and analysis. Requirements analysis is carried out first to find specifications that are needed to create blockchain-based information system software needs and man- agement of KTP and land certificates by conducting an analysis of relevant national or international journals, also by looking for Indonesian laws and regulations related to KTP and land certificates. System design is carried out after the application requirements analysis process is finished by choosing an effective and efficient application's architecture system and application's user interface design that suits blockchain-based information system software for KTP and land certificates management. Application development is carried out by devel- oping blockchain-based information system software for KTP and land certificates management applications that match system design that has been created before. The application is web based because blockchain is web related (Web3) and also for ease of development. System testing and analysis is carried out after the application development process is finished by matching the development result with the requirements analysis and system design that has been done before. System testing serves as a benchmark for the suitability of application devel- opment with research objectives.

RESULT & ANALYSIS

The NFT Document Inspector application creates and stores KTP and land certificate data as Non-Fungible Token (NFT) on the blockchain through a smart contract which is created using the Solidity

programming language. Smart contracts are program code stored in the blockchain and executed when specified conditions are met. Smart contracts are used to automate the execution of a contract or agreement so that all participants can immediately be sure of the outcome without the need for a third party or wasted time. Smart contracts can be used to automate a workflow to trigger further actions when conditions are met [16].

Non-Fungible Token (NFT) is a digital asset on the blockchain that has a unique identification address and metadata as a characteristic that differentiates it from others. NFT can represent physical objects from the real world whose process is called tokenization. NFT assets cannot be exchanged and cannot be replicated due to the unique characteristics of the token assets. NFTs are technically smart contracts in that they are programmable and cannot be modified once launched [17].

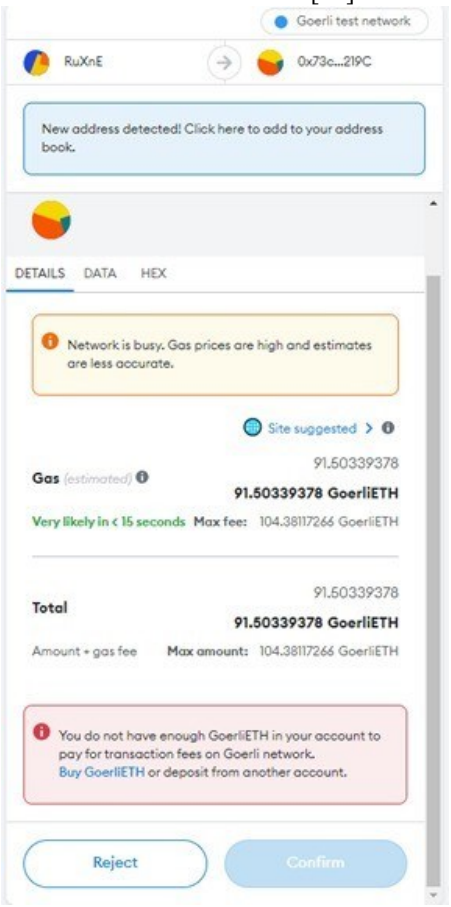

Figure 1: Transaction attempt to save a KTP photo with a size of 247 KB.

The blockchain network used to create and store NFT KTP smart contracts and land certificates is Ethereum. Ethereum itself is a blockchain network that can create decentralized applications, store asset data, carry out transactions and communicate without third party authority. Ethereum has its own cryptocurrency, namely Ether. Ether is used to pay ”taxes” for certain transactions [18], which are related to the data management process in the NFT Document Inspector application, such as doing create, update, and delete operation on data that is stored in Ethereum blockchain.

It is necessary to store data on the application form for making a KTP using a traditional database, which in this case we use PostgreSQL. Photos of KTPs registered in smart contracts also need to be stored in traditional databases because storing images in JPG/PNG format on the blockchain is very computationally expensive, which results in very expensive transaction fees on public blockchains, such as Ethereum.

This is because Ethereum uses the concept of gas as the required fee to run a transaction, which in this

case is a transaction for registering KTP data and photos into the blockchain. The fee to pay for gas in ethereum is called gas fee, which can change in value. The concept of gas in Ethereum can be analogous to gasoline which is used by cars and motorbikes to run. The gas fee in gasoline is like the cost of 1 liter of gasoline which price can change at any time depending on the government's decision, even though the amount of gasoline remains the same, which is 1 liter. Based on figure 1, the transaction fee for storing one data along with a photo of a KTP is worth around 750 ETH or if converted to Rupiah becomes Rp. 15,151,217,107.50 (this price is the result of converting Ethereum to Rupiah when the document is made, prices can change). This cost is very expensive and impossible to implement in the real world, however, this can be prevented by the government creating its own blockchain network or private blockchain to store KTP and land certificate because there is no need for a gas concept to replace computational costs during transactions.

Researchers in this case use PostgreSQL to store KTP and land certificates registered on the Blockchain as a temporary alternative solution because creating a private blockchain for the government requires further research.

Data Model of Application Form for Generating KTP Table in PostgreSQL

Property Name	Data Type	Is Primary Key	Is Unique
address Wallet	String	Yes	Yes
nama	String	No	No
kotaLahir	String	No	No
provinsiKotaLahir	String	No	No
tanggalLahir	Date	No	No
jenisKelamin	Enum	No	No
statusPerkawinan	Enum	No	No
golonganDarah	Enum	No	No
kewarganegaraan	Enum	No	No
pekerjaan	String	No	No
agama	Enum	No	No
berlakuHingga	Enum	No	No
alamat	String	No	No
rukunTetangga	String	No	No
rukunWarga	String	No	No
kelurahan	String	No	No
kecamatan	String	No	No
kotaKtpDibuat	String	No	No
provinsiKtpDibuat	String	No	No
statusValidasi	String	No	No
namaFile	String	No	No
tipeMime	String	No	No
ukuran	Integer	No	No

Table 1: Data Property for KTP.

