

**SCHOOL OF INFORMATION, COMPUTER AND COMMUNICATION
TECHNOLOGY
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY**

Final Project

Writing Smart Contracts and Decentralized Application

Group Members

Onvara Rattanapatumwong	6322770361
Napatsorn Rattanapan	6322771559
Sudarat Buatas	6322773159

**Present to
Dr. Watthanasak Jeamwatthanachai**

**DES 484 Blockchain Development
Semester 1/2023 Academic Year 2023
Digital Engineering (DE)**

Table of content

Abstract	3
Chapter 1	4
Overview	4
The development of the smart contract backend	4
The frontend interface for the decentralized application.	5
Color Palettes	5
Mock up design	5
Chapter 2	9
Case Study Selection	9
Chapter 3	11
Smart Contract Development	11
User Registration	11
Publisher Management	11
Payment System	12
Ebook Rental	12
Chapter 4	13
Frontend Development	13
Chapter 5	18
Deployment	18
Testing	20
Conclusion	22
Appendix	23
Reference	24

Abstract

As cryptocurrency and blockchain tech rapidly evolved, platforms like Ethereum started backing different smart contract types. These contracts act as computer protocols, designed to digitally assist, verify, or enforce contracts. They find wide application across financial services, prediction markets, Internet of Things (IoT), and more. This project introduces you to the smart contract lending e-book system writing by solidity is a revolutionary platform leveraging blockchain technology to facilitate decentralized lending processes. This system aims to provide a secure, transparent, and efficient lending environment while eliminating intermediaries and ensuring trust through smart contracts.

Chapter 1

Overview

The smart contract lending e-book system represents a shift in the lending landscape, offering a decentralized, transparent, and efficient alternative to traditional lending models. By harnessing the power of blockchain and smart contracts, it aims to democratize access to lending while ensuring security, reliability, and user empowerment.

The development of the smart contract backend

Currently, leave management applications operate within centralized client-server systems, a process demanding considerable time and human effort for approvals. This centralization necessitates users to place unwavering trust in the server, presenting risks of potential data damage or misuse for personal gain. To address these challenges, the integration of innovative technology, such as decentralized applications leveraging smart contracts, becomes imperative to overcome these limitations effectively.

In traditional real-world contracts, intermediaries are necessary to verify and enforce agreements among multiple parties. These intermediaries validate signings and ensure contract conditions are met. However, a smart contract is unique in its self-execution, eliminating the need for intermediaries. Enabled by cryptographic techniques, digital signatures, and Blockchain a decentralized, unalterable ledger smart contracts autonomously execute, making them independent of intermediaries and ensuring contract fulfillment seamlessly.

In this project we aim to create a smart contract using Solidity for an E-book lending system aimed to tackle prevalent issues like restrictive Digital Rights Management (DRM) limitations and accessibility challenges, while also addressing the economic barriers faced by libraries and institutions.

Smart contracts offer a transformative solution to these challenges by introducing flexibility in Digital Rights Management (DRM) solutions. For instance, they enable borrowers to lend e-books for defined durations and empower publishers to set varied prices for different user categories. Accessibility concerns, particularly for individuals with disabilities, can also be mitigated through smart contracts by ensuring e-books are compatible with assistive technologies like screen readers.

Moreover, the advantages of smart contracts extend beyond addressing these challenges:

- Enhanced Security, their self-executing nature and tamper-resistant properties bolster transaction security, reducing the likelihood of fraudulent activities.
- Automation of lending tasks through smart contracts streamlines processes, enhancing overall efficiency.
- By automating agreement terms between borrowers and lenders, smart contracts minimize errors and disputes, reducing associated risks.
- Smart contracts leverage blockchain to provide a transparent transaction record, fostering transparency in the lending process.

In summary, implementing smart contracts in e-book lending platforms not only addresses current technical and economic hurdles but also brings forth a range of benefits that enhance security, efficiency, risk mitigation, and transparency within the lending ecosystem.

The frontend interface for the decentralized application.

Since we are a lending e-book platform, our primary focus is to enhance user-friendliness to encourage accessibility, retention, competitiveness, adoption, and reduced friction in user interactions. Following this, we'll showcase our selected color palette and the reasons behind the choices, followed by a mock-up design for our website.

Color Palettes

#FFFFFF (white)

#E2E2E2 (light gray)

#000000 (black)

#00B863 (teal)

The meaning conveyed by these colors is a clean and modern appearance. White highlights content, while gray sharpens other elements, creating a clean look. Black underlines the topic, while teal adds a touch of personality.

Our decision to prioritize e-books over paper books stems from our commitment to reducing waste and encouraging a shift towards digital reading. We aim for our website to be recognized for its utilization of smart contracts, ensuring enhanced security measures.

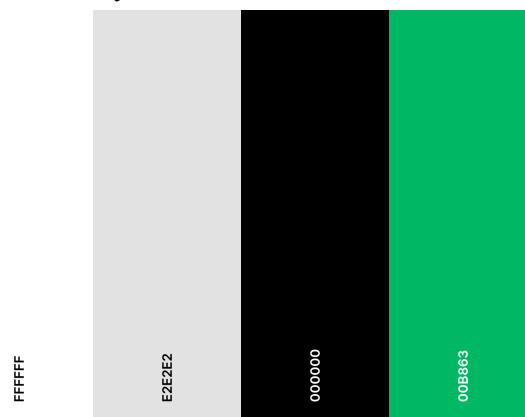


Figure 1 : Color palette

Mock up design

In our decentralized application a Register page acts as to fill the information about the user. Once all information is filled, clicking the 'submit' button records the user's registration, successfully creating the account.

For our application that doesn't contain the Login page because. If a user has been logged in before, the system would check the user information and how user have created the account.

