

FYP Final Report

Blockchain-based Ebook Transaction System for Transparent Book Sales

By

CHAN Hoi Chun

Advised by

Prof. Qian Chenxiong

FYP24045

COMP4801 Final Year Project 2024-2025

Department of Computer Science

The University of Hong Kong

Date of submission: April 21, 2025

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Professor Qian Chenxiong, for his invaluable guidance and support throughout my final year project. His clear orientation at the outset and assistance in resolving project-related challenges were instrumental in the successful completion of this work.

Abstract

The ebook industry's rapid expansion has exposed flaws in traditional royalty payment systems, often characterized by opacity and potential underpayment to authors. This report introduces a blockchain-based ebook transaction system designed to enhance transparency and enable authors to independently verify their royalties. Utilizing Ethereum smart contracts, the system automates royalty distribution for ebook purchases based on predefined rates, incorporating punitive measures to ensure timely payments. For subscription models, reading records stored on the InterPlanetary File System (IPFS) allow authors to calculate expected royalties, though validation depends on reader participation. A prototype featuring an Android application and backend server validates the system's feasibility, supporting industry-standard features like refunds and price adjustments. Key limitations include Ethereum's gas fees, which increase transaction costs for readers, and privacy concerns stemming from blockchain transparency. Future enhancements could involve alternative blockchains with lower fees and privacy-focused technologies. Beyond ebooks, this framework offers potential applications in other digital content sectors, such as music and video streaming, addressing similar royalty validation challenges and promising broader industry impact.

Table of Contents

	Page
Acknowledgment.....	I
Abstract.....	II
List of Figures.....	V
List of Equations	VI
1 Introduction.....	1
1.1 Project Overview	1
1.1.1 Background.....	1
1.1.2 Motivation	2
1.1.3 Proposed Solution.....	3
1.1.4 Objectives And Deliverables	3
1.1.5 Outline	4
1.2 Related Concepts and Terms	4
1.2.1 Ebook Purchases.....	4
1.2.2 Subscription Schemes For Reading Ebooks	5
1.2.3 Blockchain	6
1.2.4 Smart Contracts And Ethereum	7
1.2.5 IPFS	8
2 Literature Review	9
3 Methodology.....	10
3.1 System Background	10
3.2 System Overview	11
3.3 Smart Contract Design.....	15
3.3.1 Smart Contract Design For Ebook Purchases.....	15
3.3.2 Smart Contract Design For Ebook Subscriptions.....	17
3.4 IPFS For Ebook Subscriptions	19
3.5 System Design	21
3.6 Royalty Validation.....	24
3.7 System Implementation	25

3.7.1 Architecture	25
3.7.2 Scope and Assumptions.....	26
3.7.3 Ethereum Smart Contracts.....	26
3.7.4 Transaction Validation	27
3.7.5 Use cases.....	30
3.7.6 Database.....	33
4 Discussion	34
5 Conclusion	38
References.....	39
Appendices.....	43
Appendix A: pseudo-code of the framework contract for ebook purchases	43
Appendix B: pseudo-code of the framework contract for ebook subscriptions	46
Appendix C: Gantt chart of project schedule	47

List of Figures

Figure 1 The general typical ebook transaction process	1
Figure 2 Components of the system and their interactions without trusted third-party APIs	12
Figure 3 Components of the advanced system and their interactions with trusted third-party APIs	14
Figure 4 State diagram of smart contracts for ebook purchases	15
Figure 5 The timeline of contracts for ebook purchases	17
Figure 6 Fund flows in blockchain for ebook subscription without the collection of the platform account (left) and with the collection of the platform account (right)	18
Figure 7 State diagram of smart contracts for ebook subscription	18
Figure 8 Reading record structure of a subscription	20
Figure 9 The views of the relationship between users and records	20
Figure 10 A potential fraud case should be prevented in the system	22
Figure 11 Components and the relationship of the implemented system	25
Figure 12 Screenshots of the app in transaction validation	30
Figure 13 Use case diagram	31
Figure 14 ER diagram of the backend database	34

List of Equations

Equation 1 KDP ebook royalty formula for 35% royalty rate (Wogahn, 2024).....	4
Equation 2 KDP ebook royalty formula for 70% royalty rate (Wogahn, 2024).....	4
Equation 3 Royalty formula for ebook purchases	10
Equation 4 Fund formula for ebook subscriptions.....	10
Equation 5 Royalty formula for ebook subscriptions.....	10
Equation 6 Equation for generating record ID.....	21

1 Introduction

1.1 Project Overview

1.1.1 Background

Ebooks can be read on electronic devices, which provide an alternative path for reading books with the advantages of convenience, searchability, sale prices, and environmental friendliness compared with printed books (Staiger, 2012). The ebook market has become popular through the increase in the penetration rates of electronic devices and networks, as well as the change in reader habits. The market is still experiencing rapid growth in the current year. According to Errera (2024), the number of eBook sales in the United States rose from 69 million to 191 million between 2010 and 2020. It is also estimated that there will be more than 1.2 billion ebook users globally in 2027. Revankar (2025) mentioned that the global market is expected to reach a total of USD 15 billion in 2027. As the ebook market becomes a huge industry which have a large amount of transactions and fund flow, it is worth reviewing the current sales models of the ebook industry to figure out whether the models can be changed with the help of new technology in order to improve the industry.

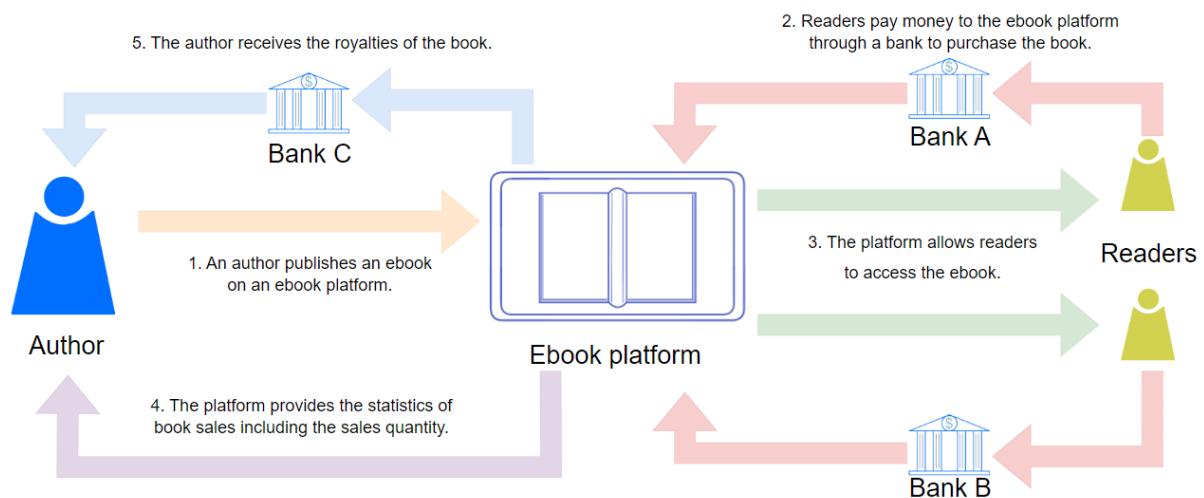


Figure 1 The general typical ebook transaction process

Although the roles and their functions in the existing ebook transaction models can be different, they generally involve three main parties: authors, readers, and the platform (Nizamuddin et al., 2019). The general typical ebook transaction process, which involves the

parties, is shown in **Figure 1**. Authors create ebook content and publish ebooks on a platform. Readers can pay money to the platform for obtaining the license to read the ebook on the platform software. Ebook platforms serve as digital distribution channels and control access through licensing. Platforms will count the sales quantity and provide the authors with royalties and relevant statistics. Although an intermediary platform is not necessary in ebook trading, it is common in the current ebook industry as it allows authors to focus on creation by bearing part of the work for a better user experience for readers, convenience in publishing ebooks, and a large number of readers (Hill, 2021). Therefore, authors and even small publishers may prefer relying on large ebook platforms to sell ebooks. For example, Kindle Direct Publishing (KDP) of Amazon is the largest ebook platform, which captures about 70% of ebook sales (Yoo, 2024). However, the business model has a potential issue that may harm the benefits of authors by receiving incorrect royalties.