Table 1 is the property for form KTP, namely the primary key in the form of an addressWallet to store the address of the MetaMask wallet for which to create a KTP so that it is registered in the blockchain, personal data (such as name, city of birth, etc.), validation status as a parameter for an application for if a KTP is approved or not, filename, mimetype, and photo size to store information related to KTP photo, createdAt and updatedAt which are created automatically as a determinant of the date when data was created and changed. KTP photos are stored in the assets folder and stored according to the type of document (KTP or land certificate) with the photo file name based on the addressWallet stored in the table. Stored data can be accessed using an API that has been built with Node.js by the researcher.

```
Router.get("/", async (req, res, next) => {
  try {
    const ktpData = await FormKtp.findAll();
    return res.json(ktpData);
  } catch (error) {
    next(error);
  }
});
```

Figure 2: API Route to Get All KTP Data

Figure 2 is the API for returning all KTP data that is registered in blockchain. The code attempts to fetch all the information stored in the "FormKtp" database table by utilizing the "FormKtp.findAll()" method. The retrieved data is then assigned to a constant variable named "ktpData", when the client sends a request, the "res.json(ktpData)" function sends the data stored in "ktpData". The "next(error)" function directs the error to the subsequent middleware within the Express.js error handling sequence if any error occurs.

```
Router.get("/:walletAddress", async (req, res, next) => {
   try {
     const ktpData = await FormKtp.findByPk(req.params.addressWallet);
     if (!ktpData) {
       return res.status(404).end();
     }
     return res.json(ktpData);
    } catch (error) {
     next(error);}
 });
```

Figure 3: API Route to Find KTP Data by Owner's Wallet Address

Figure 3 is the API for returning a KTP data that you want to register into the blockchain according to the requested MetaMask wallet address. The code listens to the "/:walletAddress" endpoint, when a request is made with a specific wallet address as a parameter, it retrieves the related data from the "FormKtp" database using "FormKtp.findByPk" and the "req.params.addressWallet" value. The router returns a 404 error and ends the response if the data is not found, but, if the data exists, the data is sent back to the client as a JSON response.

```
Router.post("/", ktpImgStorage.single("file"), async (req, res, next) => {
  try {
    const { filename, mimetype, size } = req.file;
    const statusValidasi = "DIPROSES";
    const newKtp = await FormKtp.create({
      ...req.body,
      statusValidasi,
      namaFile: filename,
      tipeMime: mimetype,
      ukuran: size,
    });

    return res.json(newKtp);
  } catch (error) {
    next(error);
  }
});
```

Figure 4: API Route to Create New KTP Data

```
Router.delete("/:walletAddress", async (req, res, next) => {
  try {
    const ktpData = await FormKtp.findByPk(req.params.addressWallet);
    if (!ktpData) {
      return res.status(404).end();
    }

    await ktpData.destroy();
    return res.status(204).end();
  } catch (error) {
    next(error);
  }
});
```

Figure 5: API Route to Delete KTP Data by Owner's Wallet Address

Figure 4 the API for registering data along with KTP selfies into the PostgreSQL database. The new KTP's status will be set "DIPROSES" by default and stored on the database before the KTP can be validated. The server creates new KTP data using "FormKTP.create" with values from req body and req file when the request is made with a certain request body and request file. The server will return new KTP data after successfully saving the new KTP. The "next(error)" function directs the error to the subsequent middleware within the Express.js error handling sequence if any error occurs during this process.

Figure 5 is the API for deleting KTP data registered in the database based on the MetaMask wallet address. The code deletes KTP data by listening to the "/:walletAddress" using the "delete" HTTP method. The method retrieves the related data from the "FormKtp" database using "FormKtp.findByPk" and the "req.params.addressWallet" value when a request is made with a specific wallet address as a parameter. The method returns a 404 error and
ends the response if the data is not found, meanwhile the data will be deleted from the database using the "de- stroy" method if the data is found. The "next(error)" function directs the error to the subsequent middleware within the Express.js error handling sequence if any error occurs during this process.

```
Router.put("/:walletAddress", async (req, res, next) => {
  try {
    const ktpData = await FormKtp.findByPk(req.params.addressWallet);
    if (!ktpData) {
      return res.status(404).end();
    }

    const keys = Object.keys(req.body);

    keys.forEach((key, index) => {
      ktpData[key] = req.body[key];
    });

    await ktpData.save();
    return res.json(ktpData);
  } catch (error) {
    next(error);
  }
});
```

Figure 6: API Route to Edit KTP Data by Owner's Wallet Address

Figure 6 is the API for changing KTP data registered in the database based on the MetaMask wallet address. The data that is changed is in accordance with the request with the limitation that the data that is changed in the column is in accordance with the model that has been made.

Model Property of KTP Photo Table in PostgreSQL

```
{
  addressWallet: {
    type: DataTypes.STRING,
    primaryKey: true,
    unique: true,
    allowNull: false,
  },
  imgString: {
    type: DataTypes.STRING(500000),
    allowNull: false,
  }
}
```

Figure 7: Model for KTP Photo

Figure 7 is the property of the KTP photo. KTP photo table consists of 4 data columns, namely the primary key in the form of addressWallet to store the MetaMask wallet address whose KTP is registered in the blockchain, imgString to store KTP photos in base64 string form, createdAt and updatedAt which are gener- ated automatically as date determinants when data is created and changed. Data stored in these tables can be accessed using an API that has been created with Node.js by the researcher.

```
Router.get("/", async (req, res, next) => {
  try {
    const ktpData = await FotoKtp.findAll();
    return res.json(ktpData);
  } catch (error) {
    next(error);
  }
});
```

Figure 8: API Route to Get All KTP Photo

Figure 8 is the API for returning all data in the KTP photo table, in the form of addressWallet and imgString. The code attempts to fetch all data that is stored in the ”FotoKtp” database table by utilizing the ”FotoKtp.findAll()” method. The retrieved data is then assigned to a constant named ”ktpData”. The ”res.json(ktpData)” function sends the data stored in ”ktpData” when the client sends a request. The ”next(error)” function directs the er- ror to the subsequent middleware within the Express.js error handling sequence if any error occurs during this process.

Figure 9 is an API for returning data based on the MetaMask wallet address registered in the blockchain, in the form of addressWallet and imgString according to the request. The code listens to the ”/:walletAddress” endpoint. The method retrieves the requested data from the ”FormKtp” database using ”FotoKtp.findByPk” and the ”req.params.addressWallet” value when a request is made with a specific wallet address as a parameter. The method will return a 404 error and end the response if the data is not found. The data will be sent to the client as a JSON response if the data is found.

```
Router.get("/:walletAddress", async (req, res, next) => {
  try {
    const ktpData = await FotoKtp.findByPk(req.params.addressWallet);
    if (!ktpData) {
      return res.status(404).end();
    }
  return res.json(ktpData);
  } catch (error) {
    next(error);
  }
});
```

Figure 9: API Route to Find KTP Photo by Owner's Wallet Address

```
Router.post("/", async (req, res, next) => {
  try {
    const { addressWallet, imgString } = req.body;
    if (!(addressWallet && imgString)) {
      return res.status(400).json({ error: "Masukkan address wallet
      dan string base64 dari fotonya!" });
    }

    const fotoKtp = await FotoKtp.create({ ...req.body });
    return res.json(fotoKtp);
  } catch (error) {
    next(error);
  }
});
```

Figure 10: API Route to Create New KTP Photo

Figure 10 is the API for adding KTP photos registered in the Blockchain into the KTP photo table. The input data is in the form of Wallet address and imgString registered in the blockchain and sent in JSON form. The "next(error)" function directs the error to the subsequent middleware within the Express.js error handling sequence if any error occurs during this process.

```
Router.delete("/:walletAddress", async (req, res, next) => {
  try {
    const fotoKtp= await FotoKtp.findByPk(req.params.addressWallet);
    if (!fotoKtp) {
      return res.status(404).end();
    }

    await fotoKtp.destroy();
    return res.status(204).end();
  } catch (error) {
    next(error);
  }});
```

Figure 11: API Route to Delete KTP Photo

Figure 11 is an API for deleting KTP photos in the KTP photo table based on the MetaMask wallet address registered in the blockchain network. The method retrieves the requested data from the ”FotoKtp” database using ”FotoKtp.findByPk” and the ”req.params.addressWallet” value when a request is made with a specific wallet address as a parameter. It will return a 404 Not Found error and end the response if the data is not found. The data will be deleted from the database using the “destroy” method if the data is found. The ”next(error)” function directs the error to the subsequent middleware within the Express.js error handling sequence if any error occurs during this process.

