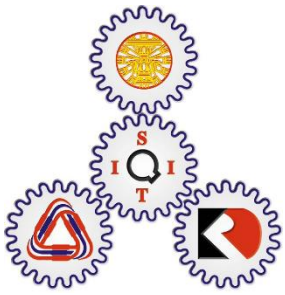

Capstone Project
Title Deed Application using Blockchain

Presented by

Methika	Aungkurboribhun	6322774462
Arisoon	Dtangpaibool	6322775015
Araya	Yawuth	6322775031

Lecturer: Dr. Watthanasak Jeamwatthanachai

Blockchain Development - CSS484 Section 1

Sirindhorn International Institute of Technology
Thammasat University

Semester 1/2023

18 December 2023

Table of Contents

Contents

Table of Figures

Chapter 1

Introduction

Problem Statement

In real estate, the secure and transparent management of property titles stands as a cornerstone for the integrity of transactions and the protection of property rights. Traditional methods of storing title deeds often involve paperwork, centralized databases, and the risk of fraud or manipulation leading to a lack of trust in the authenticity of property documents.

However, there's blockchain technology to help deal with those challenges. With its core features, decentralization, immutability, and transparency, make it an alternative way instead of traditional title deed management systems. We response to those challenges by the integration of blockchain technology, a decentralized and tamper-proof system for title deed storage.

Objectives

1. To create a decentralized application of title deed system.
2. To utilize blockchain in the title deed system with transparency and traceability.
3. To improve accessibility to the title deed storage.

Stakeholder Benefits

Transparency and Visibility

Users can view the title deed and its details. This ensures that all transactions related to property titles are visible, traceable, and verifiable, instilling confidence in the accuracy and authenticity of the recorded information.

Traceability

When a user acquire a property, you can trust the legitimacy of the transfer, as the blockchain records every change in ownership with precise details, leaving no room for ambiguity.

Accessibility

Users can access this data from anywhere, anytime, without needing to visit office. And ensuring that everyone, regardless of their technical or financial background, has equal access to title deed information.

Chapter 2

Related Documents

Blockchain

Blockchain is decentralized and immutable, which makes it difficult for unauthorized individuals to manipulate or edit the data, by continuously expanding the list of records that are kept in a container called “block” that is linked and secured by using cryptography as a “chain”. It offers enhanced security and encourages transparency because everyone using the blockchain network has access to the same data, eliminating the need for a middleman. Moreover, blockchain makes it valuable across multiple industries. Which detail each component as blockchain serves as a shared and immutable ledger, streamlining transaction recording and asset tracking in business networks. Whether tangible assets like real estate, vehicles, or cash, or intangible assets such as intellectual property, patents, copyrights, or branding, virtually anything of value can be monitored and traded on a blockchain network. This not only mitigates risks but also reduces costs for all parties involved.

Blockchain

A block can be described as a containing record or data, commonly accompanied by metadata like a block number, timestamp, digital signature, hash value, and nonce, depending on the purpose of the blockchain system designer. Each block is linked by including additional metadata called the previous hash value that points to the previous block

Cryptographic Hashing

Cryptographic hashing is a computational method that generates a unique code or fingerprint for any input information. A key attribute of a hash is its ability to consistently produce a unique code, even with a single word or bit of alteration to the input. As previously mentioned, each block is required to store its hash value along with the previous hash value of the preceding block. If a block within the chain is tampered with, the connected block will be flagged as invalid due to a mismatch in the previous hash value, and will make the rest of the chain unstable. Consequently, cryptographic hashing poses a challenge to altering the data of a single block without entirely changing the chain, contributing to the fundamental characteristic of a blockchain as “immutability”.

Distributed Network

A distributed network of computers is for maintaining the integrity of the blockchain through a peer-to-peer system. Each node possesses a complete copy of the blockchain and consistently updates itself when new blocks are added. This continuous synchronization significantly bolsters the immutability of the blockchain. Even in scenarios where malicious actors alter an entire chain, their copies of the blockchain data. This architectural design ensures the resilience of the blockchain against errors and malicious attacks. Additionally, every node within the network serves as a validator, often referred to as a “miner” in cryptocurrency terms, contributing to the logical addition of new blocks through consensus. This characteristic underlines another crucial aspect of a blockchain known as “decentralization”.

Consensus Mechanism

Before a new block is incorporated into the blockchain, participating nodes must reach a consensus on the accuracy of the information it contains. This agreement is achieved through a consensus mechanism, the specifics of which vary based on the implementation. The consensus mechanism ensures that the newly added block becomes the singular and universally accepted version of the truth acknowledged by all nodes within the network.

Proof of Work (PoW)

Nodes, often referred to as miners, engage in competition using their valuable computational resources to solve a complex mathematical puzzle. The miner who successfully solves the puzzle earns the privilege to authenticate the new block and incorporate it into the blockchain, subsequently receiving a reward. This approach, while incentivizing participation, is energy-intensive and has the potential to lead to centralization. Blockchains such as Bitcoin, Dash, Litecoin, and Monero operate under this model.

Proof of Stake (PoS)

In order to participate in the validation process, nodes are required to lock up or "stake" a specific amount of their assets, typically a widely used cryptocurrency. The extent of the stake influences the probability of being chosen to validate a block and receive a reward. Once selected as a validator, the node enters into a contract with the network, often implemented through a smart contract, wherein it agrees that in the event of the discovered fraudulent or malicious nature of the block, all staked assets will be forfeited. This staking mechanism promotes decentralization and is more environmentally sustainable compared to proof of work. Examples of blockchains utilizing this approach include Ethereum, Cardano, Polkadot, and Solana.

Proof of Authority (PoA)

In contrast to proof of work, there is no technical competition among nodes or miners in this model. Validators are chosen based on their trustworthiness or reputation rather than engaging in a competitive process. This approach can be seen as a variation of proof of stake, where nodes stake their "reputation" instead of a cryptocurrency asset. The automated nature of the process eliminates the need for validators to continually monitor their computers, distinguishing it from proof of work and proof of stake. The proof of authority model relies on a limited number of block validators, resulting in a highly scalable system. Blockchains such as VeChain, Bitgert, and Xodex operate under this model.

Smart Contract

Smart contracts offer the capability to deploy various programs on a blockchain, extending beyond data storage. These contracts are integrated into blocks and automatically executed once specified conditions are fulfilled. The implementation of smart contracts enhances efficiency, security, and transparency, enabling parties to directly fulfill agreements. This innovative tool proves effective in reshaping numerous industries and traditional contract management practices. With the ability to automate and streamline processes by executing actions based on predefined conditions, smart contracts find applications across various industries.

Decentralized Application

Some blockchain platforms, like VeChain and Ethereum, can support applications built on top of blockchain platforms and smart contracts running across a distributed system called decentralized applications (DApps), which make blockchain able to offer cutting-edge features and services. DApps have also been developed to facilitate secure, blockchain-based voting and governance. DApps can also be integrated into web browsers to function as plugins that can help serve ads, track user behavior, and solicit crypto donations. There are many examples of commercial decentralized applications deployed in various real-world use cases.