1.1.2 Motivation

From the perspective of business and law, authors as licensors allow the platform to sell their ebook, but the platform needs to pay the royalties to authors based on the protocol and the actual sales. While in this scenario, the platform as a licensee may report fewer sales than actual to reduce royalty payments and make more profit. A report from Stewart Rose and Anton (2023) found that incorrect royalty payments occurred due to different reasons, such as deliberate misrepresentation and human error. Lack of oversight and omission of products or sales may also be the reasons for underpaid royalties (Cislo et al., 2024). In 2024, Spotify sued because of alleged underpaid royalties (Brittain, 2024). The organization that filed the lawsuit claimed that Spotify wrongly reclassified its service without advance notice, which resulted in a significant 50% reduction in the royalties it had to pay under its license (AFP News Desk, 2024). The above facts indicate that licensees may underpay royalties to licensors for various reasons. Since similar issues can also occur in the ebook industry with the same business model, authors may gain the wrong royalties and be unable to validate their royalties. The project aims to develop a system for ebook transactions that prevents authors from receiving incorrect royalties.

1.1.3 Proposed Solution

To ensure authors receive correct royalties from the platform, the proposed system allows authors to validate their royalties by providing sufficient and reliable data so that authors are able to confirm their profits and know underpaid royalties as fast as possible if it exists.

Although the ebook transactions can be monitored and checked by a third party to increase confidence in the current ebook system, a trustworthy auditing firm is required, and costs and delays of the audit results may be incurred. In addition, the critical data and the details of the audit process are still intransparency in the view of the authors. On the other hand, the proposed system of this project integrates the technology of blockchain in order to publicize sufficient data or records for royalty validations. The idea of blockchain guarantees the information is reliable and everyone can access it, so that authors can validate their royalties by themselves.

1.1.4 Objectives And Deliverables

In this project, a blockchain-based ebook transaction system will be developed that enables authors to validate their royalties so that they can prevent royalty loss. Besides royalty validation, the system supports both ebook purchase and subscriptions to fit the current ebook industry model and adapt to the transaction-related functions of the current ebook system, such as refunds and price adjustments. It is important to maximize the compatibility of the designed system with those functions, otherwise, the system is difficult to apply in the current ebook industry. As the idea of blockchain is used in the system, privacy for different parties and costs related to blockchain are considered and discussed in the project.

In addition, to provide a clearer view of the proposed system and show that the system is implementable, the major parts of the system will be implemented and tested, including smart contracts which can be deploy to Ethereum blockchain, an Android mobile application which allows authors and readers use the services provided by the platform, and a backend server which supports the services and intercats with the blockchain. The performance related to the ebook transaction in the developed system will also be discussed in this report.

1.1.5 Outline

The remaining parts of this report are outlined as follows. In the remaining parts of section 1, critical technology and ideas for this project will be described and introduced to provide a more comprehensive background of the topic. In Section 2, the related work is mentioned as a review of current research relevant to the project topic. The details and structure of the system and each component of the system are shown in Section 3. Then, the next section will explain how the proposed system addresses the issue of the ebook industry and its limitations. Finally, Section 5 concludes the project and the report.

1.2 Related Concepts and Terms

1.2.1 Ebook Purchases

In typical ebook transactions, readers buy the desired ebooks by paying the selling price of the ebooks and receive the right to access the ebooks without a time limit. In this case, the platform can receive the payment and needs to pay the royalties based on the calculation method agreed upon by both the platform and the authors. For example, Kindle Direct Publishing (KDP) of Amazon, an e-book publishing platform, provides the calculation details as follows:

$$35\% * (List\ price - Applicable\ taxes) = Royalty$$

Equation 1 KDP ebook royalty formula for 35% royalty rate (Wogahn, 2024)

$$70\% * (List\ price - Applicable\ taxes - Delivery\ costs) = Royalty$$

Equation 2 KDP ebook royalty formula for 70% royalty rate (Wogahn, 2024)

KDP provides two equations with different royalty rates for calculating royalties. The price of ebooks and the choices of authors determine which equation is applied to calculate royalties. List price is the price set by the author. The applicable taxes may exist and affect the royalties of ebooks because some countries may charge a value-added tax or a goods and services tax. The delivery costs in **Equation 2** depend on the size of the ebook and the country. In conclusion, royalty rate, list price, ebook size, and the countries of authors and readers are the factors that determine the royalty for each transaction in KDP.

1.2.2 Subscription Schemes For Reading Ebooks

Besides ebook purchases, subscribing to a reading scheme can be another measure for readers to access ebooks (Polanka, 2013). Readers can subscribe to a plan or scheme provided by the platform monthly or yearly, and then freely access a large number of ebooks within that period. For example, Kindle Unlimited (KU) in KDP is one of the ebook subscription schemes that provides more than 4 million ebooks and allows subscribers to access 20 ebooks simultaneously (Chen, 2023). Readers who have reading habits may tend to subscribe to the service compared with purchasing the ebooks separately, as it may have Higher economic benefits. Huang et al. (2022) mentioned that subscription services produce a stable and long-term income for ebook platforms because they cultivate a group of subscribers who renew their subscriptions. Authors may join the plan with their ebooks to increase exposure and promotion of their brands. Although authors may gain a lower income after joining the plan because of the slim profit (Larson, 2022), as well as the profit distribution scheme for income allocation may be biased towards certain types and contents of ebooks (Shannon, 2015), it is still necessary to consider the market of subscription of ebook industry, as subscribers with KU have read more than 3 billion books and authors gain around 3.5 billion from this scheme (Battersby, 2024).

In the ebook subscription model, the algorithm for distributing funds to the platform and authors directly affects their income. Although the distribution algorithm is not fully transparent in KU, the main steps of the distribution can be stated as follows (Haines, 2023):

1. Determine the funds that are available in a given month as the royalty payouts. This value could be affected by the income belonging to the month and the fee taken by the platform.
2. Calculate the total number of valid pages read by the subscriber in the month, then divide the available funds by the total number of pages to determine the average return of each page.
3. Based on the value of the average return and the total number of pages read of each ebook, the royalties of ebooks can be found by multiplying them.

This pay-per-page (PPP) model eliminates the factor of the length of ebooks in the distribution, and reasonable royalties will be counted even if a subscriber did not read the whole book, so it is friendly to ebooks of different lengths. KU also applies Kindle Edition Normalized Page Count, which standardizes the format of texts, such as font and line spacing. And even non-text elements such as images and graphs will be counted (Sherrie, 2023).

1.2.3 Blockchain

Blockchain is the essential technology used in the proposed system to ensure that authors can access reliable data and validate their royalties. As the transaction records are necessary for royalty validation, the idea and characteristics of blockchain will be mentioned in this part to explain that it helps royalty validation by allowing authors to access the transaction records between the platform and readers.