The API for changing KTP photo data is not created because there are only two data fields, so it's enough to delete and re-create the data when it wants to be updated.

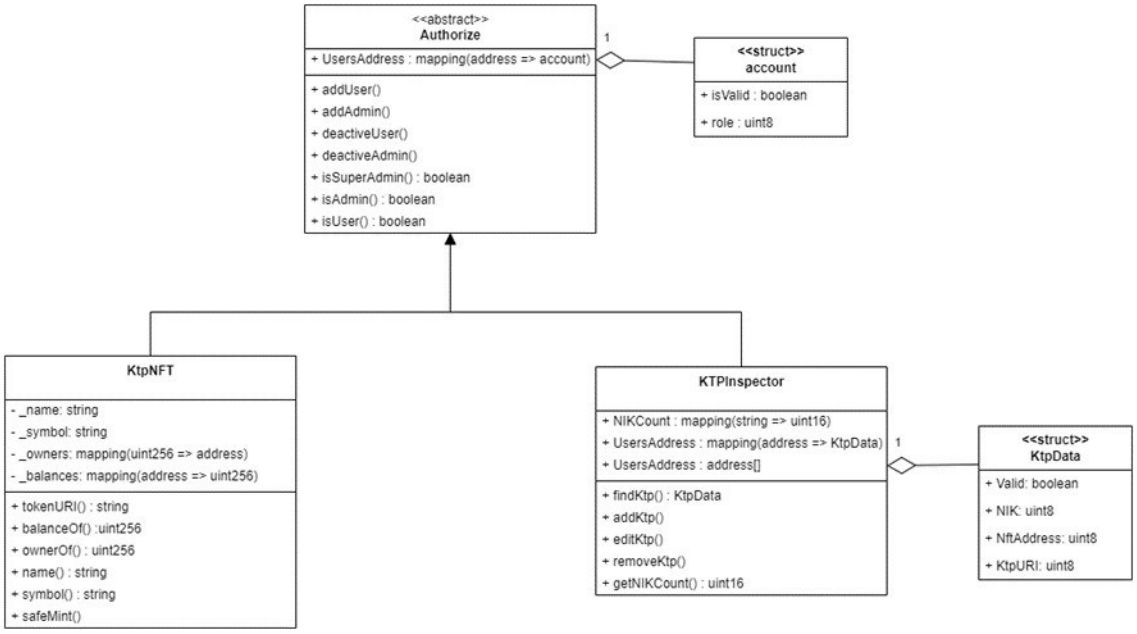

Figure 12: KTP Smart Contract Class Diagram

Smart Contract Design

The development of this application uses smart contracts as program instructions for data storage in the blockchain so an appropriate smart contract design is needed to create and store KTP data safely.

Authorize contract contains user accounts as a struct and maps it with the user's wallet address. Account struct contains two properties which are isValid, for the account validity, and role, for account access authoriza- tion to the smart contract. isValid property can also act as soft-deletion when the value is set to false. There are also some contract methods to support contract functionality. Some contract methods need to only be accessed by certain authorized users for example like admin or superadmin. Admin can interact with every contract method except contract methods relating to the right of authority while super admin can interact with every contract method like admin including right of authority. Method addUser is used for adding a new account as an user, this method can only be interacted with by admin and super admin because it requires certain regula- tions in adding new users. Method addAdmin is used for adding user accounts as an admin and can only be called by super admin because admin can call crucial smart contract's method that holds user's private data, therefore super admin is needed as the highest role for authorization to control user's application access rights, especially when adding a new admin. Method deactivateUser can be interacted with by admin and super admin for deactivating users and can be called by super admin, method deactivateUser can be used when there is a malicious user in the application. Method deactivateAdmin for deactivating admin and can be called only by super admin,method deactivateAdmin can be used when there is a malicious admin in the application. Method isSuperAdmin for checking

superadmin status of a registered user. Method isAdmin for checking admin status of a registered user. Method isUser for checking validation status of a registered user.

KtpNFT contract contains the name of the NFT, symbol of the NFT, owners mapping by token id, and NFT balances mapping by owners. There are also some methods to support contract functionality. Method tokenURI to get token data by Id, Method balanceOf for checking owner's NFT balance, Method ownerOf for checking owner of NFT by Id,and Method safeMint can be interacted with by admin and super admin to create new KTP NFT because only authorized users can make KTP which are super admin and admin.