HTML and CSS

HTML

HTML or Hypertext Markup Language, and CSS, or Cascading Style Sheets, are two of the core technologies for building web page interfaces. While HTML provides the structure of the page, CSS provides visual and aural layout along with graphics and interactions. HTML gives authors the means to:

- Publish online documents with proper structure, including headings, text, tables, lists, photos, etc.
- Retrieve online information such as tracking details and product details via hypertext links at the click of a button or component in the web page.
- Design forms for conducting transactions with remote services, such as searching for information, making reservations, registering to the system, ordering products, sending products, etc.
- Include spreadsheets, video clips, sound clips, and other applications directly in their documents.

CSS

CSS is a programming language used to define how web pages should look and be presented, encompassing elements like colors, fonts, and layout. It enables customization for various devices, such as screens of different sizes or printers. CSS operates independently from HTML. By separating HTML and CSS, websites become easier to manage, as style sheets can be shared across multiple pages and pages can be adapted to different environments more efficiently.

Bootstrap

Bootstrap is an open-source front-end development framework. It is designed for crafting websites and web applications. It simplifies web development by offering a 12-column grid system and a collection of syntax for template designs, built on HTML, CSS, and JavaScript. It is designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template, so developer only need to insert the into pre-define grid system.

Ganache

Ganache is a blockchain application designed for Ethereum development and testing purposes. It functions as a local Ethereum network running on a developer's machine, allowing simulation of Ethereum behavior without real network connections or spending actual Ether. Ganache provides predefined test accounts with test Ether, enabling developers to iterate and debug smart contracts and decentralized applications (DApps) efficiently. With features like on- demand block mining, instant transaction confirmation, and gas price control, Ganache offers control over the blockchain environment during development and testing. Its user-friendly interface and developer-friendly APIs make it a popular choice for Ethereum developers to locally develop, unit test, and debug smart contracts and DApps before deployment to the live Ethereum network.

Chapter 3

Design and Implementation

Key Requirements

User Role Separation

We designed to separate the users of the application into 2 groups with different use cases: Regular user and Staff user.

Regular User

- No need to authenticate.
- Enter the sheet number to view the title deed.

Staff User

- Authenticate themselves before accessing the system (Using Metamask).
- Able to add new, edit, and other title deed functions.

Title deed functionality

Title deed functionality in the context of blockchain technology typically involves digitizing and securing property records. This includes creating a decentralized and immutable ledger for recording property ownership and transaction history. Blockchain enhances security and transparency in property transactions, reduces fraud, and streamlines the process of transferring and verifying property ownership. This technology also allows for easier access to title information, making it more efficient and user-friendly compared to traditional paper-based systems.

Adding deed

The group of requirements show the staff function to add the new deed to the platform, there are two requirements that the prototype must satisfy:

1. The platform should allow any staff to create a new deed and add it to the platform. It should be done when the staff logs in to the platform.
2. The created deed should have a unique title generated by the platform.

Editing deed

The platform should allow any staff to edit the deed and save both new information and staff log in the platform. It should be done when the staff logs in to the platform.

Viewing deed

A requirement on viewing output is given to let the object representing the title deed detail. Therefore, the staff user can see the details of each title deed and logs. The regular user can only view the deed title with the matching string title only.

User Requirements

Requirement Table

User ID	Role	Requirement
UID001	Regular User	Get the title deed history of editing.
		I want to see my deeds, others, and the history of each deed.
		I want to request for separate/merge the deed.
		I would like to edit my information in the deed.
UID002	Staff User	Can create new deed, cancel deed.
		Can edit document inside the deed.
		Add authorize officer, revoke officer.

System Requirements

Non-functional Requirements

Usability Requirements

1. The system should be compatibility with a range of devices including desktop computers, laptops, tablets, and smartphones.
2. The system should be a web-based application.

Security Requirements

1. A robust blockchain platform that supports smart contracts.
2. Secure and efficient consensus mechanism in Blockchain.

Functional Requirements

1. **Retrieval Information Requirements**
    a. Regular User: the user should know the valid sheet number to find the deed in the system.
2. **Register Information Requirements**

    a. Staff User: the staff should know the owner detail, land detail, and land address to create a new deed in the system.

3. Authentication Requirements

a. Staff User: The staff should have metamask account to be able to authenticate himself.

Use case Diagram

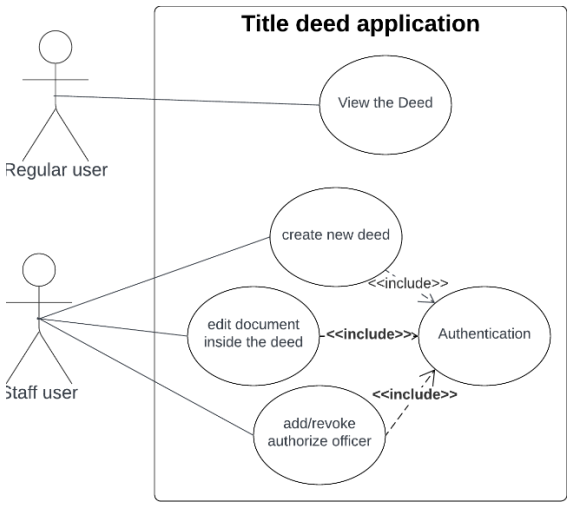

Figure 1. Use case diagram of title deed finder.

Blockchain Network Requirements

The blockchain network should be permissionless so that any nodes can connect and disconnect from the system because the product tracking information is visible to any customers. Moreover, they should not charge a gas fee for retrieving the title information. In cases of consensus mechanisms, Proof of Authority (PoA) should be implemented because of its speed of execution, performance, transparency, and security.

Software Implementation and deployment

Title deed and User Information

Global Variables

1. Hold history
    a. Holder national ID: string
    b. Holder hold time: string
    c. Parcel number: string
    d. Holder name: string
2. Parcel history
    a. Parcel number: string
    b. Land size: string
    c. Create time: string
    d. Coordinates: string

e. Land address: string
f. Photo of land

3. Metadata
    a. Sheet number: string
    b. Create time: string
    c. Assigner address: string
    d. Status: string
4. Last record
    a. Holder national ID: string
    b. Holder hold time: string
    c. Parcel number: string
    d. Holder name: string
    e. Land size: string
    f. Assign time: string
    g. Coordinates: string
    h. Land address: string
    i. Photo of land: string