In traditional transactions, people have to trust a third party, such as banks, to finish trading. In 2008, Satoshi Nakamoto introduced Bitcoin, the first proposed cryptocurrency to showcase blockchain as a decentralized infrastructure (Nakamoto, 2008, as cited in Taherdoost, 2023). Blockchain allows people to perform transactions in a peer-to-peer network without the need for trusted third parties. There are some critical characteristics of blockchain (Viriyasitavat & Hoonsopon, 2018; Xinyi et al., 2018; Ali et al., 2023):

- **Decentralization:** All transaction records are stored in the nodes of the network instead of a central server. It prevents a single node from blocking the spread of valid transactions and blocks.
- **Immutable:** Once a block of transactions is added to the blockchain, it is almost impossible to modify or hide the content of the block.
- **Pseudonymous:** The public key generated from the private key is the identity in the network. The balance of the public key can be consumed by signing a transaction with the private key.

- **Verifiable:** Digital signatures allow people to validate transactions and blocks to identity and intercept invalid data. Everyone can use the public key to verify the signature of transactions.
- **Transparent:** The information of the transactions, such as the sender, receiver, and the payment, is transparent. Otherwise, people can not validate the transaction.
- **Consistent:** Consensus, such as Proof-of-Work and the longest chain rule in Bitcoin, ensures the nodes in the network agree on the same chain of blocks.

In the case of an ebook purchase, readers can sign a transaction and deploy it to the network as a payment. After the transaction is added to the blockchain. Nodes, including the platform, can validate the transaction and the block. If it is valid, nodes can admit the new block and agree that the reader pays the cryptocurrency to the platform. The platform cannot modify or hide the transparent transactions of the blockchain. If all related transactions are finished on a blockchain, it ensures that authors can access reliable data to validate royalties.

To ensure the transaction is admitted by the majority of the network permanently, the new block of the transaction may require additional time to wait for confirmations of new blocks. In Bitcoin, confirmations provided by new blocks increase the security and finality of the block (Kaur, 2025). Royalty validation may require more information about the transactions, such as the IDs of the ebooks readers are going to purchase, which should be mentioned in the transaction. It is compatible with the idea of blockchain, blockchain can provide extra segmentation in transactions for customized additional information or smart contracts.

1.2.4 Smart Contracts And Ethereum

Smart contract is a function developed based on blockchain. Although the idea of blockchain is used for transactions originally, the idea can be used to store scripts and the interaction with the scripts, as digital signatures are also available for transactions, but also for other digital data. It may improve the management of extra information and control complicated transactions, such as refunds and royalty distributions in the ebook industry. Smart contracts are code-based, so they allow people to deploy a program to the blockchain (Zou et al., 2019). Contracts can hold variables, functions, and cryptocurrency. The codes of contracts

are immutable once they are deployed to the blockchain, while the design of the contracts is flexible, so smart contracts allow more complicated applications to make good use of blockchain. Blockchain-based smart contracts may contribute to different businesses, including supply chain, IoT, and digital right management, by providing the benefits of real-time updates, high accuracy, lower cost, and new business models (Mohanta et al., 2018).

Ethereum is one of the popular blockchain platforms that supports smart contracts. Solidity is a high-level language that can be used to write Ethereum smart contracts. According to Wang et al. (2018), the contracts can be executed after being compiled into bytecodes in a specific format which can be executed in a Turing-complete virtual machine called Ethereum Virtual Machine (EVM). Similar to transactions, the bytecodes and all interactions with the contracts should be deployed and stored in the blockchain, so all nodes can execute and verify the changes to the contracts. For example, Alice is going to pay some cryptocurrency to Bob, but she wants to limit the time allowing Bob to receive the cryptocurrency. She can deploy a contract and deposit the cryptocurrency into the contract. The contract provides a function that can be called by Bob. It allows Bob to receive the cryptocurrency if Bob deploys a transaction to call the function within the period.

1.2.5 IPFS

The InterPlanetary File System (IPFS) is an alternative way to store and share data. It is a peer-to-peer content-addressable distributed file system (Psaras & Dias, 2020) and provides benefits such as decentralized storage, elimination of single points of failure, ensured data availability, efficient storage management, content addressing, and support for large data in blockchain systems (Naz et al., 2019). These advantages help data sharing and storing, such as trust, security, and scalability, making IPFS a valuable component of the proposed platform (Kumar & Tripathi, 2020).

2 Literature Review

This section will provide a review of related work. It mainly concerns the application of blockchain in the ebook industry or other similar businesses.

Chi et al. (2020) designed a blockchain-based ebook transaction system for self-published ebook trading. The secure and reliable system is designed for self-published trading. The trade in this system only involves readers and authors. As the model does not involve any third parties, authors can take all the profits and do not need to validate their profits. However, piracy activities may not be easy to forbid as no third party can control the trading. A repository is needed to store the encrypted ebook content, and a service application is required to support the system services. An extra cost may be incurred for the services. It also mentioned that the system is available for purchase only, not for rent. Nevertheless, it shows that blockchain technology can be applied in an ebook system and provide a possible way to trade ebooks and finish payment without the participation of third parties.

Nizamuddin et al. (2019) proposed a blockchain-based framework for ebook transaction systems to guarantee that authors receive expected royalties. An Ethereum smart contract was implemented, and the framework is proposed to explain how authors, readers, and the platform interact with the contracts to complete a transaction for an ebook purchase. The provided framework supports refunds if the purchase processes fail. The contract was tested to show that it is bug-free and secure against known attacks and vulnerabilities. However, the system may be possible to be improved in terms of transaction times, cost, and simplification, as it requires multiple interactions of different parties to the contracts.

Although there may be a lack of research related to ebook subscription models with blockchain, a blockchain-based system has been designed for the pay-per-play subscription model of music streaming (Chavan et al., 2019). When users play the music, a fixed amount of tokens from a pool is given to the artist. It also supports further distribution by smart contracts. They also stated that a fixed percentage of tokens will be put into the pool when a block is mined. However, the details of the mechanism for preventing the situation of no tokens in the pool were not mentioned. Furthermore, the idea may not address the problem of

an artist playing the role of a user and playing their music repeatedly in order to gain tokens from the pool illegally.

3 Methodology

3.1 System Background

As mentioned in the previous sections, the main purpose of the proposed system is to allow authors to validate their royalties. As refunds and the ebook subscription scheme are planned to be available in the system, some assumptions and rules about them should be defined clearly. They could affect the correctness of royalty validation and further the system design.

$$\text{Royalty rate} * \text{List price} = \text{Royalty}$$

Equation 3 Royalty formula for ebook purchases

Equation 3 presents how to calculate the expected royalty if an ebook is sold. For instance, the platform gains 30% of the list price as a handling fee by setting the royalty rate to 70%. The platform and the author will obtain \$15 and \$35 respectively if the list price of the ebook is \$50. As mentioned before, the royalty may not only be affected by the rate and the price, but also the size of the ebook and the countries of the authors and readers. Although the simple formula already allows the adjustment of the profit between the platform and the authors, it will be discussed in the report later whether if complicated formula can be applied in the system.

$$\text{Royalty rate} * \text{Subscription fee} = \text{Available funds}$$

Equation 4 Fund formula for ebook subscriptions

$$\frac{\text{Available funds}}{\text{Total pages read for all books}} * \text{Total pages read for a book} = \text{Royalty}$$

Equation 5 Royalty formula for ebook subscriptions

Equations 4 and **5** describe the method to calculate the royalty for each ebook in a signal ebook subscription. Similar to ebook purchases, the platform may take a part of the

subscription fee as the handling fee, and the remaining funds are the total funds distributed to authors. The royalties for each ebook are based on the total number of pages read for all books by the subscriber and the total number of pages read for the book. For instance:

- Subscription fee = \$200
- Royalty rate = 25%
- Available funds = \$150
- 50 pages of book A read and 100 pages of book B read

Then, the author of book A gains \$50, the author of book B gains \$100, and the platform gains \$50.