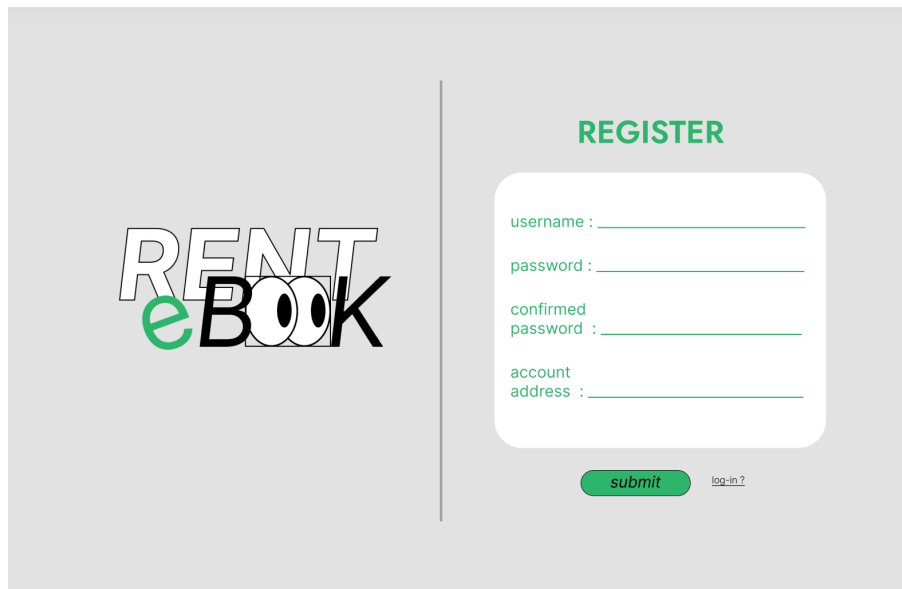
The image shows a web page for registering an account. On the left, there is a logo for 'RENT eBook' where 'RENT' is in a large, outlined font, 'e' is in green, and 'BOOK' is in a bold, black font. To the right of the logo is a vertical line. Further right, the word 'REGISTER' is displayed in green. Below this, there is a white rounded rectangle containing four input fields: 'username :', 'password :', 'confirmed password :', and 'account address :'. Each field has a green underline. At the bottom of this rectangle is a green button labeled 'submit'. To the right of the 'submit' button is a link that says 'log-in ?'.

Figure 3 : Register page

Upon successful Login, users are directed to the Home page where they can explore books of interest categorized into literature, magazines, novels, and comics. Positioned at the top right corner, users can view their coin balance, essential for book lending. To access lending history and check remaining lending periods, users can click on the book icon also located on the top right corner for detailed information.

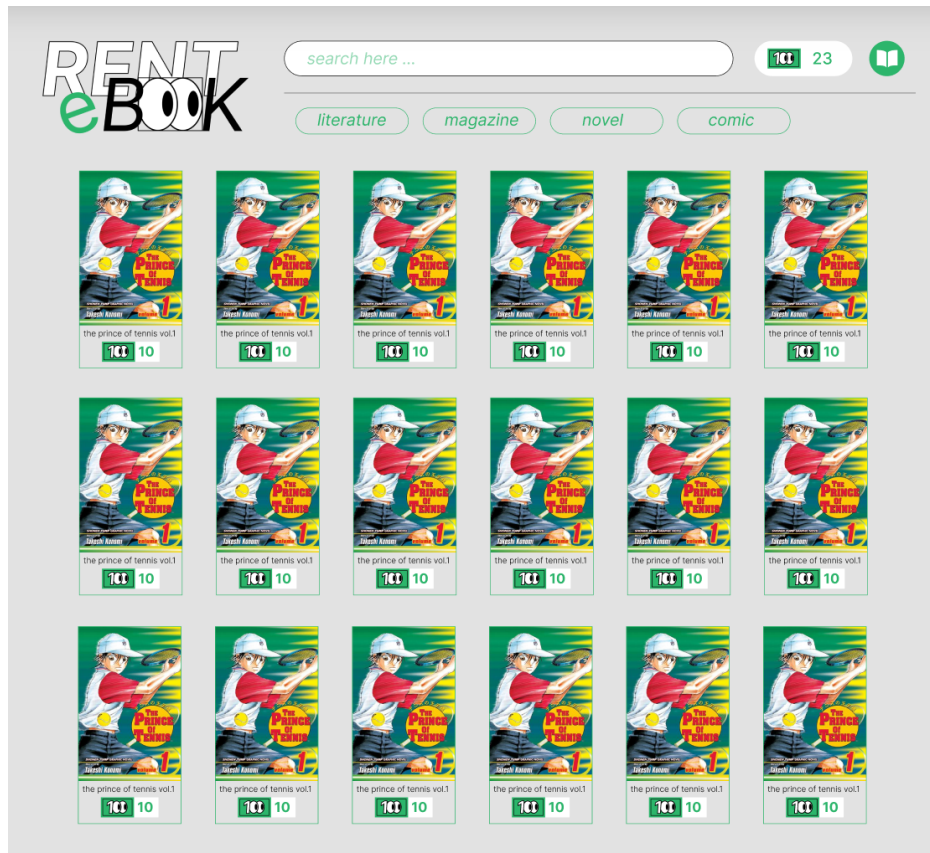


Figure 4 : Home page

Upon clicking the book icon, users are directed to the Bookshelf page where their username, profile picture, and the option to modify profile details are prominently displayed. Positioned below, the coin balance is showcased, offering the capability to top-up coins for lending purposes.

The focal point of this page lies in presenting the book lending history and the remaining time period for each lending session. To ensure user attention and acknowledgment of lending periods, colors signify the time left:

- Green denotes more than 100 days remaining in the lending period.
- Yellow indicates 100 days or less remaining in the lending period.
- Red serves as a critical reminder, signifying the end of the lending period, restricting access to the e-book until a new lending purchase is made.