Front-end Design

Webpage Design

Home Page

Regular user, they can search the deed by insert the deed number in the box

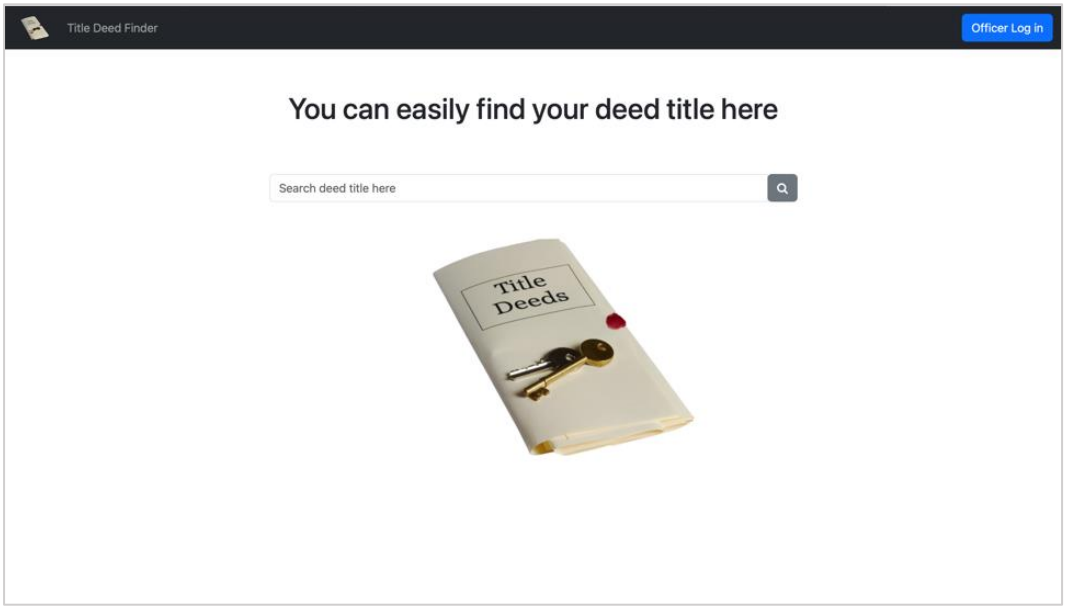

Figure 2: The home page.

Staff user, Homepage Enters with the Title Deed list that they able to manage.

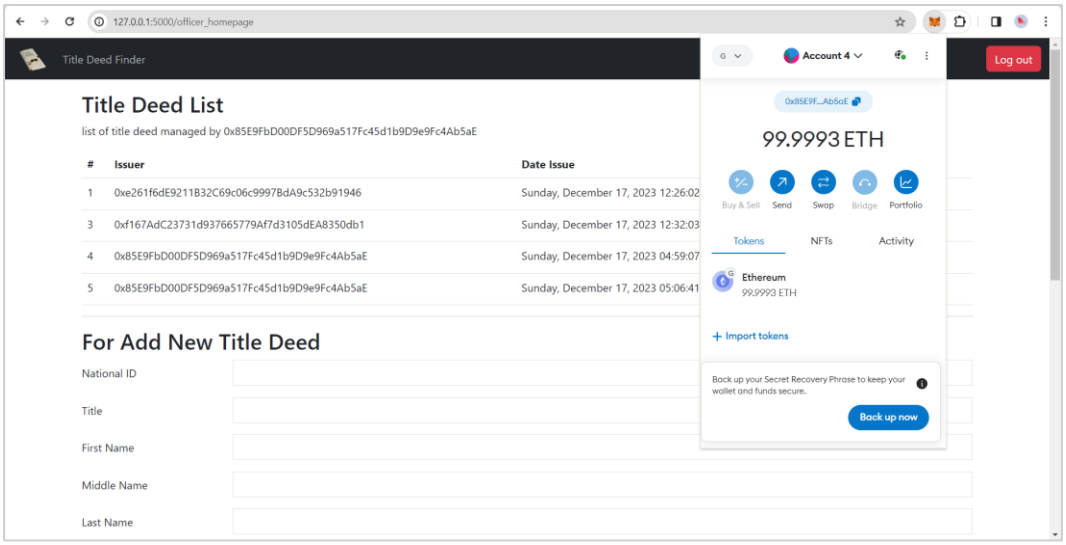

Figure 3: The staff user logs in by Metamask.

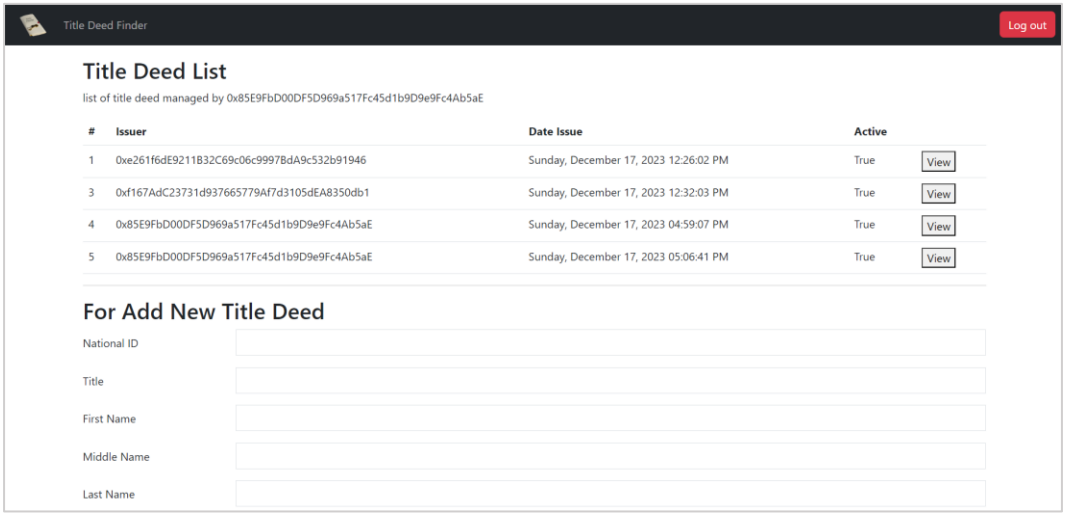

Figure 4: The staff home page.

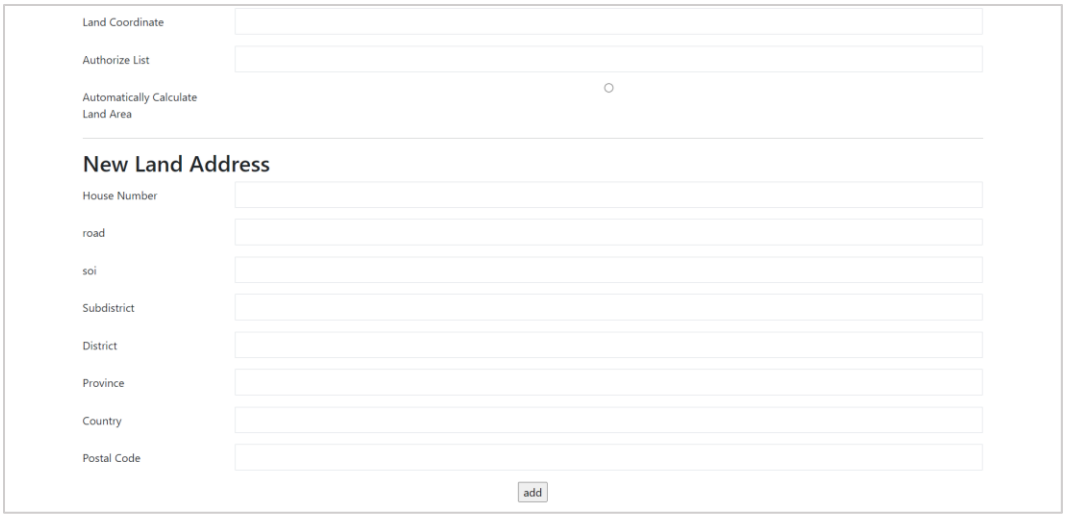

Figure 5: The staff home page (continue).