It is important to note that the royalties are counted independently in different subscriptions. And the reading summaries of the subscriptions should be public for royalty validation. Secondly, the pages counted in the equation should be standardized, and the algorithm for standardization should be public, so that it produces a fairer royalty distribution algorithm to authors and a reliable method for royalty validation. For texts, the font size, line spacing, and other text formats should be unified.

Finally, we assume that only full refunds for readers are allowed in the system. The platform can decide to refund after readers apply for refunds. It will be discussed in the report later whether complicated refunds can be applied in the system.

3.2 System Overview

The system overview will be illustrated in this section. The system mainly involves the communication between the mobile app or web, the backend server of the platform, the IPFS, and the Ethereum blockchain.

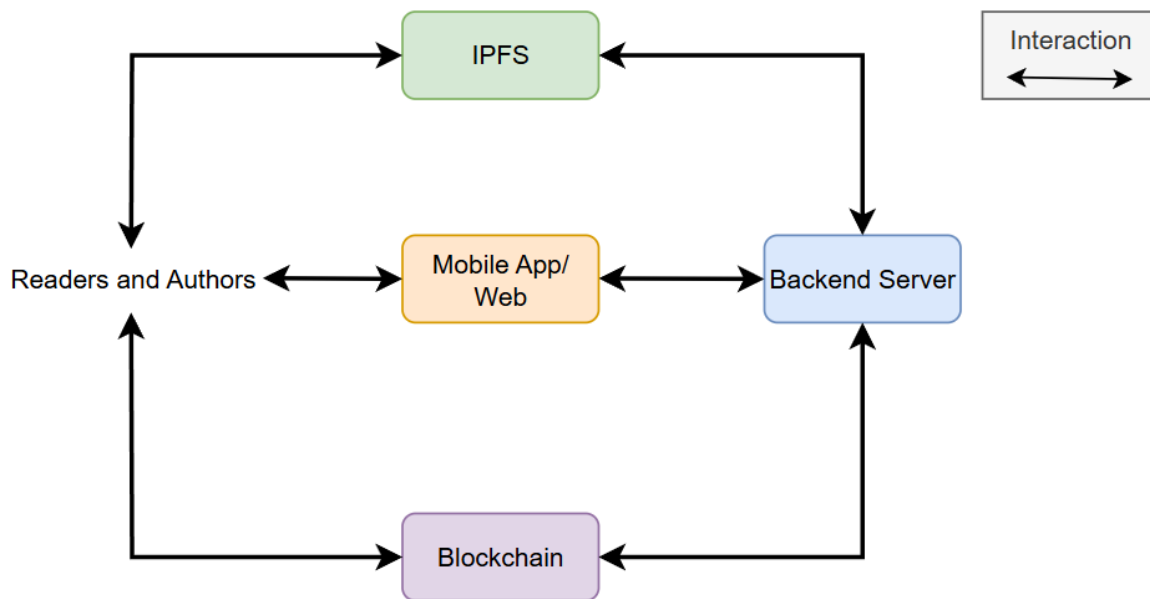


Figure 2 Components of the system and their interactions without trusted third-party APIs

The interactions between each entity are shown in **Figure 2**, and they are explained as follows:

Users and frontend:

To provide the services to readers and authors, a mobile application or website should be designed to enjoy the services of the platform. For example, authors may want to publish their ebooks and view the statistics about their ebooks on the software. Readers may want to search, preview, and read ebooks on their mobile phones or computers.

Frontend and backend:

To support the function of the mobile app or website, a backend server is needed to handle the requests from the user-end software and respond to them. The server may need a database to store the data, such as the information of users and ebooks.

Blockchain and server/users:

The server needs to keep track of the state of the related transactions and even deploy transactions for paying royalties and refunds. Authors and readers may need to interact with the blockchain to validate royalties and deploy transactions respectively.

IPFS and server/users:

Besides the related transaction information, more necessary information should be provided to authors and readers for royalty validation. Specifically, the information is the reading summaries of each subscription. Compared with storing the data in a platform server only, sharing the data on IPFS prevents the platform from modifying the data after public or rejecting access from users. While IPFS and blockchain are both decentralized, the massive data can be stored in IPFS at lower costs.

In an ideal situation, the server and authors should store the data of the blockchain and validate new blocks and transactions by following the consensus of the blockchain, as it is a node of the network. However, the costs of storing and maintaining the entire blockchain could be expensive because of the large size of the blockchain, communications with other nodes through the network, and computing power for validation. Reliable and trusted APIs provided by a third party allow users to interact with the blockchain and IPFS without handling the massive data. Users can deploy contracts and transactions, check their balance, and keep track of the events through the APIs. The interactions between each entity with the help of trusted APIs are shown in **Figure 3**.

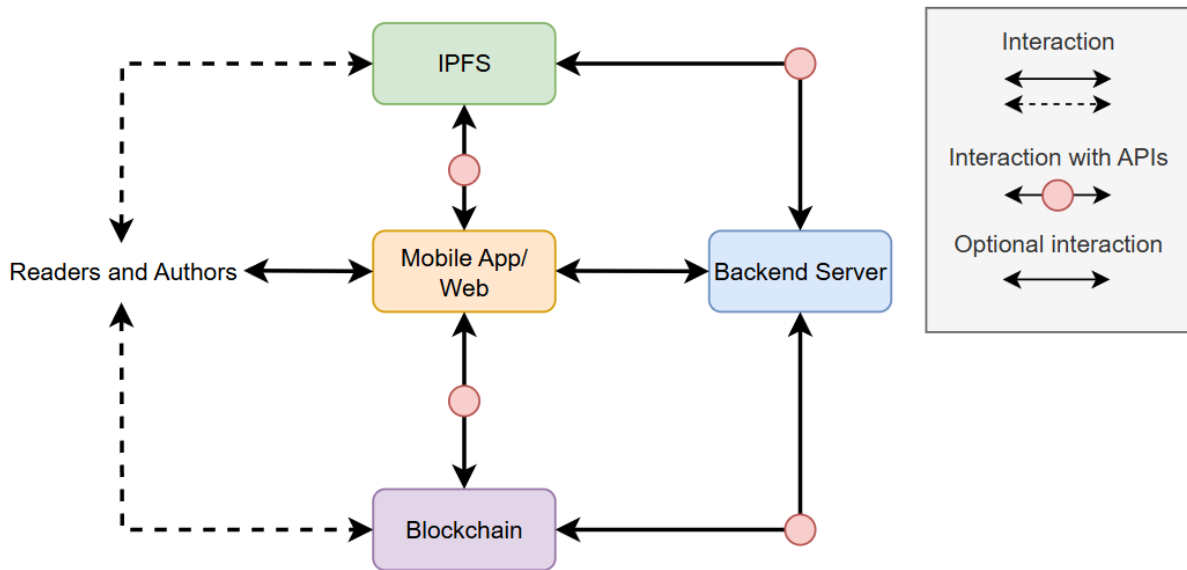


Figure 3 Components of the advanced system and their interactions with trusted third-party APIs

Even if the advanced system integrates the third-party APIs, authors can still interact with the blockchain and the IPFS with or without APIs. For further improvement of the accessibility of blockchain and IPFS, users can access the summarized and related data in the blockchain and the IPFS through the software. Nevertheless, it is possible that the platform provides wrong or fake data to users, so the mobile application and website should be open source to increase transparency and trustworthiness.