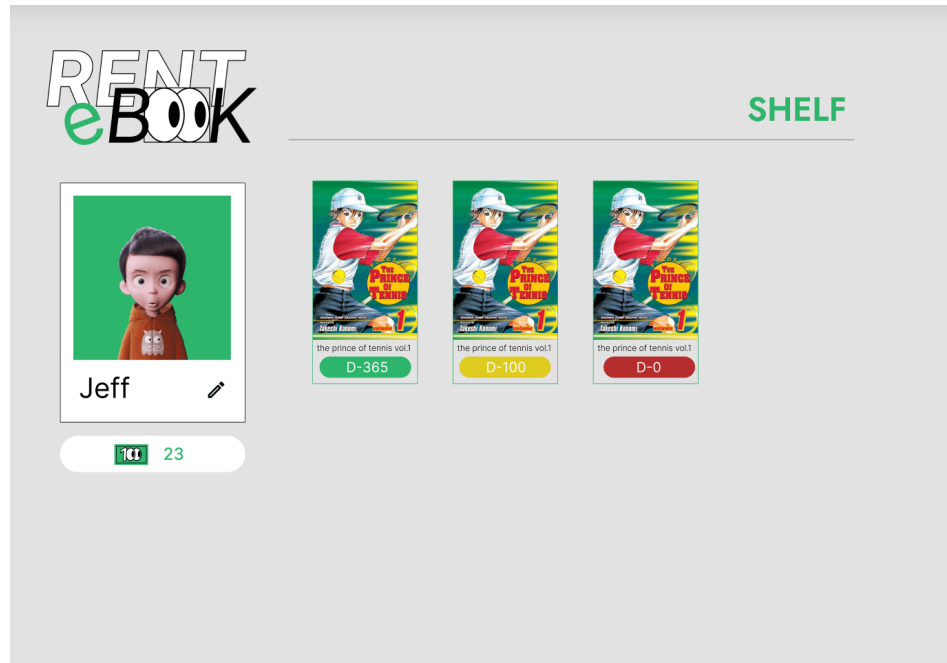


Figure 5 : Bookshelf page

Chapter 2

Case Study Selection

One case study in the E-book lending system is OverDrive, OneDrive founded in 1986, is a leading digital media platform providing libraries with e-books, audiobooks, and streaming video for lending to their patrons. It's a prime example of a successful e-book lending platform in the real world, overcoming many challenges faced by traditional systems.

E-books face challenges as they are manually shared via links or download options, lacking the automation and security found in specialized lending platforms. OneDrive's limitations include the absence of loan period settings, return reminders, or automatic deletion of borrowed files, making it arduous to track and manage lending activities. Additionally, OneDrive's native DRM support is constrained, posing difficulties in controlling the copying or distribution of borrowed e-books.

Implementing a decentralized application using smart contracts could alleviate these issues. For instance, by removing reliance on centralized platforms like OverDrive, smart contracts enable direct lending and borrowing of e-books among libraries and individuals. This decentralized system reduces costs and boosts transparency. Picture a network where libraries and individuals are interconnected through a blockchain-based platform, fostering a peer-to-peer lending model that diversifies e-book selections and offers flexible borrowing options.

Moreover, smart contracts' self-executing and immutable nature guarantees the security and transparency of transactions, ensuring that agreements, from borrowing to returning an e-book, remain unaltered. Additionally, leveraging smart contracts allows for more adaptable DRM solutions, empowering libraries and individuals to define specific borrowing terms such as loan duration and the number of simultaneous readers.

Here's why e-book lending is a good fit for a decentralized application.

Decentralization

- No intermediaries : Eliminates the need for centralized platforms like OverDrive, reducing costs and increasing transparency. Libraries and individuals can directly lend and borrow e-book, fostering a peer-to-peer lending model.
- Reduced reliance on trusted third parties : DApps operate on blockchain networks, eliminating the dependence on a single entity for managing transactions and data. This reduces the risk of censorship, manipulation, or downtime.

Security and Transparency

- Tamper-proof transactions : Smart contracts used in dApps are self-executing and immutable, ensuring secure and transparent transactions. All lending and borrowing activities are recorded on a blockchain, providing a verifiable audit trail.
- Reduced fraud risk : Automated dispute resolution mechanisms built into smart contracts can minimize the risk of fraud and ensure fair outcomes for both lenders and borrowers.

Flexibility and Innovation:

- Open standards : DApps can be built on open standards, allowing for compatibility with different e-reader devices and platforms, enhancing accessibility and user experience.
- Customizable lending options : Smart contracts can be programmed with flexible terms and conditions, enabling lenders to set loan durations, access restrictions, and even micro-payments for reading time.

Improved User Experience:

- Streamlined borrowing and lending : DApps can offer user-friendly interfaces for searching, borrowing, and returning e-books, simplifying the lending process.

Chapter 3

Smart Contract Development

In our e-book lending decentralized application, our focus is on developing smart contracts enabling user registration, facilitating payments systems for selected items, implementing an e-book rental verification system, and exclusively allowing publishers to register and display books for lending purposes.

As above so that our decentralized application consist of 4 smart contract including :

User Registration

On the “UserRegistration” file it manages user registration and profile updates.

The contract enables users to register with a username, email, and password hash. It maintains a mapping of user addresses to their respective profile information and provides functions for registration and profile updates while enforcing security measures like verifying passwords.

Users can register and once registered, they can update their profile information, ensuring that only the owner or registered users can execute specific functions on the application. Additionally, the contract emits events to notify external systems or applications about user registration and profile updates.

Publisher Management

On the “PublisherManagement” file it facilitates the management of publishers and their respective books.

The contract enables publishers to register by submitting their name and contact details. Once registered, publishers can expand their catalog by adding books, specifying details like book name, author, description, and price. Structured storage within the contract organizes book details by both publishers and unique book IDs. Additionally, the contract offers a function to retrieve book information using the publisher's address and book ID, ensuring accessible and transparent access to book details

This contract forms a basic framework for managing publishers and their respective book catalogs on the blockchain, allowing publishers to register and add books to their inventory, and users to retrieve book information from specific publishers.

Payment System

This system facilitates payments between users by allowing one user to send Ether (cryptocurrency) directly to another address (the publisher that the user is lending to). It performs checks to ensure the payment matches the specified amount and verifies the success of the payment transaction.

Ebook Rental

This contract serves as the core of a decentralized ebook rental system, streamlining the user experience for renting books on the blockchain. It verifies user registration, fetches book details, securely processes payments, and maintains a transparent record of rented books.