Title Deed Details Page

When the user enters the correct/available deed number. The web will be routed to show details page.

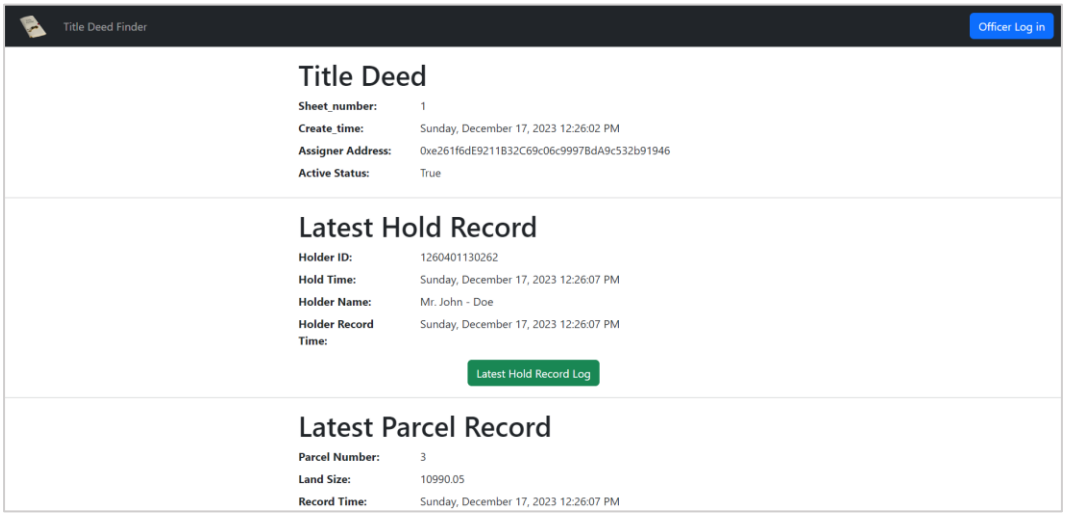

Figure 6: The deed page.

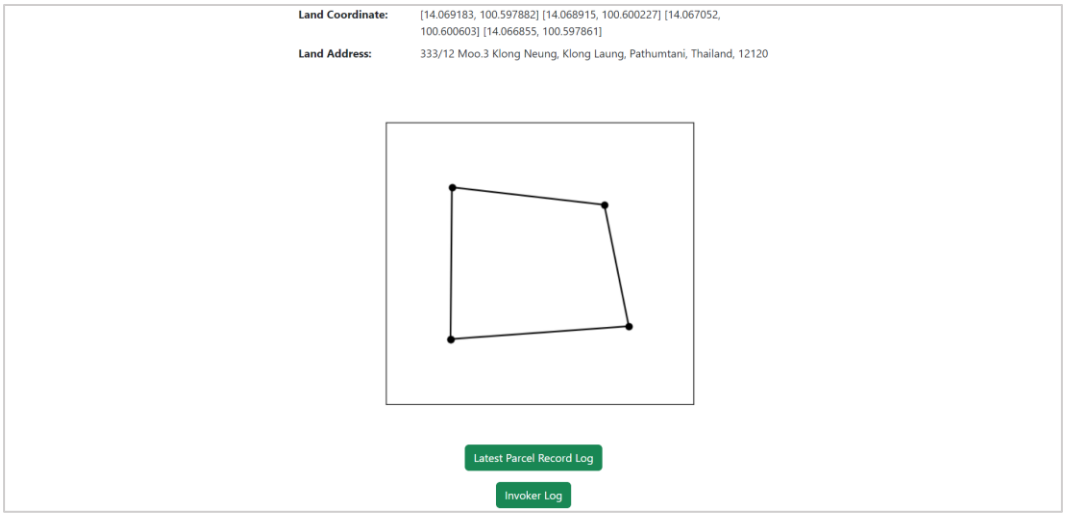

Figure 7: The regular user deed page (continue).

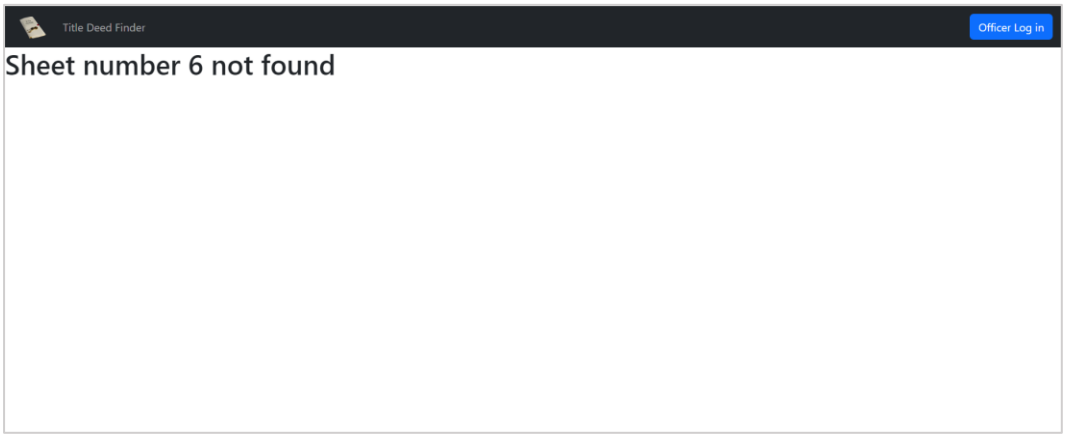

Figure 8: the unavailable deed number is inserted.

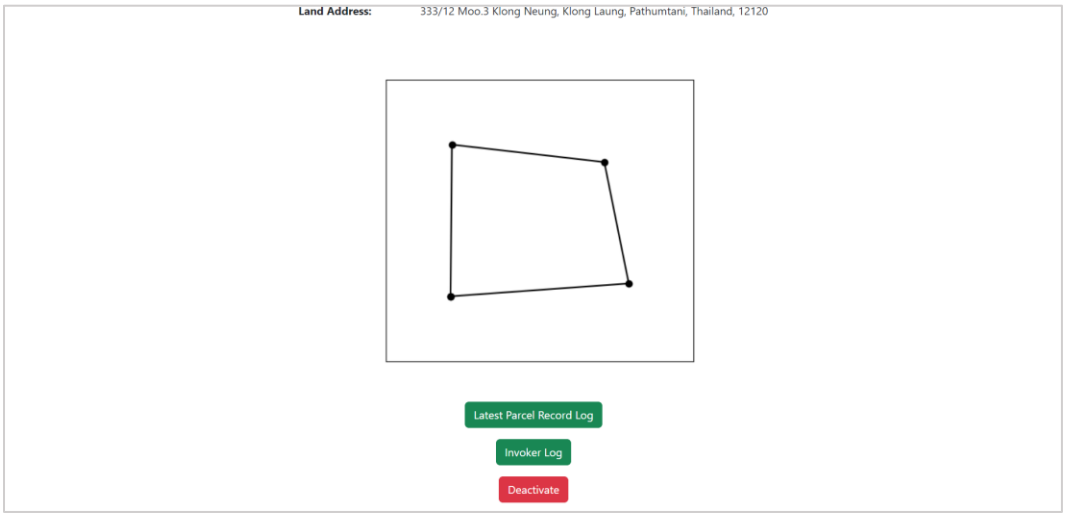

Figure 9: the staff user deed page (add Deactivate/Reactivate button).