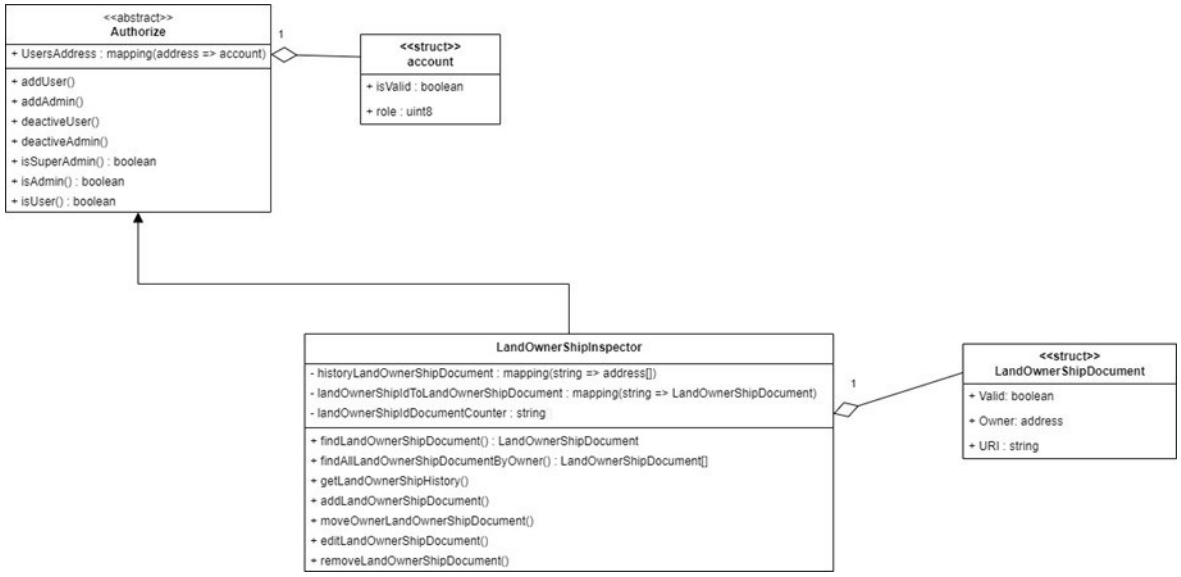

Figure 13: KTP Smart Contract Class Diagram

KTPInspector contract contains NIK count for generating NIK for new KTP, KTP owner counter stores all owners wallet addresses, and Ktp data struct and maps it with every user's wallet address. KTP data struct has properties Valid for the KTP data validity, NIK or id of KTP, address of KTP NFT, and KTP data. There are also some methods to support contract functionality. Method findKtp to get KTP data by owner's address. Method editKTP is used for editing KTP data and can only be called by admin and super admin. Method editKTP cannot edit NIK because the NIK must be unique to every person and it will never change in the entire lifetime of the user. Method removeKtp is used for removing a KTP data by owner's address wallet and can be called only by admin and super admin because only authorized users can remove KTP data which are super admin and admin. Method getNIKCount for count KTP that have the same first 12 digits of NIK and can only be called by admin and super admin. Method addKtp for adding new KTP data to blockchain and can be interacted with by admin and super admin because data KTP contains sensitive information so only certain authorized users can add new KTP data into the blockchain network. Method addKtp generates new NIK for users by getting the NIK code count from blockchain and concat it to the NIK code. Every time method AddKTP is called, the counter of NIK code will be incremented.

Figure 13 contains a smart contract class diagram for Land Certificate.The smart contract is divided into two contracts which are Authorize and LandOwnerShipInspector. Authorize contract is used as an authorization setting for smart contract access rights by managing users role to interact with the smart contract. LandOwn- erShipInspector contract is used to store mapping of land ownership data as well as ownership history of each land ownership document and as an interface between the users and the blockchain in validating and searching stored Land Ownership data.

Authorize contract contains user accounts as a struct and maps it with the user's wallet address. Account struct contains two properties which are isValid, for the account validity, and role, for account access authoriza- tion to the smart contract. isValid property can also act as soft-deletion when its' value is set to false. There are also some contract methods to support contract functionality. Some contract methods need to only be accessed by certain authorized users for example like admin or superadmin. Admin can

interact with every contract method except contract methods relating to the right of authority while super admin can interact with every contract method like admin including right of authority. Method addUser is used for adding a new account as an user, this method can only be interacted with by admin and super admin because it requires certain regula- tions in adding new users. Method addAdmin is used for adding user accounts as an admin and can only be called by super admin because admin can call crucial smart contract's method that holds user's private data, therefore super admin is needed as the highest role for authorization to control user's application access rights, especially when adding a new admin. Method deactivateUser can be interacted with by admin and super admin for deactivating users and can be called by super admin, method deactivateUser can be used when there is a malicious user in the application. Method deactivateAdmin for deactivating admin and can be called only by super admin,method deactivateAdmin can be used when there is a malicious admin in the application. Method isSuperAdmin for checking superadmin status of a registered user. Method isAdmin for checking admin status of a registered user. Method isUser for checking validation status of a registered user.

LandOwnerShipInspector contract contains ownership history of each land ownership document which is an array of wallet addresses mapping by land ownership document id and land ownership document struct map- ping by string id of document. Land OwnerShip Document struct has properties valid for the document validity, current owner's wallet address, and Land OwnerShip Document data. There are also some methods to support contract functionality. Method findLandOwnerShipDocument to get land certificate document by id. Method findAllLandOwnerShipDocumentByOwner is used to get every land certificate document that is owned by a certain owner's wallet address. Method getLandOwnerShipHistory is used to get ownership history for land certificate and can be called only by the current owner, admin, and superadmin. Method addLandOwnerShip- Document is used for adding new land certificates into the blockchain network and can be called only by admin and super admin to prevent malicious user to accessing this method. Method moveOwnerLandOwnerShipDoc- ument is used for moving ownership of land certificate and can only be called by admin and super admin to prevent malicious user accessing this method. Method editLandOwnerShipDocument is used for editing land certificate can be called only by admin and super admin. Method editLandOwnerShipDocument needs cer- tain regulation logic and only authorized users which are admin and super admin can edit ownership of land certificates. Method removeLandOwnerShipDocument used for removing land certificates. Method remove-LandOwnerShipDocument can be called only by authorized users which are admin and super admin.