By validating user registration through the UserRegistration contract and obtaining essential book information from PublisherManagement, the contract ensures the legitimacy of users. Secure payment transactions are made possible through integration with the PaymentSystem contract. Notably, the system maintains transparent rental records, allowing users to track their rental history and expiration status.

Chapter 4

Frontend Development

In light of the ongoing development process, certain components have been modified and may not align perfectly with the initial mockup. The development framework employed for this project utilizes Parcel as the development tool, with HTML, CSS, and JavaScript serving as the primary development languages. Additionally, the Metamask and Ethers libraries are incorporated as intermediaries to facilitate interaction with the webpage.

The homepage is designed to display a list of books available for users to rent. The top panel features a search bar to facilitate book searches. Adjacent to the search bar is a cash icon, representing the token we intend to integrate into our decentralized application (dApp). Lastly, a shelf icon is included to provide a direct link to the profile page. To enhance user navigation, genre categories such as action, comedy, romance, fantasy, and drama are provided beneath the top panel. These categories aim to assist users in exploring different types of books.

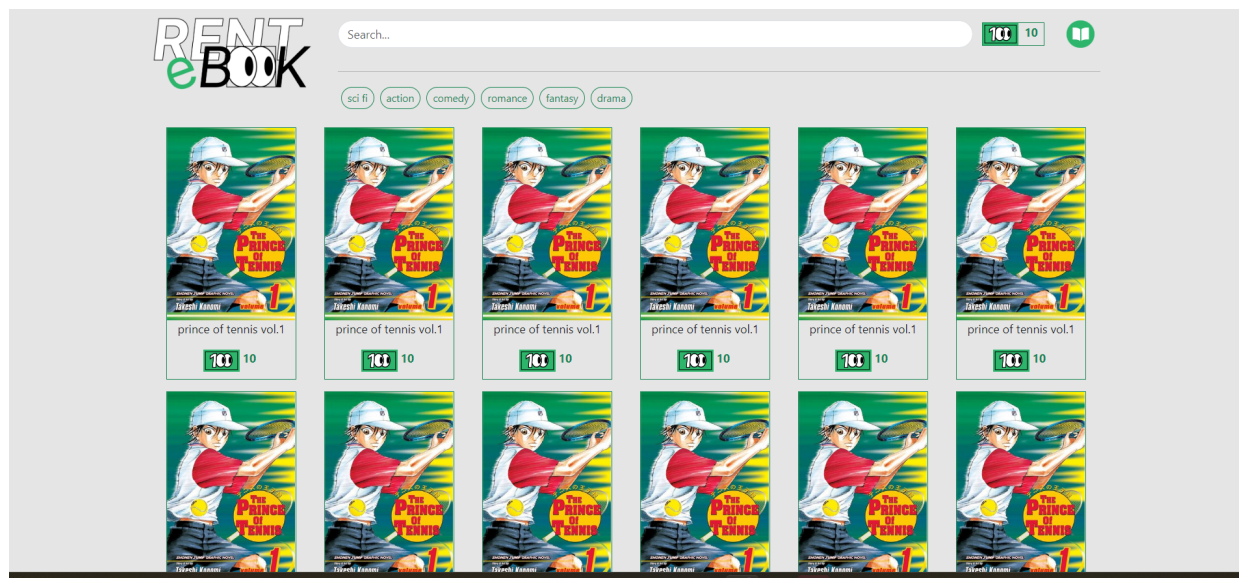


Figure 6 : home page

The Profile or Shelf page is multifunctional. Primarily, it displays the books listed under the shelf, which represent the books that the user has rented, along with the number of days remaining for each rental. On the left side of the book list, the user's profile details are displayed. These details can be updated by clicking on the yellow icon, which will then display the page as shown in Figure 8 below. This design allows for a seamless and user-friendly experience.