When clicking the Latest Hold Record log button, it will route to Latest Hold Record log for regular user. For staff user will have the option for transferring deed title.

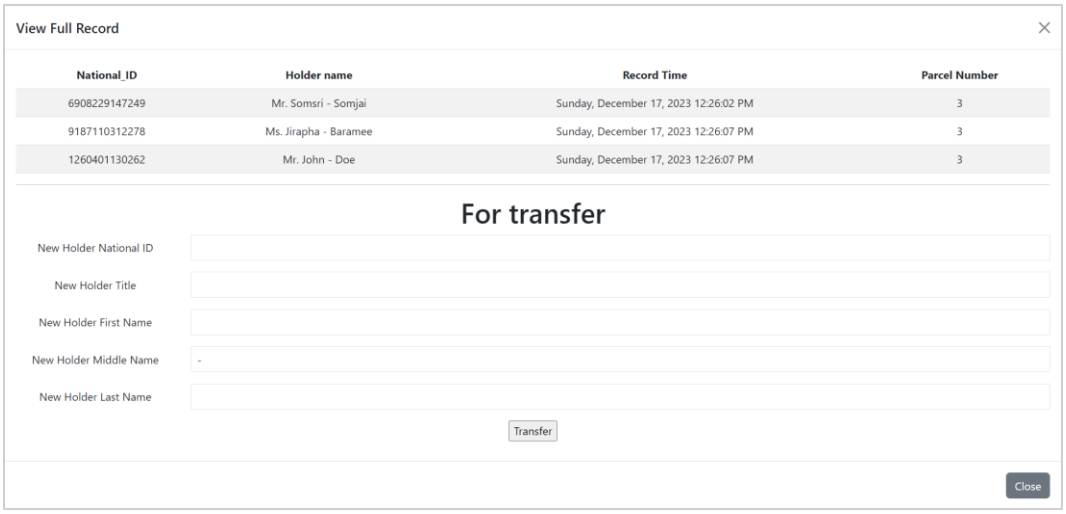

Figure 10: the staff user transfer page.

When clicking the Latest Parcel Record log button, it will route to Latest Parcel Record log for regular user. For staff user will have the option for Modifying Parcel.

Figure 11: the staff user modifies parcel page.

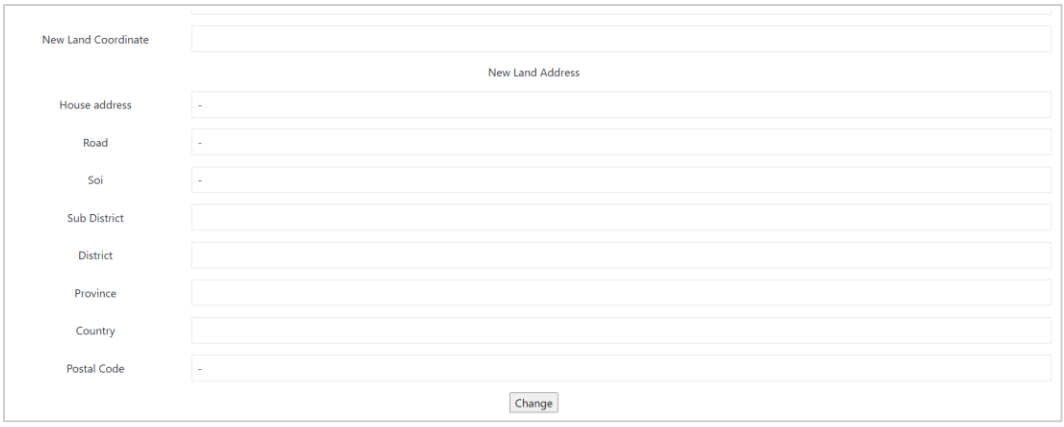

Figure 12: the staff user modifies parcel page.

Land Shape Coordinate design

- function is_coordinate_valid which has sub function is_in_thailand
- function reorder_coordinate
- function calculate_area
- function land_image associate the figure with Matplotlib and FigureCanvasAgg

```
def calculate_area(lat_lon_list):
    lat_lon_list = reorder_coordinate(lat_lon_list)
    if(lat_lon_list[0] != lat_lon_list[-1]):
        lat_lon_list.append(lat_lon_list[0])

    obj = {'type':'Polygon','coordinates':[lat_lon_list]}
    return area(obj)
```

Figure 13: the calculate area function.

Smart Contract Functional design

The smart contract functional is to display the land shape by generated by the provided coordination, provide essential details for a title deed, allow multiple officers to manage a single title deed, ACL system, and permit modifications of key parameters such as holder during transfer and parcel details like address, size, and coordinates, as needed.

Select contract:

```python
def select_contract(sheet_number) -> dict:

    global smart_contract, global_sheet_number
    contract_address = get_contract_by_num(sheet_number)
    if(contract_address["status"] <= 0):
        return {"status":0,"output":f"Sheet number {sheet_number} does not exist"}

    contract_address = contract_address["output"]

    try:
        with open(os.path.join(current_path,f"./abi/sheet_number{sheet_number}_abi.json"),"r") as f:
            contract_abi = f.read()

    except:
        return {"status":0,"output":f"cannot find ABI"}

    try:
        smart_contract = web3.eth.contract(address=contract_address,abi=contract_abi)

    except:
        return {"status":0,"output":"Either contract address or blockchain address is invalid"}

    global_sheet_number = sheet_number
    return {"status":1,"output":f"select contract address {contract_address} successfully."}
```

Figure 14: Select contract code.

Cancel document:

```python
def cancel_document(caller_address):
    global smart_contract, global_sheet_number

    if(not web3.is_address(caller_address)):
        return {"status":0,"output":f"address {caller_address} is invalid."}

    if(len(web3.eth.get_code(caller_address)) > 2):
        return {"status":0,"output":f"address {caller_address} is not account address."}

    try:
        result = smart_contract.functions.cancel_document().transact({"from":caller_address})
        return {"status":1,"output":f"cancel sheet {global_sheet_number} successfully"}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 15: Cancel document code.

Reactivate:

```
def reactivate(caller_address):
    global smart_contract, global_sheet_number

    if(not web3.is_address(caller_address)):
        return {"status":0,"output":f"address {caller_address} is invalid."}

    if(len(web3.eth.get_code(caller_address)) > 2):
        return {"status":0,"output":f"address {caller_address} is not account address."}

    try:
        result = smart_contract.functions.reactivate().transact({"from":caller_address})
        return {"status":1,"output":f"reactivate sheet {global_sheet_number} successfully"}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 16: Reactivate the deed code.