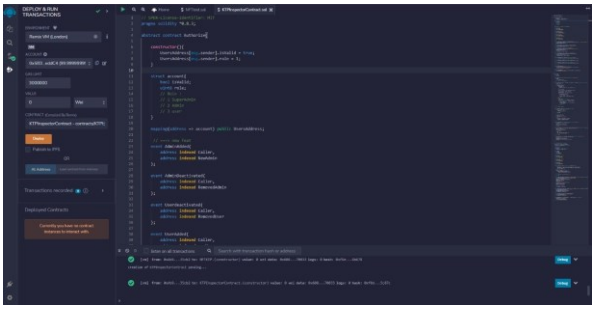

Figure 14: Remix IDE Before Deploying Smart Contract

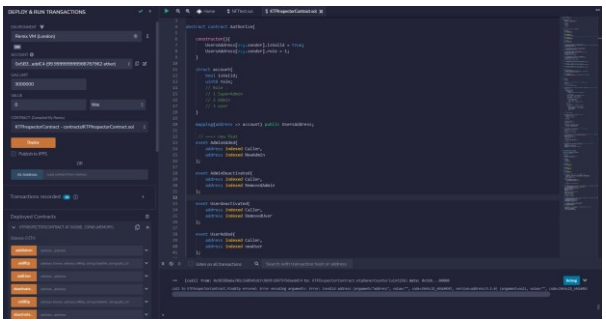

Figure 15: Remix IDE After Smart Contract Deployed

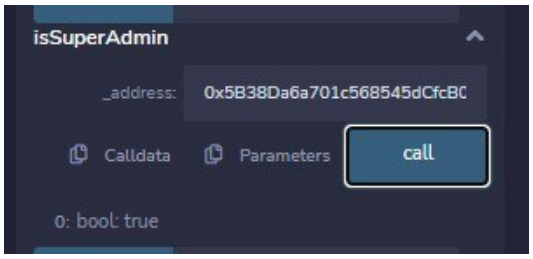

Figure 16: Function for Check Access Rights

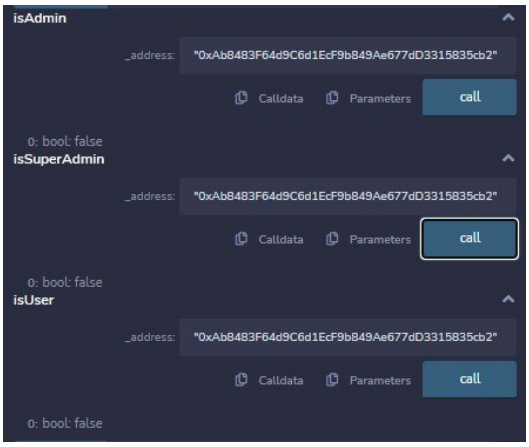

Figure 17: Testing Unregistered Wallet

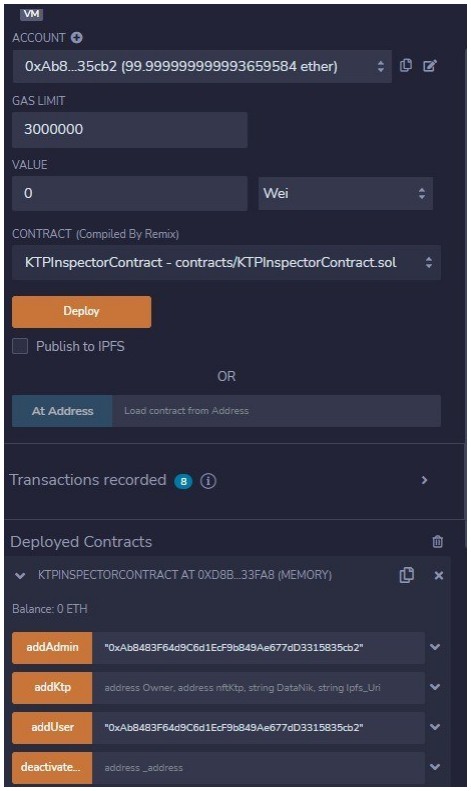

Figure 18: Testing Adding Admin Wallet Access Right with Random Wallet

Figure 19: Reverted Transaction Messages

Smart Contract Testing

Based on figure 14, there's an option for selecting which account to launch the smart contract to blockchain and also the amount of test ethereum. Smart contract testing is necessary because once a smart contract is deployed, it can't be edited because the nature of blockchain itself is immutable. Figure 15 shows the Remix IDE after smart contract deployed where there are many buttons for smart contract interaction.

The smart contract interface in figure 16 is used for testing access rights of the smart contract deployer as a super admin to manage the entire smart contract including access rights of another user, then there's access rights as an admin that can manage KTP data that are stored in the blockchain. Access rights as a user is limited to register it's KTP and see it's KTP once registered. Figure 17 shows the testing process for unregistered wallet addresses so they don't have any access rights to the smart contract.

Figure 18 and figure 19 show the process of testing of adding admin access rights for an account that does not have the rights for it. The contract's transaction is reverted by blockchain because the requirement is not fulfilled.

Figure 20 hows the testing for add new admin, the user input new wallet address to "addAdmin" method with superadmin account. Figure 21 shows the role of the wallet address after calling method "addAdmin" by calling isAdmin method which is now the returned value is true.

Figure 22 shows the testing process to add a new KTP. The parameters for this contract interaction are the contract owner's address wallet, NIK, and encrypted KTP data. Figure 23 shows the testing process to find an existing KTP, where only super admin, admin, and the owner of the KTP itself can access it as shown in figure 24.