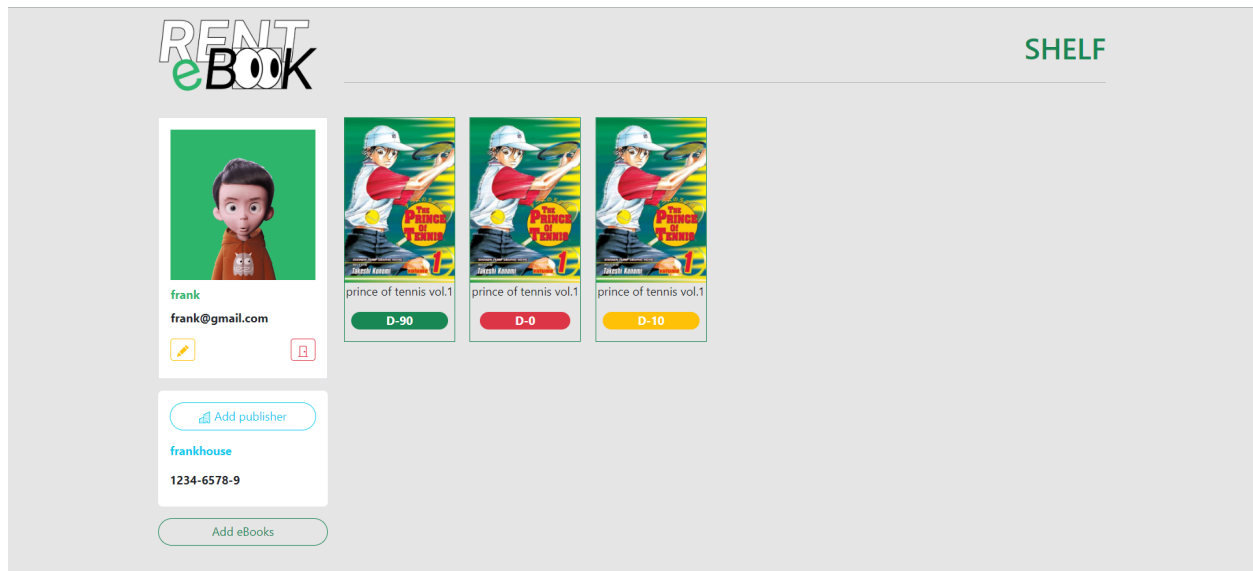


Figure 7 : profile page

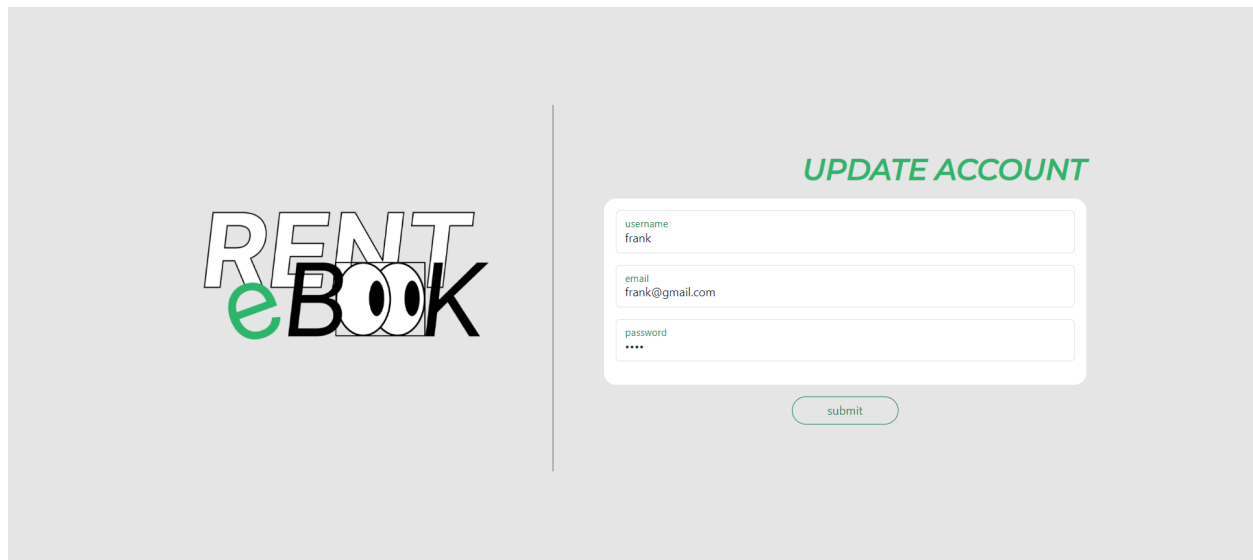


Figure 8 : Update profile page

From the Profile page, if you select the logout button (represented by a red icon) located next to the update profile option (represented by a yellow icon), you will be redirected to the registration page. This page allows users to register a new account. If an attempt is made to register an account that has already been registered, the page displayed will resemble Figure 10. However, if you alter the account in the Metamask wallet, the resulting page will be similar to Figure 12.

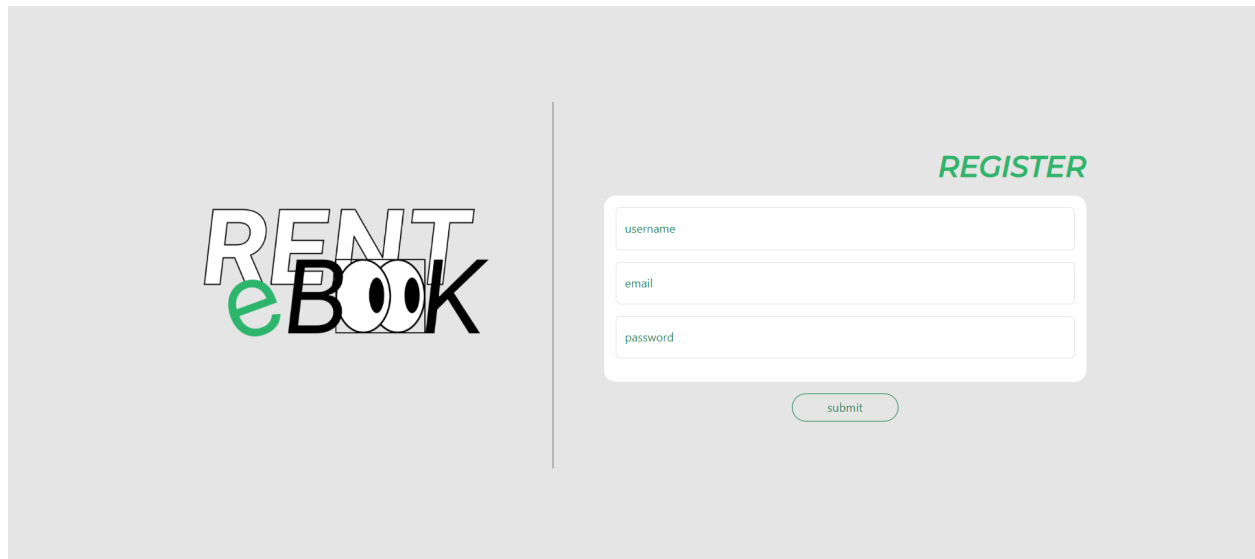
The image shows a registration page for 'RENT eBook'. On the left is the logo, which consists of the word 'RENT' in a white, outlined, sans-serif font, and 'eBOOK' in a green, bold, sans-serif font. A vertical line separates the logo from the registration form on the right. The form is titled 'REGISTER' in green. It contains three input fields: 'username', 'email', and 'password', each with a small green icon to its left. Below the fields is a green 'submit' button.