In addition, a better user experience in the process of making transactions is allowed in the advanced system. Readers can purchase ebooks by signing and deploying smart contracts through the platform software directly. The platform can help the readers deploy contracts correctly. For example, if a reader wants the purchase an ebook, the platform disallows the reader to sign up and deploy the transaction if the reader has already bought the ebook. On the other hand, the platform could generate the correct transactions and data for readers to sign, such as the payment and author account address.

3.3 Smart Contract Design

In this part, we state the design of smart contracts used in ebook purchases and subscriptions. Although readers can deploy simple transactions to the blockchain to complete the payment, there is a lack of required information for royalty validation, such as the ebook IDs in ebook purchases. Contracts also provide a better management of fund distributions and refunds.

3.3.1 Smart Contract Design For Ebook Purchases

There are a lot of ways to use smart contracts for ebook purchases. The idea of the contract design of this project is to create one contract for handling each payment, that is, one contract for purchasing one ebook, and one contract for each subscription every time. In this manner, a refund function can be provided in the contract. The function can be called by the platform to refund through the contract. Similarly, another function can be designed to distribute the cryptocurrency to the platform and authors. Therefore, the fund flows are controlled by smart contracts only, and they will be recorded in the blockchain. Authors can validate their royalties by validating the states of each contract.

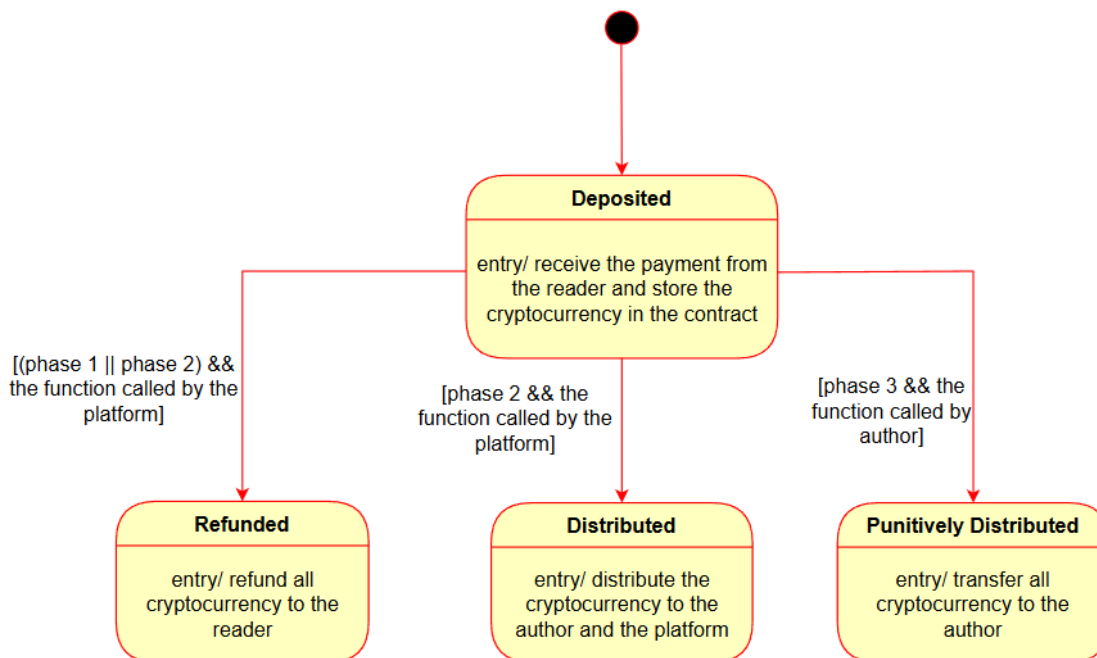


Figure 4 State diagram of smart contracts for ebook purchases

The contracts limited the interactions with it by its immutable code. **Figure 4** is a state diagram describing the state flow of the contracts. Each state in the diagram represents to different states of the funds. In the initialization of the contract, a payment will be deposited in the contract. Obviously, the funds in this state are held by the contract. Depending on further interaction, it affects the movement of the state of the contract and funds. If the platform calls a function to refund in the first phase or second phase, the state of the contract is marked as “refunded”, meaning the funds in the contract are transferred back to the reader. If the platform calls a function to distribute the funds in the second phase, the state of the contract is marked as “distributed”, meaning the funds in the contract are distributed to the platform and the author based on the royalty rate coded in the contract. If the author calls a function to distribute the funds punitively in the third phase, the state of the contract is marked as “punitively distributed”, meaning the funds in the contract are transferred to the author.

Generally, any interactions with the blockchain involving modification of the content of the contracts, but not simply accessing the data, such as calling a function to transfer the cryptocurrency from a contract, may produce fees to the caller because the new message incurs costs to the network and the nodes. Therefore, to avoid the platform refusal to allocate funds because of the high fee for interaction, a punitive function is provided allowing authors to take all the funds as royalties after a long period from the creation of the contract.

If a refund is allowed after a distribution of the fund, a full refund will result in a loss for the platform. For instance, after a fund distribution of an ebook with a price of \$100, if the royalty rate is 70%, the platform and the author earn \$30 and \$70 respectively. However, if a full refund has occurred after the distribution, the platform will lose \$70. To avoid this situation while keeping the functionality of the refund, a period (the first phase) is only for refunding. The platform can distribute funds in the second phase. The contracts accept the platform to perform a refund in the second phase if the funds have not been distributed yet, in case the platform may need more time to make the decision. **Figure 5** shows the timeline of the lifetime of each function. Readers should apply for the refund only in phase 1, because the platform may distribute the funds before the application.

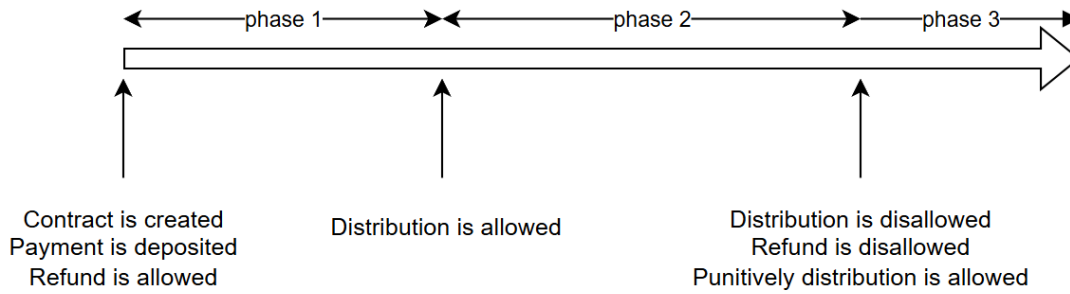


Figure 5 The timeline of contracts for ebook purchases

The proposed smart contract design does not provide a state just after “deposited” to indicate that the platform admits the contract, and the access permission is provided to the reader. The design is still reasonable because it does not affect the correctness of royalty validation. It also saves cost and time by reducing a step of blockchain interaction. In addition, the platform can notify the readers through the mobile app or email after it validates the contract.

In summary, the smart contracts ensure the authors receive the correct amount of royalties if the contracts are created correctly. The platform can make a profit only if it distributes the funds based on the royalty rate coded in the contracts. The design motivates the platform to assign correct royalties. Authors can still receive the royalties even if the platform denies assigning the funds.