Change region:

```
def change_region(caller_address,land_size,coordinates,land_address):
    if(not web3.is_address(caller_address)):
        return {"status":0,"output":f"address {caller_address} is invalid."}

    if(len(web3.eth.get_code(caller_address)) > 2):
        return {"status":0,"output":f"address {caller_address} is not account address."}

    if(land_size <= 0):
        return {"status":0,"output":f"land size {land_size} is invalid."}

    for c in coordinates:
        if(not util_lib.is_coordinate_valid(c)):
            return {"status":0,"output":f"coordinates {c} is invalid"}

    #if(not util_lib.is_valid_address(land_address)):
    #    return {"status":0,"output":f"land address {land_address} is invalid."}

    try:
        result = smart_contract.functions.cha (parameter) caller_address: Any rdinates)
        ,str(land_address)).transact({"from":caller_address})
        return {"status":1,"output":f"modify region sheet {global_sheet_number} successfully"}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 17: Change the region in the deed code.

Transfer:

```
def transfer(caller_address, national_ID, holder_name):

    global smart_contract, global_sheet_number

    if(not web3.is_address(caller_address)):
        return {"status":0,"output":f"address {caller_address} is invalid."}

    if(len(web3.eth.get_code(caller_address)) > 2):
        return {"status":0,"output":f"address {caller_address} is not account address."}

    if(not util_lib.is_valid_thai_id(national_ID)):
        return {"status":0,"output":f"National ID {national_ID} is invalid."}

    if(not util_lib.is_valid_holder_name(holder_name)):
        return {"status":0,"output":f"Holder name {holder_name} is invalid"}

    try:
        result = smart_contract.functions.transfer(national_ID," ".join(holder_name)).transact({"from":caller_address})
        return {"status":1,"output":f"transfer ownership of sheet {global_sheet_number} successfully"}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 18: Transfer the deed code.

Get log:

```
def get_log():
    global smart_contract
    output = []
    result = smart_contract.functions.get_log().call()
    for log in result:
        event_time = util_lib.timestamp_to_date_string(log[0])
        output.append([event_time, log[1], log[2]])
    return {"status":1,"output":output}
```

Figure 19: Get the log code.

Get hold history:

```
def get_hold_history(show_ID=False):
    global smart_contract
    keys = ["holder_national_ID","holder_hold_time","parcel_number","holder_name"]
    result = smart_contract.functions.get_hold_history().call()
    for i in range(len(result)):
        result[i] = list(result[i])
        result[i][1] = util_lib.timestamp_to_date_string(result[i][1])
        result[i] = dict(map(lambda i,j : (i,j) , keys,result[i]))

        if(show_ID == False):
            holder_ID = result[i]["holder_national_ID"]
            holder_ID = str(holder_ID)
            new_ID = ""
            for j in range(len(holder_ID)):
                if(j in [7,8,9,10,11,12]):
                    new_ID += "X"
                else:
                    new_ID += holder_ID[j]

            result[i]["holder_national_ID"] = new_ID
    return {"status":1,"output":result}
```

Figure 20: Get hold history code.

Get parcel history:

```
def get_parcel_history():
    global smart_contract
    result = smart_contract.functions.get_parcel_history().call()

    keys = ["parcel_number","land_size","create_time","coordinates","land_address"]
    for i in range(len(result)):
        result[i] = list(result[i])
        result[i][2] = util_lib.timestamp_to_date_string(result[i][2])
        result[i][1] = float(result[i][1])
        result[i][3] = eval(result[i][3])

        coor = ""
        for j in result[i][3]:
            coor += (str(j)+" ")

        result[i][3] = coor

        result[i] = dict(map(lambda i,j : (i,j) , keys,result[i]))

    return {"status":1,"output":result}
```

Figure 21: Get the parcel history code.

Get metadata:

```
def get_metadata():
    global smart_contract
    result = smart_contract.functions.get_metadata().call()
    keys = ["sheet_number","create_time","assigner_address","active"]
    result = dict(map(lambda i,j : (i,j) , keys,result))
    result["formatted_create_time"] = util_lib.timestamp_to_date_string(result["create_time"])
    return {"status":1,"output":result}
```

Figure 22: Get the metadata code.

Get last record:

```
def get_last_record():
    global smart_contract
    result = smart_contract.functions.get_last_record().call()
    keys = ["holder_national_ID","holder_hold_time","parcel_number","holder_name","land_size","assign_time"
    ,"coordinates","land_address"]
    result = dict(map(lambda i,j : (i,j) , keys,result))
    result["coordinates"] = eval(result["coordinates"])
    result["land_address"] = result["land_address"].split()
    result["holder_hold_time"] = util_lib.timestamp_to_date_string(result["holder_hold_time"])
    result["land_size"] = float(result["land_size"])
    result["assign_time"] = util_lib.timestamp_to_date_string(result["assign_time"])
    if(result["coordinates"][0] == result["coordinates"][-1]):
        del result["coordinates"][-1]

    return {"status":1,"output":result}
```

Figure 23: get last record code.

Authorize:

```
def authorize(caller_address:str,user_address:str) -> dict:
    global smart_contract
    if(caller_address not in web3.eth.accounts):
        return {"status":0,"output":f"Address {caller_address} does not exist in blockchain"}

    if(user_address not in web3.eth.accounts):
        return {"status":0,"output":f"Address {user_address} does not exist in blockchain"}

    try:
        smart_contract.functions.authorize(user_address).transact({"from":caller_address})
        return {"status":1,"output":f"Authorized {user_address} successfully."}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 24: Authorize the staff code.

Revoke:

```
def revoke(caller_address:str,user_address:str) -> dict:
    global smart_contract
    if(caller_address not in web3.eth.accounts):
        return {"status":0,"output":f"Address {caller_address} does not exist in blockchain"}

    if(user_address not in web3.eth.accounts):
        return {"status":0,"output":f"Address {user_address} does not exist in blockchain"}

    try:
        smart_contract.functions.revoke(user_address).transact({"from":caller_address})
        return {"status":1,"output":f"Revoked {user_address} successfully."}

    except Exception as e:
        return {"status":0,"output":e}
```

Figure 25: Revoke code.

Check Authorize:

```
def check_authorize(caller_address:str) -> dict:
    global smart_contract
    if(caller_address not in web3.eth.accounts):
        return {"status":0,"output":f"Address {caller_address} does not exist in blockchain"}

    result = smart_contract.functions.check_authorize(caller_address).call()
    if(result):
        return {"status":1,"output":f"{caller_address} have permission"}