Figure 9 : Register page

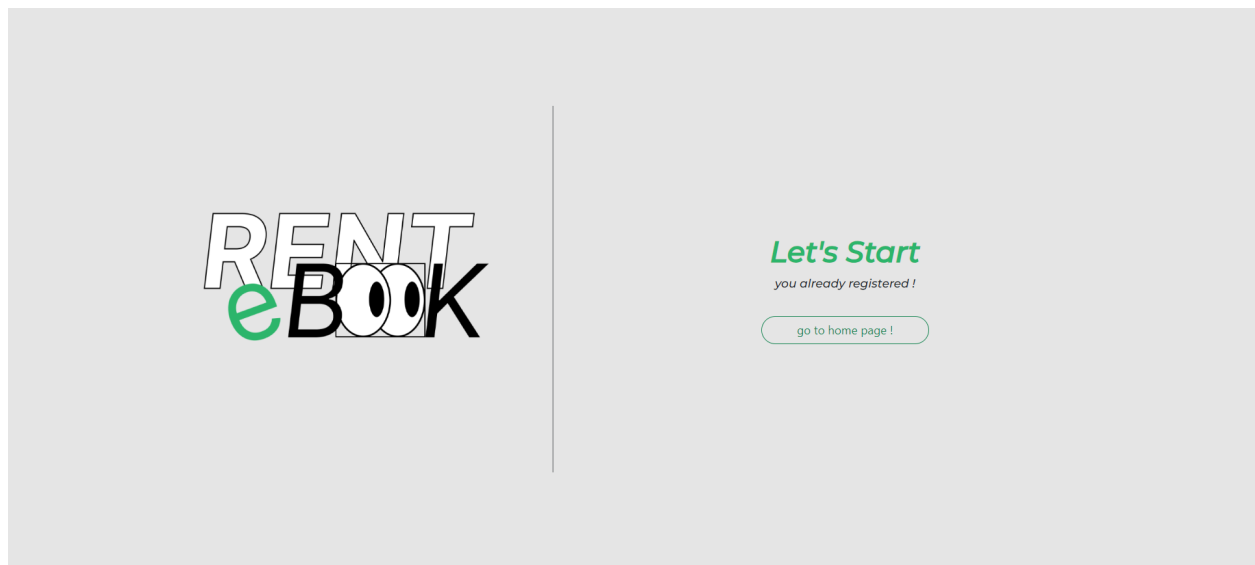
The image shows the result of a registration attempt. On the left is the same 'RENT eBook' logo as in Figure 9. A vertical line separates the logo from the result message on the right. The message is titled 'Let's Start' in green, followed by 'you already registered !' in a smaller green font. Below the message is a green button that says 'go to home page !'.

Figure 10 : Register Result

The Add Publisher button, represented by a blue icon, is designed for registering a publisher account. This functionality enables you to add eBooks to our website. Upon successful registration, the Metamask panel will be displayed for you to confirm the transaction. This design ensures a secure and efficient user experience. Please refer to Figure 12 for a visual representation of this process.

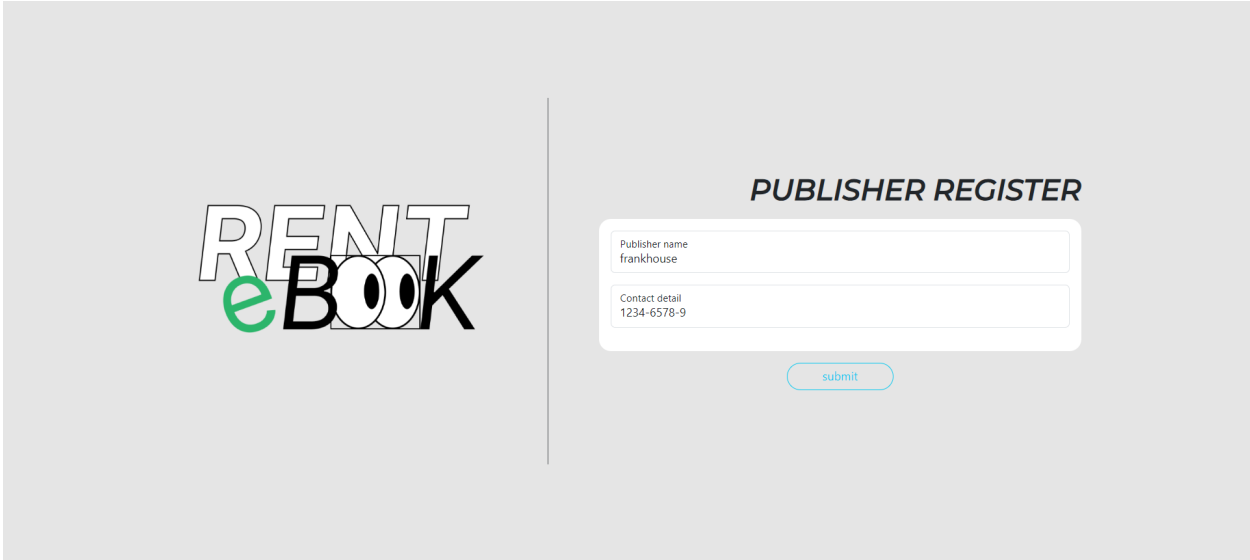


Figure 11 : publisher register page

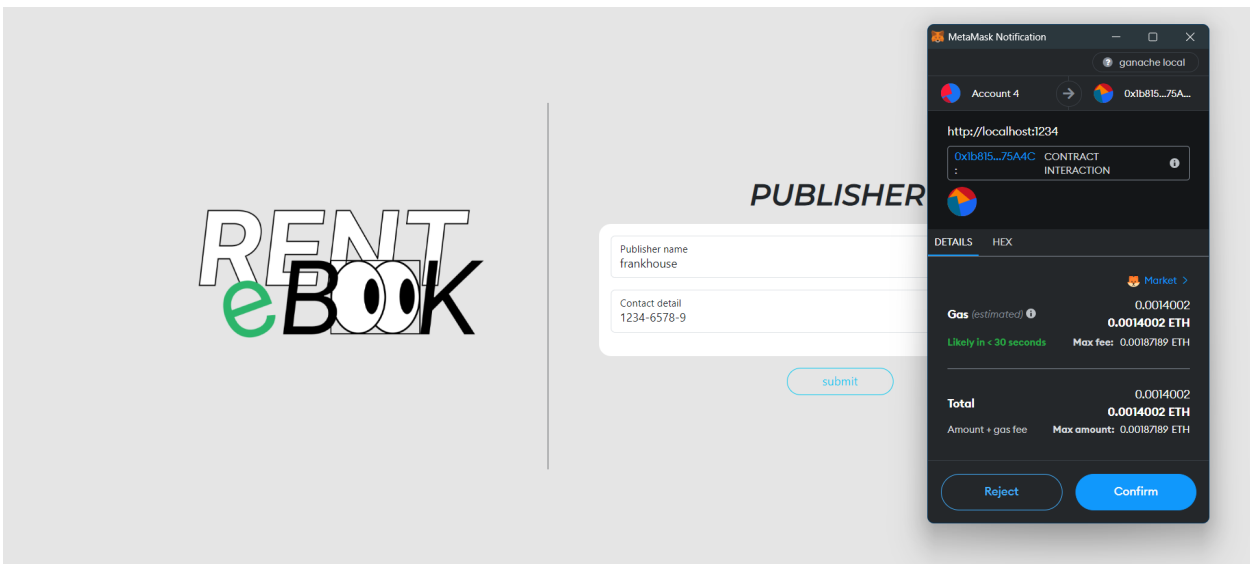


Figure 12 : metamask panel

Finally, users have the ability to add eBooks by clicking on the ‘Add eBooks’ button. This action will display a modal where users can enter the details of the eBooks they wish to add. This feature enhances the interactive nature of our platform, allowing users to contribute to our eBook collection.