3.3.2 Smart Contract Design For Ebook Subscriptions

As we mentioned, one contract will be created for each subscription. Compared to the design for ebook purchases, the contracts for subscriptions have a lower level of limitation to the platform because the platform needs more authority to assign the funds based on the reading records of the readers. Moreover, instead of holding the funds in the contracts, the funds will be transferred to the account of the platform directly to reduce the frequency and cost of interacting with the blockchain. The cost increases when complicated fund flow occurs.

Figure 6 shows the fund flows of different cases. Assume there are 10 subscribers and they read the 10 ebooks written by different authors. It is worth noting that smart contracts are still needed for storing the information about the transaction and the function for refunds.

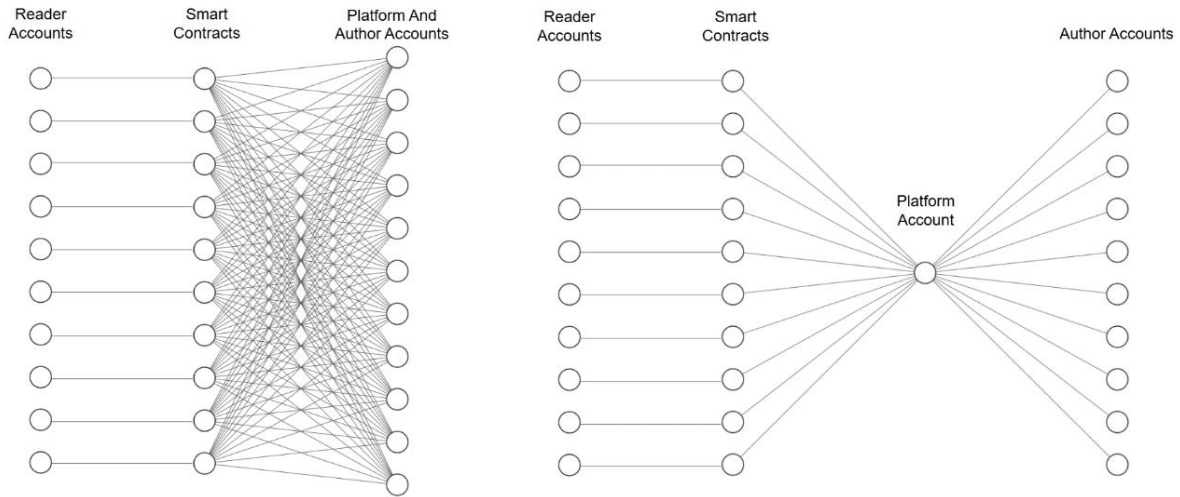


Figure 6 Fund flows in blockchain for ebook subscription without the collection of the platform account (left) and with the collection of the platform account (right)

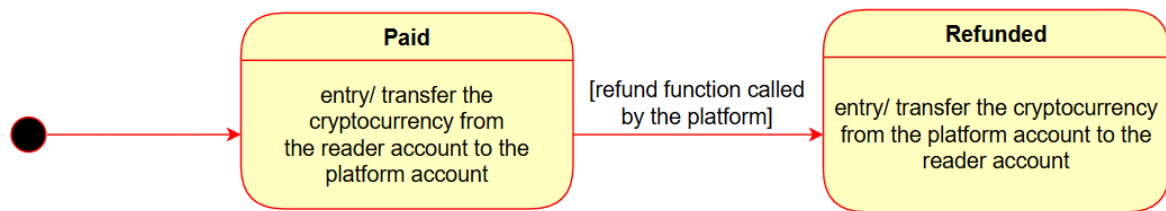


Figure 7 State diagram of smart contracts for ebook subscription

Figure 7 shows the state diagram of the contracts. The contract will directly transfer the payment to the account of the platform. If the platform wants to refund the payment to the subscriber, it can call the refund function in the smart contract, which transfers the payment from the account of the platform to the account of the subscriber. Therefore, authors can know the state of the contract to identify the validity of the subscription.

Although the contracts for subscription are relatively simple and may not be necessary, they provide critical information for royalty validation, such as the payment and its state. For other massive information, such as the reading summaries of each subscription, will be deployed to IPFS. However, everyone can upload a file to IPFS, so the platform still needs to provide a method that allows authors and readers to identify which data is published by the platform.

The platform can publicize the content identifiers CIDs of those files in the platform app or web. CIDs allow people to search and identify the files in IPFS and ensure the files have not been modified. Also, the properties of a digital signature and the immutability of blockchain smart contracts can prevent the platform from modifying the CIDs that are admitted. The platform can deploy a message using its account, which claims that the platform admits the files of the CIDs.

3.4 IPFS For Ebook Subscriptions

If the services of a subscription have expired, the reading summary that belongs to the subscription will not be changed anymore. And a reading record of the subscription can be uploaded to the IPFS for validation. **Figure 8** shows the structure of a reading record of a subscription. Each record should contain a unique ID, the payment that the subscriber has paid, the state of the contract, and the reading summary. For each ebook read, their IDs should be recorded in the summary with the number of pages of the ebook read. All information for royalty validation is included in the record. The contract state is optional. It is needed only if the system wants to allow refunds even if the subscription server has ended, and the record is deployed. The platform can upload the same record ID with the new contract state “Refunded”, meaning the royalties of the record will not be counted. However, it causes the process of royalty validation to be more complicated since authors and readers cannot ensure that the state of the contract no longer changes.

Reading Record ID	
Payment	
Contract State	
Reading Summary	
Book ID	# of pages read
Book ID	# of pages read
Book ID	# of pages read
⋮	

Figure 8 Reading record structure of a subscription

Although the record ID can be linked to the corresponding smart contract to validate the payment and the contract state, authors cannot validate the reading summary without the help of readers. It is because authors are unable to know the actual summary of each reader. Therefore, royalty validation requires the participation of readers in this system. Since the readers need to validate their summary, we further plan that readers also validate the remaining parts of their record, that is payment section and the contract state section. This idea improves reader privacy by unlinking the information about readers, such as the smart contracts, but they still can validate the information because they know what contracts they have signed.