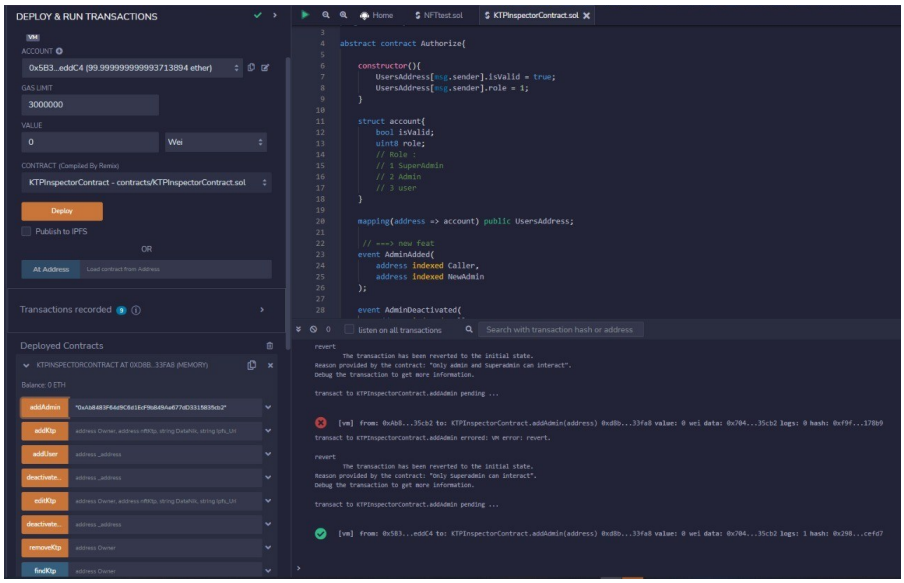

Figure 20: Testing Adding Admin Wallet Access Right with Superadmin

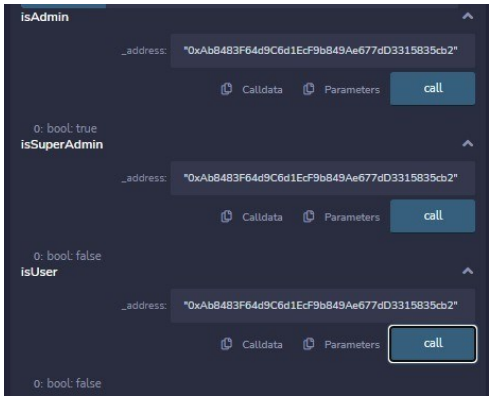

Figure 21: Testing Access Right New Admin

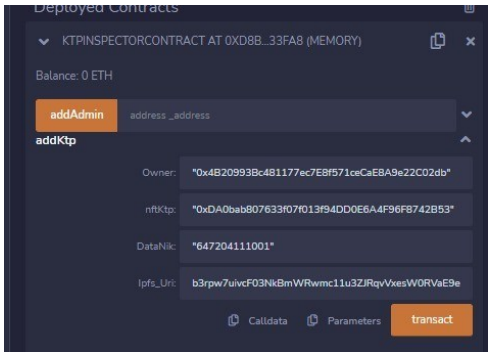

Figure 22: Testing Adding New KTP

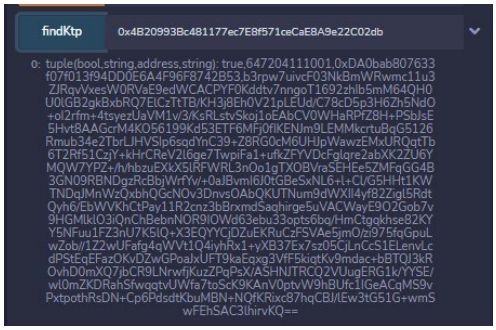

Figure 23: Messages After Contract Transaction

```
function findKtp(address Owner) public view returns(KtpData memory)  {
    require(
        (UsersAddress[msg.sender].role == 1 && UsersAddress[msg.sender].isValid) ||
        (UsersAddress[msg.sender].role == 2 && UsersAddress[msg.sender].isValid) ||
        Owner == msg.sender,
        "You can't access this data"
    );
```

Figure 24: Messages After Contract Transaction

Figure 25: Remix IDE After Deploying Land Certificate Inspector Smart Contract

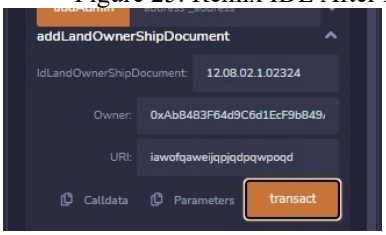

Figure 26: Adding New land certificate

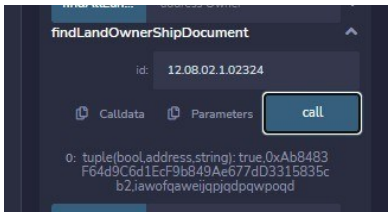

Figure 27: Testing Find New land certificate

Based on figure 25, Remix IDE shows many buttons for contract interaction to test the smart contract. Figure 26 and 27 shows the testing process for adding a new land certificate document. The parameters for this function are id land certificate document, owner address wallet, and NFT URI of the certificate. The land certificate document data will be available in blockchain after the transaction completes. This can be proven by using the function to find a land certificate document by id which will return the land certificate data.\

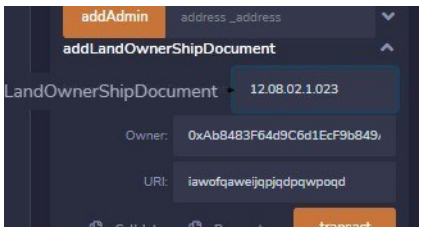

Figure 28: Adding New land Certificate on Same User

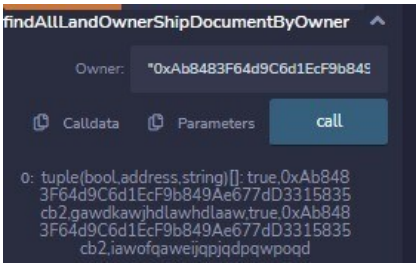

Figure 29: Testing Find Land Certifacate by User

Figure 28 and figure 29 shows users can have multiple land certificate documents. Those certificates are stored in blockchain and the data can be searched by using the find All function by Owner's address wallet after adding the new land certificate with a different id and the same owner.

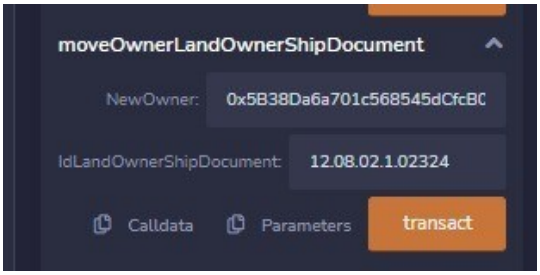

Figure 30: Moving The Ownership of A Land Certificate to Another Account

Figure 30 shows smart contract ability to edit the owner of a land certificate document by moving the ownership. The moveOwnerLandOwnerShipDocument requires new owner address and id of land ownership document.

Figure 31 and figure 32 shows the land certificate document owned by each user after successfully moving the ownership of the land certificate document. Each of the owners now have one land ownership certificate meaning the move ownership of the document is successful.

Figure 33 shows the testing process for removing a land certificate from blockchain by Id, then the land certificate document will not be able to be searched as shown in figure 34 and figure 35.