    else:
        return {"status":0,"output":f"{caller_address} don't have permission"}
```

Figure 26: check authorized code.

System Deployment Design

Host Device

- 24 GB Memory
- CPU 3.5 GHz
- SSD 512GB
- 4 Cores CPU
- Windows 11 OS

Blockchain Network Service on Host (Ganache)

- 10 Nodes / Accounts
- Ethereum Merge Fork which used PoA simulated as its consensus mechanism
- Deploy on local host with Port 7545, Network ID 5777
- Each account maintains 100 ETH
- The gas limit is 10000000
- The gas price is 0.02 ETH

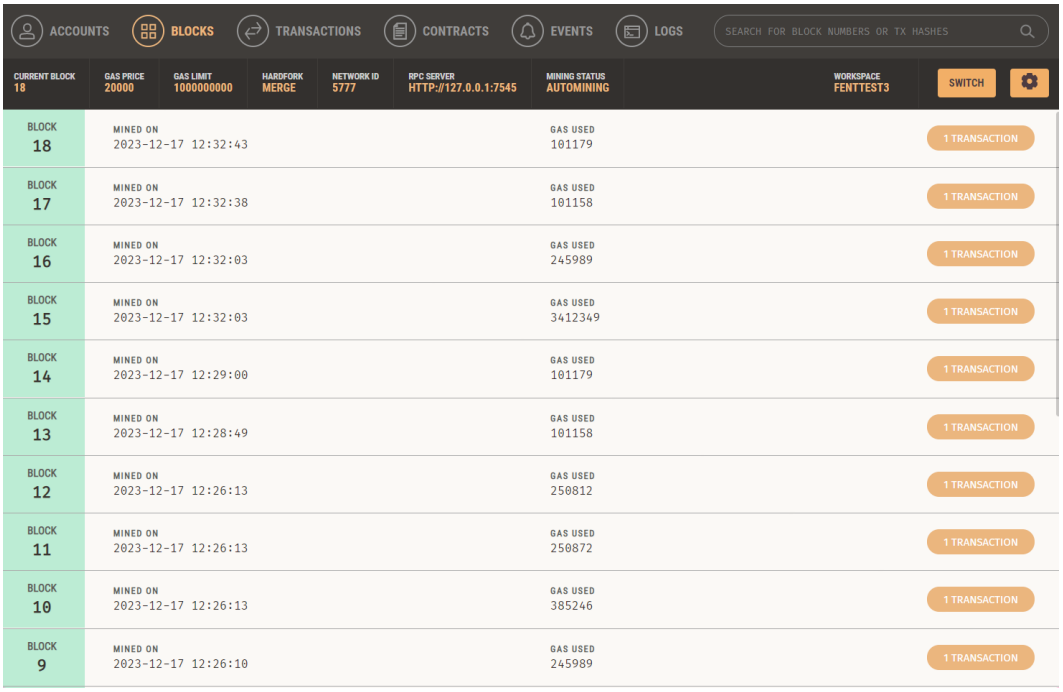

Figure 27: Ganache workspace.

Web-Hosting Service on Host

Hosting on local and use ngrok to able the live server.

```
ngrok

Introducing Pay-as-you-go pricing: https://ngrok.com/r/payg

Session Status                online
Account                       Methika Aungkurboribhun (Plan: Free)
Version                       3.5.0
Region                        Asia Pacific (ap)
Latency                       28ms
Web Interface                 http://127.0.0.1:4040
Forwarding                    https://f3b1-203-131-209-66.ngrok-free.app -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              270     0       0.07    0.08    0.03    0.12