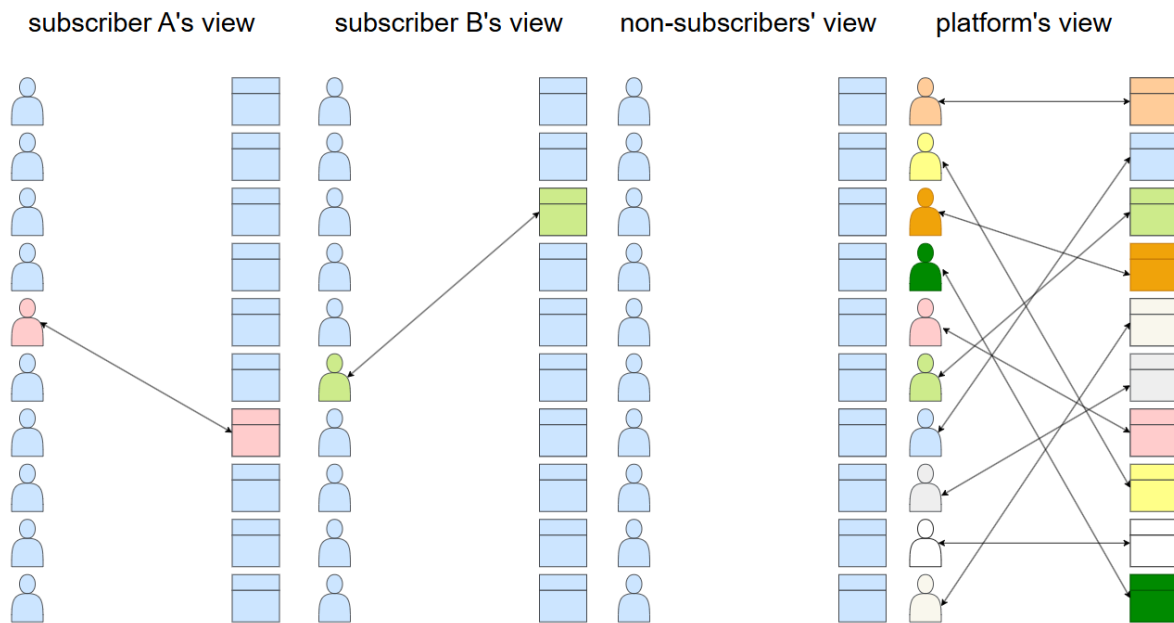


Figure 9 The views of the relationship between users and records

Figure 9 illustrates the views of the relationship between users and records of different parties. Subscribers only know which record belongs to them. The platform has a full view of the relations by accessing the related data in the database.

The above idea can be implemented by assigning a random, unique ID to each reading record and sending it to the subscriber via the platform software. Nevertheless, it provides a chance for the platform to scam. The platform can provide the same ID, which points to the same reading record, to two subscriptions if the payment and reading summary are the same. Then the platform has a chance to upload one extra fake record to earn additional profit. Although the probability of this event happening can be very low, a possible solution is provided and explained below:

$$\text{Record ID} = \text{Hash}(\text{UserID} || \text{FixedLengthNonce})$$

Equation 6 Equation for generating record ID

Equation 6 can be used to generate a reading record ID by hashing the concatenation of the user ID and a fixed-length nonce. A hash function, such as SHA-256, can be used to generate different fixed-length IDs with different variable-length inputs. It is almost impossible to find the user ID from the record ID without knowing the long-bit nonce, but it is easy to find the record ID given the user ID and the nonce, so that the reader can validate the record.

3.5 System Design

In the validation of reading records, readers are expected to validate the number of pages read of each ebook, given the ebook IDs. However, readers may not know the actual ebook IDs for each ebook, rendering the validation invalid. Therefore, accessible ebook IDs should be provided in the user interface of the mobile app or website, so that readers can validate the correctness of the reading record. Then, authors can calculate the royalties of each record and sum them up to find the total profit they should gain, assuming all records are correct.

But a malicious fraud from the platform is still possible, which is illustrated in **Figure 10**. The ID of the same ebook is different in the displays of the author and subscribers. For example, if an author publishes an ebook and the platform returns 100 as the ID of the book. On the other hand, for other users, the platform always returns 101 to the users if they want to know the ID of the ebook. By uploading a fake reading record by replacing the ebook ID

from 100 to 101, the platform can pay less royalties to the author, and the fraud will not be discovered by validation.

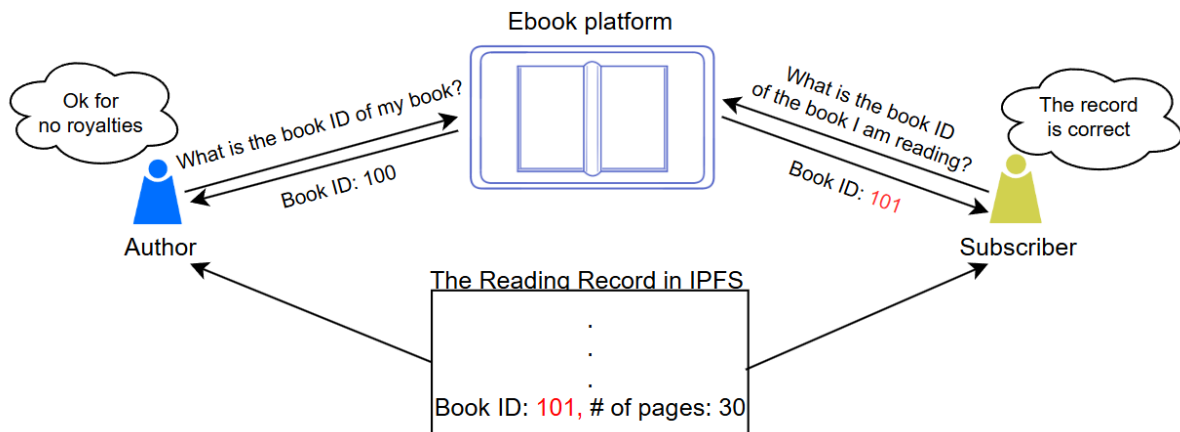


Figure 10 A potential fraud case should be prevented in the system

One implicit requirement of the fraud is that the platform needs to identify which users are not authors, otherwise, the fraud can be detected by validation. Ebook searching and previewing functions without the necessary login are possible in the ebook application and website. In this manner, it is difficult to classify the identity of the users because of the lack of information about the visitors. The ebook ID can be provided in the result of ebook searching and previewing. It is important to note that this proposed solution may not guarantee that the platform cannot guess the identity based on the data about networking and the request records, but it increases the chance that the fraud will be discovered through validation. On the side of the authors, they can check the ebook IDs displayed on the software frequently with different devices and networks in order to increase the effectiveness of validation.

As we mentioned in section 3.3.1, the smart contracts designed for ebook purchases ensure the authors gain the correct amount of royalties if the contracts are created correctly. For royalty validation, it is expected that the system should prevent a wrong contract will not signed by readers. The main processes of an ebook purchase in the proposed system are outlined as follows:

1. When a reader wants to purchase an ebook, the reader will click the purchase button in the software to create a smart contract for signing.
2. After the platform server receives the request, the server will provide the smart contract for the reader if it is available.
3. After the reader checks the content of the contract, the reader will sign the contract using the private key and deploy it to the network. Depending on the concrete implementation of the system, the reader signs and deploys the contract either via the platform software or by themselves.

The processes indicate that the reader needs to validate the contract before signing. In fact, most of the parts of the contract could be fixed based on the consensus of the platform, authors, and readers. For example, the format of the function for refunding can be standardized, and some of the code can be hard-coded in the contract, such as the distribution equation, the address of the platform account, and the period of different states. Nonetheless, some variables cannot be fixed as they are changed in different contracts:

Variables	Description
Reader Account Address	It is important because the address of the account of reader is the destination of the funds if it is refunded. Usually, it is the address of the payment account.
Payment	A wrong payment can influence the benefits of different entities.
Author Account Address	It decides the destination of the royalties after distribution.
Ebook ID	It should be included in the contract as it indicates what product the reader bought.
User ID (Optional)	It is for better management and tracking.

Royalty Rate (Optional)	It could be hard-coded in the contract directly if only one scheme of royalty rate is provided.
-------------------------	---

The fixed part of the contract can be validated by following the standard or the protocol. The reader account can be validated directly as they know their addresses. The payment is the list price of the ebook. It is reasonable that the platform will not provide a payment lower than the list price, as this would harm its benefit. The reader would not sign the contract if the payment is higher than the list price. Because of the transparency of contracts, if the platform attempts to provide a high list price to readers, the authors can discover it after the contract is deployed. As we discussed, the ebook IDs should be validated by the authors. Authors can keep monitoring the software to ensure it provides the correct ebook ID for users. The same idea can be applied to the validation of the address of author account. Therefore, the readers can assume that the address of author account and the ebook ID are correct.