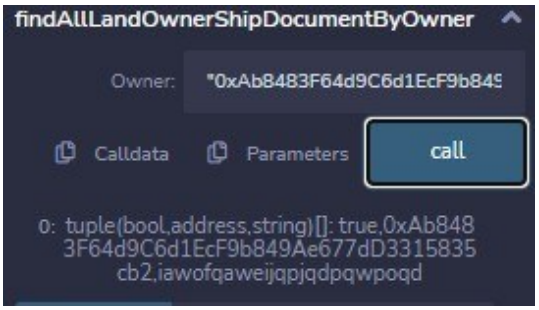

Figure 31: Check Land Certificate Data from Previous User

Figure 32: Check Land Certificate Data from New User

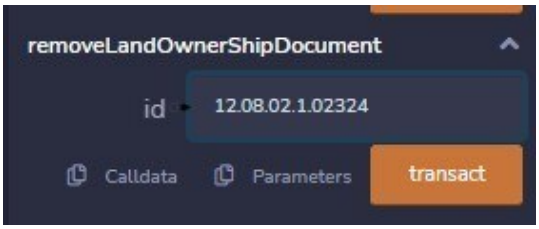

Figure 33: Remove Land Certificate by Id

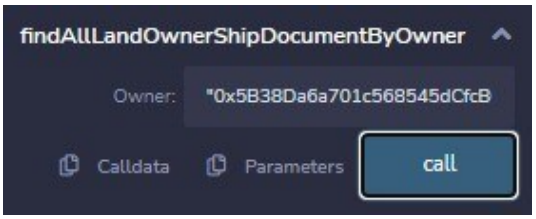

Figure 34: Check Owner of Removed Land Certificate

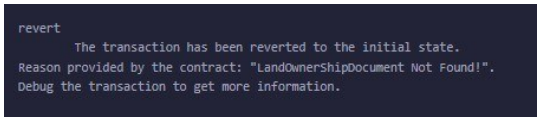

Figure 35: Message After Check Owner of Removed Land Certificate

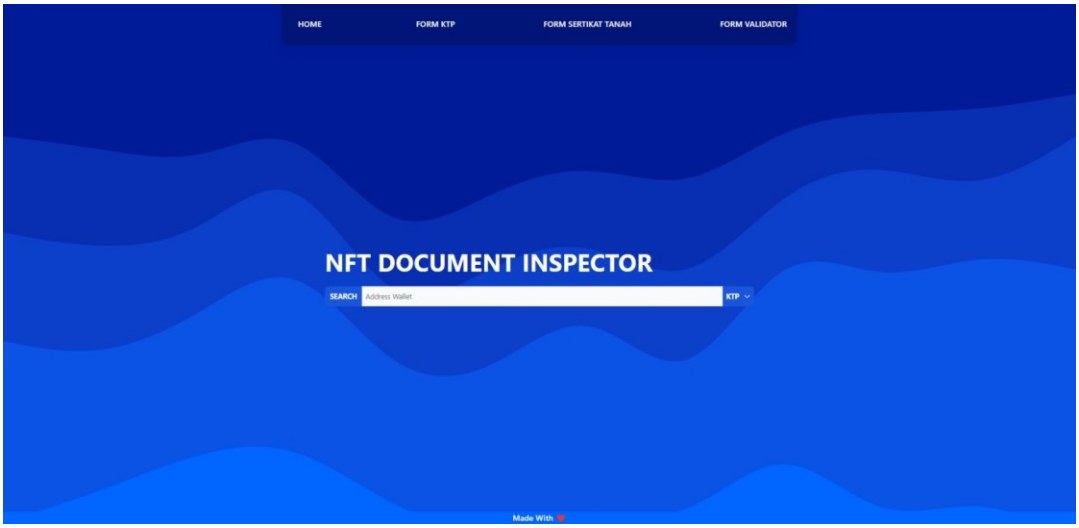

Figure 36: Home page view

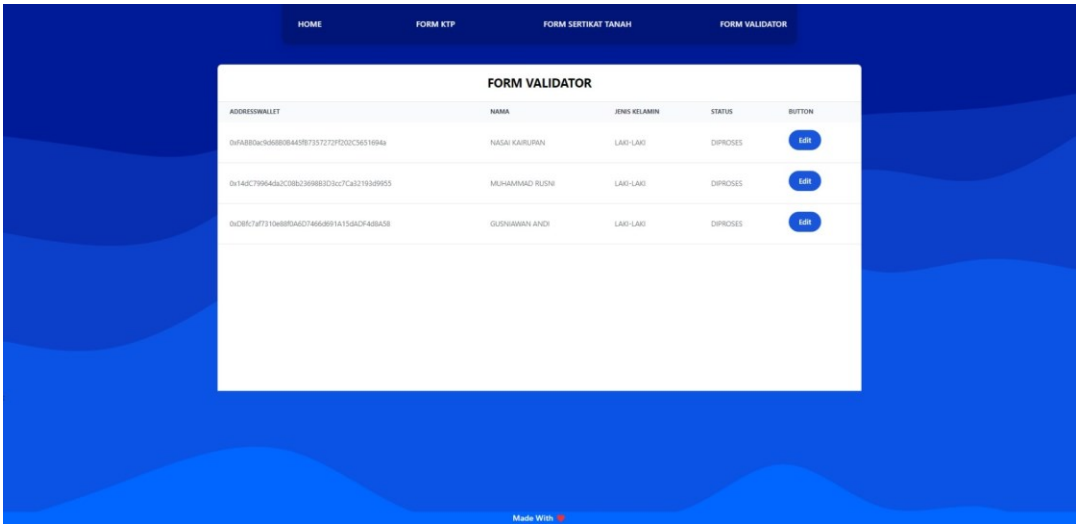

Figure 37: Validator Dashboard View

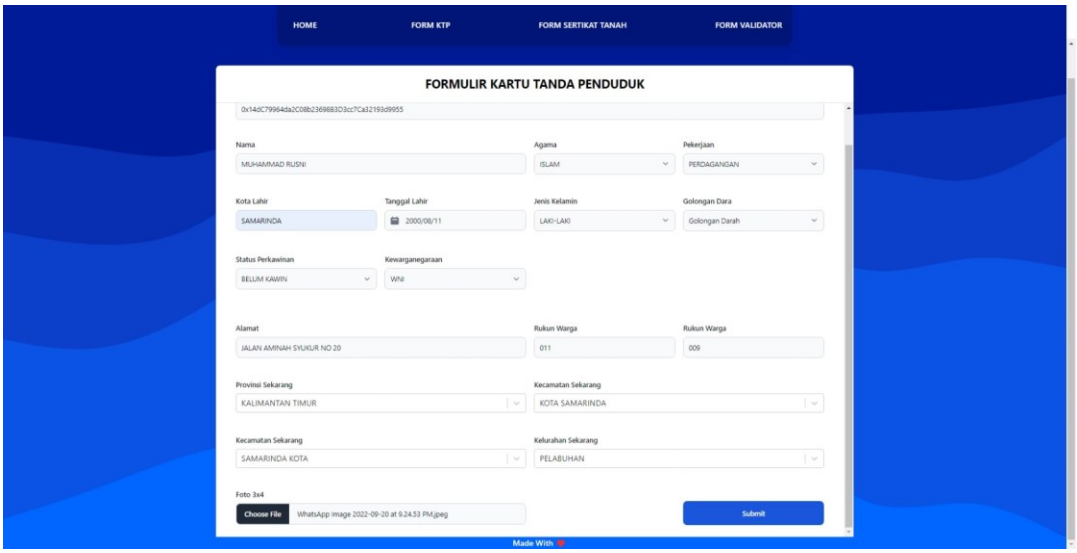

Figure 38: KTP form view

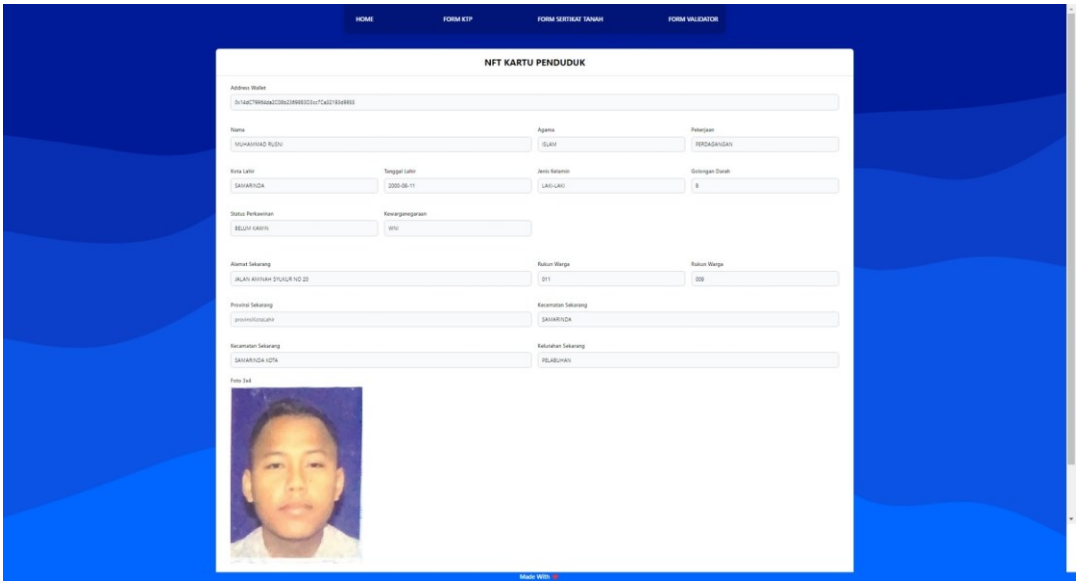

Figure 39: KTP search result

User Interface

Home page contains a navbar, the application title, and a search bar. Users can access KTP forms, land certificate forms, and validator forms via the buttons on the navbar. The search bar is used to search for KTP or land certificates registered in the application and the blockchain network.