HTTP Requests
-------------

GET /static/img/logo.png               304 NOT MODIFIED
GET /search                            200 OK
GET /static/css/template.css           304 NOT MODIFIED
GET /static/script/metamask_connect.js 304 NOT MODIFIED
GET /static/script/metamask_connect.js 304 NOT MODIFIED
GET /static/css/template.css           304 NOT MODIFIED
GET /static/img/logo.png               304 NOT MODIFIED
GET /search                            200 OK
GET /static/css/template.css           304 NOT MODIFIED
GET /static/img/logo.png               304 NOT MODIFIED
```

Figure 28: Terminal.

Set up and deployment

1. Open Ganache and set up Ganache environment.
2. Connect MetaMask account to the extension.
3. Open file in VSCode.
4. Create Virtual Environment of Python.
5. Run init.py.
6. Optional: Run add_sample.py to add initial sample.
7. Run app.py.
8. Open http://127.0.0.1:5000/ on web browser to open application.
9. Enjoy application.

Chapter 4

Evaluation

User Experience Evaluation

Positive Comments

- Responsive in both desktop and mobile users
- Easy to use and understand.
- Take a little time to create a new title deed.

Negative Comments

- The design may be too simple.
- Regular users want to request.

Functionality Evaluation

Title deed adding

After filling out the form then click add button, adding new deed title block chain triggers smart contract add_sheet_list. Able to create 2 blocks due to the smart contract triggered. Terminal shows success statement. Front end website shows able to deploy smart contract. Block generated 2 blocks in the Ganache.

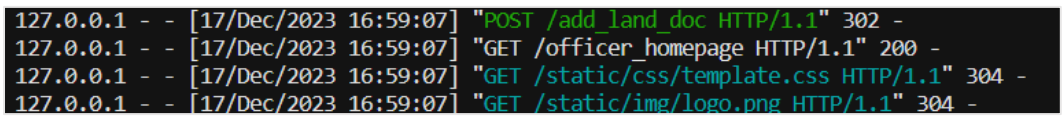

Figure 29: Title deed added terminal output shown.

Figure 30: website response after generating new staff.

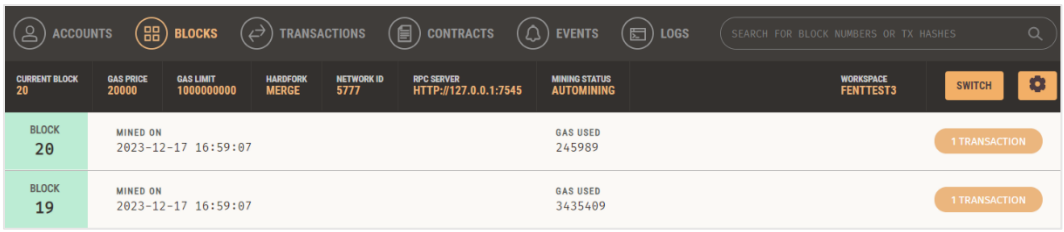

Figure 31: Block created after generating new staff.

Title deed transferring

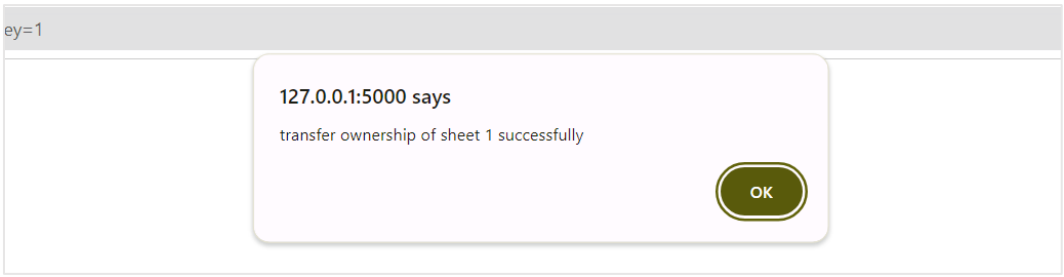

Figure 32: Block created after title deed transferring.

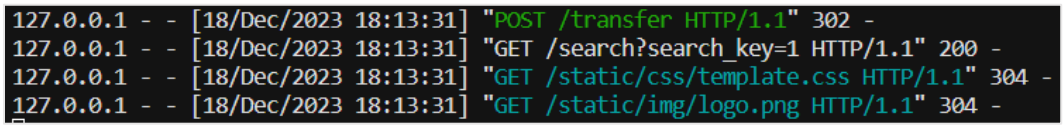

Figure 33: Block created after title deed transferring.

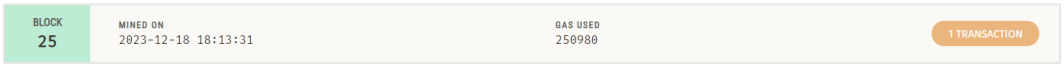

Figure 34: Block created after title deed transferring.

Title deed deactivation

This function triggers smart contract to create 1 block.

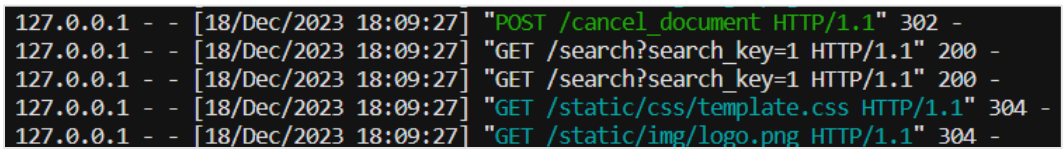

Figure 35: Block created after title deed deactivation.

Figure 36: Block created after title deed deactivation.

Title deed reactivation

This function triggers smart contract to create 1 block.

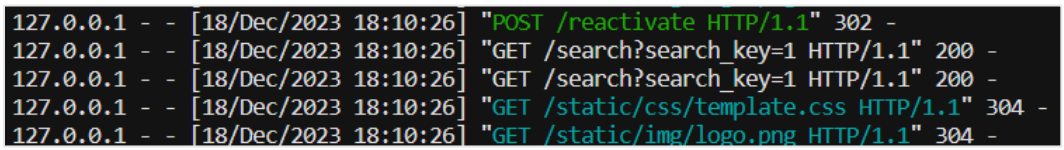

Figure 37: Block created after title deed reactivation.

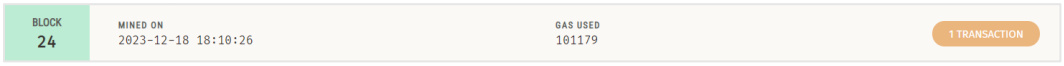

Figure 38: Block created after title deed reactivation.

Land image forming

Land coordinates forming located in Thailand using test coordinates of [14.069183, 100.597882] [14.068915, 100.600227] [14.067052, 100.600603] [14.066855, 100.597861] can result the image below. Which is related to the shape that we used to construct coordinates.

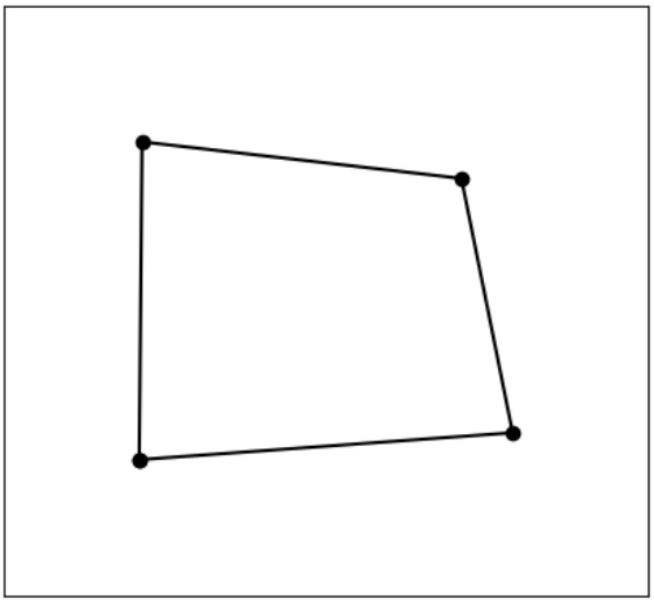

Figure 39: Land shape formed by coordinates.

Chapter 5

Conclusion and Discussion

Conclusion

In summary, our capstone project involves designing and developing a decentralized application for a title deed system, aiming to enhance the credibility and security of title deed data through the unique features of blockchain technology. Additionally, we have incorporated a web-based service with a robust authentication system to cater to a diverse user base, including desktop users, mobile users, and various devices or platforms. It's important to note that our project is currently deployed in simulation environments, and as a result, certain challenges in real-world scenarios may not have been fully addressed. The evaluation process, encompassing both functional and performance evaluations, remains a crucial criterion for assessing the overall effectiveness of our solution.

Obtained Benefits

1. Understand how to utilize the characteristics of the blockchain and its Smart contract.
2. Understand how to design and develop a full system of application including both UI/UX design, front-end design, and back-end design.
3. Understand how to develop decentralized applications.

Further work

User is able to request for adding, modifying, and deleting the deed title. Then staff user is able to accept or decline the regular user request.

Improve UI design to have more visual.

References

Antolin, M. (2022). *What Is Proof-of-Authority?* https://www.coindesk.com/learn /what-is-proof-of-authority/

Cointelegraph. (March 2023). *Proof-of-authority vs. proof-of-stake: Key differences explained*. https://cointelegraph.com/blockchain-for-beginners/proof-of-authority-vs-proof-of-stake-key-differences-explained.

Ben Lutkevich. (2020). *HTML (Hypertext Markup Language).* https://www.theserverside.com/definition/HTML-Hypertext-Markup-Language

Buterin, V. (2014). *A Next-Generation Smart Contract and Decentralized Application Platform.* https://ethereum.org/en/whitepaper/

ConsenSys Software Inc. (2022). *Ganache Ethereum workspace overview.* https://trufflesuite.com/docs/ganache/concepts/ethereum-workspace/overview/

GeeksforGeeks. (2022). *Consensus Algorithms in Blockchain*. https://www.geeksforgeeks.org/consensus-algorithms-in-blockchain/

Georgia Weston. (August 2022). *What is Ganache Blockchain* https://101blockchains.com/ganache-blockchain/

Abdelhadi Dyouri. (August 2021). *How To Create Your First Web Application Using Flask and Python 3.* https://www.digitalocean.com/community/tutorials/how-to-create-your-first-web-application-using-flask-and-python-3

Gregory McCubbin. (March 2023). *Intro to Web3.py, Ethereum for Python Developers* https://www.dappuniversity.com/articles/web3-py-intro

James Humphreys. (April 2023). *What is traceability and how to implement in manufacturing?.* https://katanamrp.com/blog/product-traceability/

King, S., & Nadal, S. (2012). *PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake.* https://peercoin.net/assets/paper/peercoin-paper.pdf

Phemex. (2022). *What Is Proof-of-Authority: Staking Credibility Instead of Coins*. https://phemex.com/academy/what-is-proof-of-authority

Venafi, Inc. (2022). *What is PKI and How Does it Work?.* https://venafi.com/machine-identity-basics/what-is-pki-and-how-does-it-work/

Wikipedia. (May 2023). ***Blockchain***. https://en.wikipedia.org/wiki/Blockchain

Techtarget. (August 2022). Bootstrap. https://www.techtarget.com/whatis/definition/bootstrap