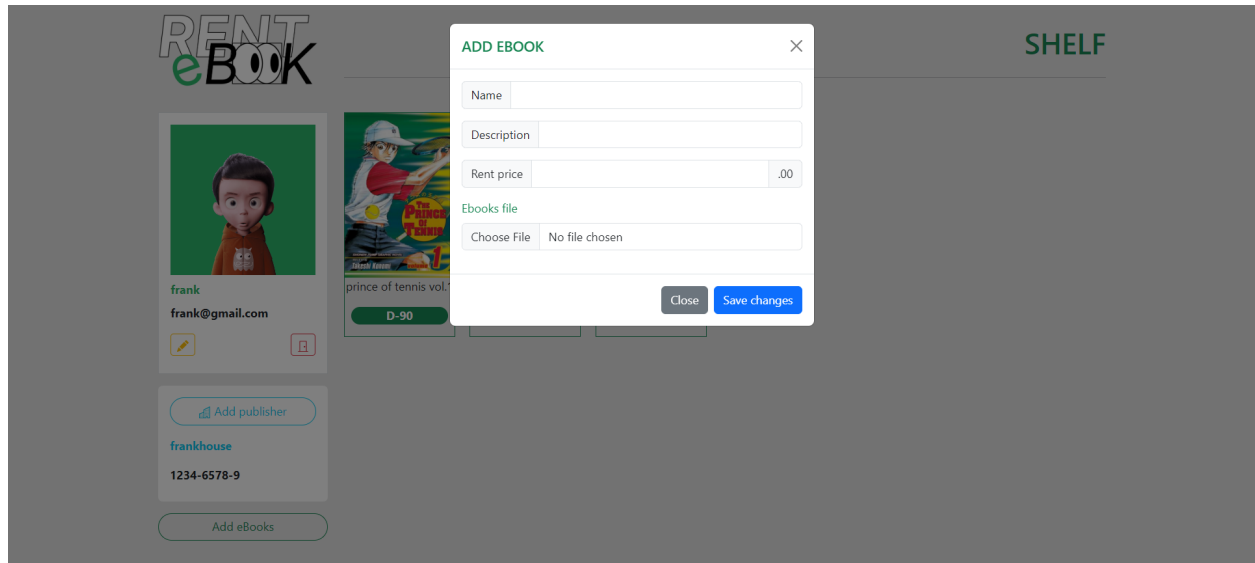


Figure 13 : add ebooks modal

Chapter 5

Deployment

We utilized the Truffle framework, a widely adopted development framework for Ethereum, to deploy our smart contracts. Initially, we imported Ethereum smart contract artifacts for four contracts: UserRegistration, PublisherManagement, PaymentSystem, and EbookRental.

Following this, we deployed the UserRegistration, PublisherManagement, and PaymentSystem contracts individually, securing instances of each for subsequent interactions. These instances serve as the interface for programmatically engaging with the deployed contracts.

Subsequently, we deployed the EbookRental contract, a pivotal component, by providing it with three parameters. Notably, these parameters are Ethereum addresses corresponding to the previously deployed UserRegistration, PublisherManagement, and PaymentSystem contracts. This linking mechanism establishes essential connections and dependencies between the contracts, forming a cohesive Ethereum-based system for managing ebook rentals.

```
1_deploy_ebookRental.js
=====

Replacing 'UserRegistration'
-----
> transaction hash: 0xf7656cbe82c985fe817d2ac530995a183dd9ae583f8ae4714655bbd32d4bf383
> Blocks: 0        Seconds: 0
> contract address: 0xd364A2BAf8f02061ebc6898b8cb60586B2216a40
> block number:    1
> block timestamp: 1702812905
> account:         0xfb093b00b2a32B78C162f84Ac9BD0d78Ff9a39d8
> balance:         99.996130387
> gas used:        1146552 (0x117eb8)
> gas price:       3.375 gwei
> value sent:      0 ETH
> total cost:      0.003869613 ETH
```

Figure 14: Deployment of User Registration

```
Replacing 'PublisherManagement'
-----
> transaction hash: 0x48bcc46721dcfca65e049078a28caba72a7b1a979da0b9d4be0cb5870669dd9b
> Blocks: 0        Seconds: 0
> contract address: 0x435996bfF3b59f00c80f8649A2BE0683055580A5
> block number:    2
> block timestamp: 1702812905
> account:         0xfb093b00b2a32B78C162f84Ac9BD0d78Ff9a39d8
> balance:         99.992583997446011232
> gas used:        1073708 (0x10622c)
> gas price:       3.302936696 gwei
> value sent:      0 ETH
> total cost:      0.003546389553988768 ETH
```

Figure 15: Deployment of Publisher Management

Replacing 'PaymentSystem'

```
-----
> transaction hash: 0x7d9d9f4b91686d4088ef4fa557d22ca318ca9ed1b7c24ff76c56d83eb9356f67
> Blocks: 0        Seconds: 0
> contract address: 0xdA09618B6e17f9a94eA52F4F3551a378FCAEF123
> block number:    3
> block timestamp: 1702812905
> account:         0xfB093b00b2a32B78C162f84Ac9BD0d78Ff9a39d8
> balance:         99.991593549558913941
> gas used:        306201 (0x4ac19)
> gas price:       3.234633091 gwei
> value sent:      0 ETH
> total cost:      0.000990447887097291 ETH
```