Validator dashboard is a page made specifically for admins and super admins from NFT Document Inspector. Admins and super admins can approve or reject applications for KTPs and land certificates from users so that they will or won't be registered into the blockchain network.

KTP form page contains a list of user personal data that must be filled in to create a digital KTP in the form of a Non-Fungible Token (NFT) which will be registered into the blockchain network. The results of approval/rejection of requests from users will be notified by the application.

KTP or land certificates that have been registered in the application will be displayed for the super admin/admin of NFT Document Inspector or the owner of the document when searched based on their wallet address.

Conclusion

Based on the results of research that has been done, it is possible to create system information application of KTP and land certificate as public service with blockchain as a database system. Blockchain provides semi-immutable data storage, hashed data, and log activities that improves data transparency, data security, and data reliability, still it has some drawbacks if it is compared with existing technology used by the government now, which is cloud database system.

Blockchain is a distributed ledger system so every node in a blockchain network is required to have the same data. The more nodes are created, the longer the transaction will take because each node needs to validate new data that will be registered into the blockchain, this makes blockchain storage system difficult to implement in Indonesia, which has a large area, therefore it is required to provide an effective and efficient model storage with existing infrastructure.

It is also very important for Indonesia government to create a private network if it wants to implement blockchain as database to store citizen's important assets, such as KTP and land certificate, because private network can ignore gas concept as fee transaction, which is used in public blockchain network, to increases scalability of the system, especially for storing files and images that are related to government's private assets.

References

[1] Direktorat Jenderal Kekayaan Negara, Pentingnya Standar Pelayanan Publik, Kementerian Keuangan Republik Indonesia, accessed on 16 October 2022, .

[2] Kementerian Komunikasi dan Informatika Republik Indonesia, Presiden Teken UU Adminduk, Kini Pelayanan KTP, KK, dan Akta Kelahiran Semua Gratis, Kementerian Komunikasi dan Informatika Republik Indonesia, accesed in 16 October 2022,.

[3] Kementerian Komunikasi dan Informatika Republik Indonesia, Program PTSL Pastikan Penyelesaian Sertifikasi Lahan Akan Sesuai Target, Kementerian Komunikasi dan Informatika Republik Indonesia, accessed on 16 October 2022, .

[4] Berawi, M.A., Sari, M., Addiani, A.F. Madyaningrum, N. 2021, 'International Journal of Technology', Developing a Blockchain-based Data Storage System Model to Improve Government Agencies' Organizational Performance, vol. 12, no. 5, pp. 1038-1047.

[5] CNBC Indonesia, Heboh! Data KTP Hingga Nomor HP 279 Juta Warga RI Bocor?, CNBC Indonesia, accessed on 16 October 2022, .

[6] Komisi II Dewan Perwakilan Rakyat Republik Indonesia, Kementerian ATR BPN Diminta Selesaikan Sengketa Lahan, Dewan Perwakilan Rakyat Republik Indonesia, accessed on 16 October 2022, .

[7] Cai, C.W., 2018. 'Accounting and Finance', Disruption of Financial Intermediation by FinTech: A Review on Crowdfunding and Blockchain, vol. 58, no. 4, pp. 965–992.

[8] Szabo, N. 1996, 'Extropy Journal of Transhuman Thought', Smart Contracts: Building Blocks for Digital Free Markets, accessed in 18 October 2022, .

[9] Berawi, M.A., Radjilun, M.K.Z., Sari, M., 2020, 'Innovations in Digital Economy - 2nd International Scientific Conference, SPBPU IDE 2020, Revised Selected Papers', Developing Blockchain-Based Crowdfunding Model for Property Investment, vol. 1445, pp. 23-39, DOI:10.1007/978-3-030-84845-32.

[10] Stephen, R. Alex, A. 2018, 'IOP Conference Series: Materials Science and Engineering', A Review on BlockChain Security, advance online publication, DOI:10.1088/1757-899X/396/1/012030.

[11] Ølnes, S., Ubacht, J., Janssen, M., 2017. 'Government Information Quarterly', Blockchain in Government: Benefits and Implications of Distributed Ledger Technology for Information Sharing, vol. 34, no. 3, pp. 355–364.

[12] Alketbi, A., Nasir, Q., Talib, M.A., 2018, '2018 15th Learning and Technology Conference (LT)', Blockchain for Government Services-Use Cases, Security Benefits and Challenges, pp. 112–119, DOI: 10.1109/LT.2018.8368494.

[13]Razzaq, A., Khan, M.M., Talib, R., Butt, A.D., Hanif, N., Afzal, S., Raouf, M.R., 2019. 'International Journal of Advanced Computer Science and Applications', Use of Blockchain in Governance: A Systematic Literature Review, vol. 10 no. 5, pp. 685–691.

[14]Kurniawan, I.A., Yusman, D., Aprilia, I.O., 2021,'Aptisi Transactions on Management (ATM)', Utilization of Blockchain Technology Revolution in Electronic ID Card Data Integrity, vol. 5, no. 21, pp. 137-142.

[15]Christine, H., Novelianto, K.T., Restiawati, M., Jayanti, H.Y., Afriyadi, 2022, 'Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi dan Komunikasi ', A Study of Permissioned Blockchain-Based Framework for Land Ownership Tracking in Indonesia, vol. 3, no. 2, pp. 119–126.

[16]International Business Machines (IBM), Smart contracts defined, What are smart contracts on blockchain?, accessed on 18 October 2022, .

[17]Musamih, A., Salah, K., Jayaraman, R., Yaqoob, I., Puthal, D., Ellahham, S. 2022, 'IEEE Consumer Electronics Magazine', NFTs in healthcare: Vision, opportunities, and challenges, DOI: 10.1109/MCE.2022.319648

[18] Ethereum, Summary, What is Ethereum?, accessed on 18 October 2022, .