Figure 16: Deployment of Payment system

Replacing 'EbookRental'

```
-----
> transaction hash: 0x5c7ab762921cbdfaec22d3c020559e0873427cfd07de6f0d116338b2d942e0e2
> Blocks: 0        Seconds: 0
> contract address: 0xef6E5167606eE903e50942d80DcCa66A2817d3C6
> block number:    4
> block timestamp: 1702812905
> account:         0xfB093b00b2a32B78C162f84Ac9BD0d78Ff9a39d8
> balance:         99.987824907795022041
> gas used:        1195950 (0x123fae)
> gas price:       3.151170002 gwei
> value sent:      0 ETH
> total cost:      0.0037686417638919 ETH
```

Figure 17: Deployment of Ebook rental

> Saving artifacts

> Total cost: 0.012175092204977959 ETH

Summary

=====

> Total deployments: 4

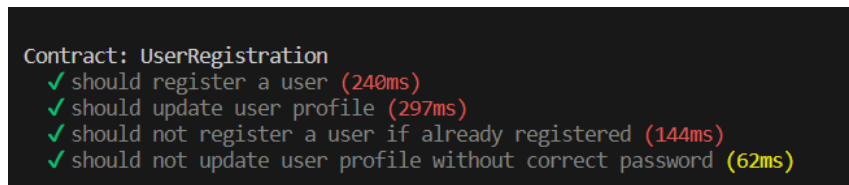
> Final cost: 0.012175092204977959 ETH

Figure 18: Summary of deployment

Testing

User Registration contract

1. Should register a user:
 - This test checks whether the contract allows the successful registration of a user.
2. Should update user profile:
 - This test verifies the functionality to update a user's profile.
3. Should not register a user if already registered :
 - The test ensures that the contract prevents the registration of a user who is already registered.should not
4. Update user profile without correct password :
 - This test checks that updating a user's profile requires the correct password for authentication.

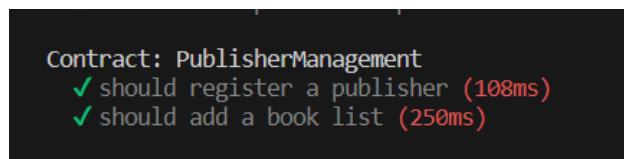


```
Contract: UserRegistration
✓ should register a user (240ms)
✓ should update user profile (297ms)
✓ should not register a user if already registered (144ms)
✓ should not update user profile without correct password (62ms)
```

Figure 19: Testing of User Registration contract

Publisher Management contract

1. Should register a publisher:
 - This test confirms that the contract allows the successful registration of a publisher.
2. Should add a book list:
 - The test checks the functionality to add a list of books by a publisher.



```
Contract: PublisherManagement
✓ should register a publisher (108ms)
✓ should add a book list (250ms)
```

Figure 20: Testing of Publisher Management contract

Payment System contract

1. Should make payment
 - This test ensures that the payment system contract can successfully process a payment.
2. Should fail if the payment amount is incorrect

- This test checks if the payment system correctly fails when the payment amount is incorrect. The result indicates that this test passed.

```
Contract: PaymentSystem
✓ should make a payment (64ms)
✓ should fail if the payment amount is incorrect
```

Figure 21: Testing of Payment System contract

Ebook Rental contract

1. should rent book successfully
 - This test verifies that use can rent book successfully
2. Should not allow renting a book if the user is not registered:
 - This test checks that the contract prevents renting a book if the user is not registered. "User is not registered. Skipping rental test" is logged when the test script checks whether the user is registered before attempting to rent a book. If the user is not registered, the test script decides to skip the rental test since renting a book requires a registered user.
3. Should allow reading a rented book within the rental period:
 - This test checks whether a user can read a rented book within the rental period.
4. Should prevent reading a rented book after the rental period has expired:
 - This test checks that the contract correctly prevents reading a rented book after the rental period has expired
5. Should prevent reading a book that the user has not rented
 - This test confirms that the contract prevents reading a book that the user has not rented.

All tests in the Ebook rental are marked as passing but a persistent issue remains in the Ebook rental system related to user registration. The system encountered an error message indicating they are not registered, and despite our ongoing efforts to address this challenge, a resolution has proven difficult to attain.

```
Contract: EbookRental
✓ should rent book successfully (52ms)
User is not registered. Skipping rental test.
✓ should not allow renting a book if the user is not registered
Error during rental operation: Error: VM Exception while processing transaction: revert User is not registered -- Reason given: User is not registered.
canRead is undefined. Check for errors during the rental operation.
✓ should allow reading a rented book within the rental period
Error during the rental operation: Error: VM Exception while processing transaction: revert User is not registered -- Reason given: User is not registered.
canRead is undefined. Check for errors during the rental operation.
✓ should prevent reading a rented book after the rental period has expired
Error during canReadBook operation: Error: VM Exception while processing transaction: revert User is not registered
canRead is undefined. Check for errors during the canReadBook operation.
✓ should prevent reading a book that the user has not rented (70ms)
```

Figure 22: Testing of Ebook Rental contract

Conclusion

In conclusion, our project lays the groundwork for a decentralized eBook rental platform. It exemplifies the integration of various smart contracts, including User Registration, Publisher Management, Payment System, and eBook Rental contract, to form a unified system. Thorough testing has been conducted to ensure the validity of critical functionalities. However, we encountered challenges in testing the eBook rental contract. Due to the constraints of time and our current level of knowledge, we were unable to overcome these challenges. As a result, we have successfully completed some tests, but the testing for the renting functionality remains pending. This experience has provided us with valuable insights and will guide our future efforts in this domain.

Appendix

Source code : <https://github.com/Napatsorn1559/ebookRental>

Reference

[1] Barnett, A. (2018, November 16). The new OverDrive streamlines library eBook and audiobook lending - OverDrive. OverDrive.

<https://company.overdrive.com/2016/11/16/the-new-overdrive-streamlines-library-ebook-and-audiobook-lending/>