3.6 Royalty Validation

Ebook purchases:

Authors could complete the following steps to finish royalty validation in ebook purchases. Authors should keep monitoring the software of the platform. If an ebook ID or the address of an author account is wrong, the author should report it to the platform. It also means that a loss of royalties may already have happened. The frequent monitoring allows authors to discover the mistakes of the platform in an early stage. In addition, authors access the blockchain to find the purchase contracts related to their ebooks. If some contracts reach state 3, which allows punitive distribution, authors can deploy a transaction to call the function and earn the royalties.

Although readers need to validate the content of the contracts before signing, it could be acceptable and reasonable because signing incorrect contracts may lead the platform to refuse to admit the contract. Also, signing a smart contract without knowing the contents would be unsafety.

Ebook subscriptions:

In ebook subscriptions, the platform can deploy the reading record of the subscriptions that have expired in the previous month, settle and pay the royalties to the authors. The CID of those files can be given to authors via the app, website, or email, and optional to sign them in the blockchain to increase the reliability. To simplify the calculations, assume refunds are not allowed after the subscriptions have expired. Authors can validate the royalties by comparing the expected royalties calculated using the records and the royalties that they received.

On the other hand, readers can find their records by the record IDs. They should check any abnormal data in the records. If it happens, meaning that the assignment of the royalties may not be correct. However, readers may not want to make the effort to validate the record because of the lack of motivation. More subscribers to validate the records increase the credibility of the records. Platform-driven reward or punishment mechanisms may encourage subscribers to validate. For example, in the platform software, answering a question about the information of the reading record correctly has a chance to obtain a discount on the next payment. But more evidences are required for this idea.

3.7 System Implementation

3.7.1 Architecture

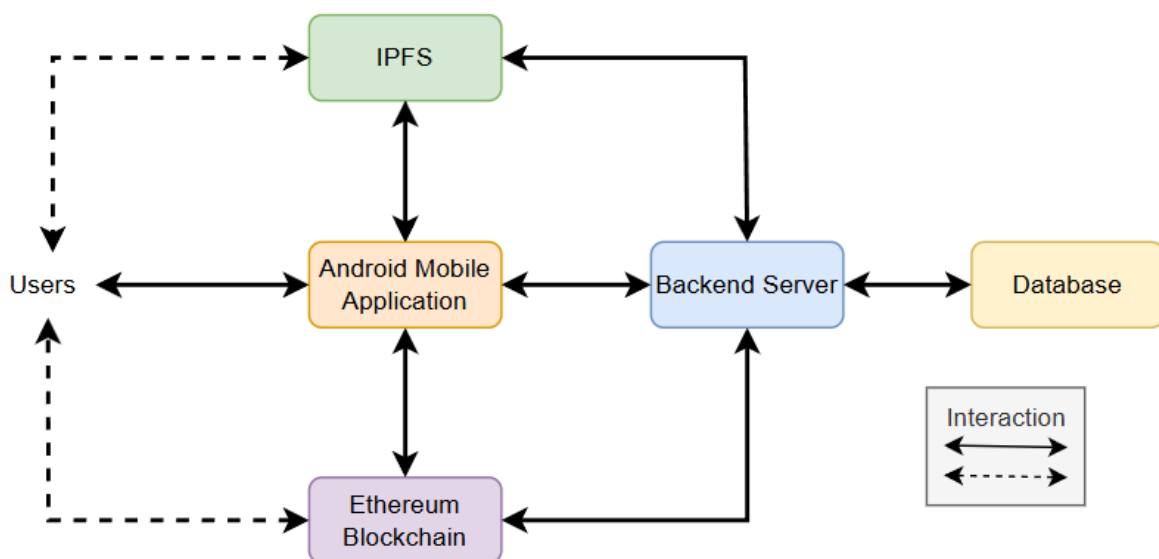


Figure 11 Components and the relationship of the implemented system

The proposed system is implemented in the project. The components and the interactions are shown in **Figure 11**. An Android mobile application is developed on Android Studio using Java. Web3j, an open-source library, is used to allow the mobile app to interact with the Ethereum blockchain. The backend server is developed with Express, Node.js, and Web3.js. Web3.js allows the server to interact with the blockchain and IPFS. In addition, both the app and server need Infura APIs to interact with the blockchain. A database is developed with MySQL.

3.7.2 Scope and Assumptions

In the implemented system, the app allows users to sign and deploy the contracts for purchases or subscriptions via the app. Some information that the users may want to view, such as the details of transactions, will be provided. They are returned from the backend server, but not the blockchain and IPFS directly. In other words, it is assumed that users access the data of blockchain and IPFS by themselves if they want to validate the reading records and contracts for royalty validation. But the validation of the contracts before signing is allowed.

For simplification, the refund function is not available in the system, but the smart contracts supports the refund function, which is tested to demonstrate that the proposed system can be implemented in the Ethereum blockchain. In addition, to avoid a complicated algorithm to handle the change of the total pages and further the counting of the pages read in subscription, it is assumed that the content of the ebook cannot be modified after publishing. Assume the royalty rate is fixed at 70%. Only text is allowed in the ebook contexts.

3.7.3 Ethereum Smart Contracts

Ethereum Smart Contracts are implemented by following the proposed contract designs, but more improvements are provided. They do not enhance the correctness of royalty validation, but the user experience for customers. As we mentioned, in the case of ebook purchases, most parts of the contracts are the same, including the royalty rate (by assumption) and the equation of the royalty distribution. The platform can deploy the framework for purchase contracts that contain the fixed parts, it acts as an official format to buy an ebook. In this

manner, buyers only need to validate the framework once and remember the address of the framework contract to perform multiple purchases. Therefore, only the framework contract address and a few variables need to be validated after the first purchase. Moreover, the address of the framework contract is a valuable parameter for filtering other unrelated transactions when authors want to track the contracts.

In Ethereum, a transaction is any action that modifies the state of the blockchain, such as transferring cryptocurrency, calling a function in a smart contract, or deploying a new smart contract. A fee is needed to modify the state of a blockchain because it incentivizes miners or validators to process and include transactions in the blockchain. It is reasonable that the validators or miners prioritize adding transactions with high fees to increase profits. In the case of Ethereum, a transaction with a low fee may be added to the chain after a few days or weeks of deployment. It may not be expected for the subscribers. To avoid the situation, the framework contract accepts an input of a deadline. The contract will not create a purchase contract if the transaction has expired. See **Appendix A** for the pseudo-code of the framework contract for ebook purchase. Note that the input parameters of the contract are payment, book ID, user ID, and author account address. The address of the reader account is the address that deploys the transaction. There is nothing new in the framework contract for the ebook subscriptions. See **Appendix B** for the pseudo-code.

3.7.4 Transaction Validation

In the implemented system, to validate the ebook ID and author account address, the application provides:

1. **SearchByID(ID)**: A function that allows users to input an ebook ID, the app will make a request to the server with the ebook ID only. The server should return a bunch of information about an ebook, including the author account address.
2. **SearchByPrompt(Prompt)**: Another function that allows users to input a text prompt, the app will make a request to the server with the prompt only. The server should return a bunch of ebooks with their IDs and author account addresses. The returned ebooks should be relevant to the prompt, like the common search algorithm of ebook platforms.

After an author uploads an account address ADD_A and an ebook to the platform, the author will receive an ebook ID_A for the ebook. Then, the author uses different prompts, such as a section of the book title and the author's name, to search the uploaded ebook:

Case 1: If the uploaded ebook is not included in the result, report the case to the platform to identify the problem. Validation fails in this case.

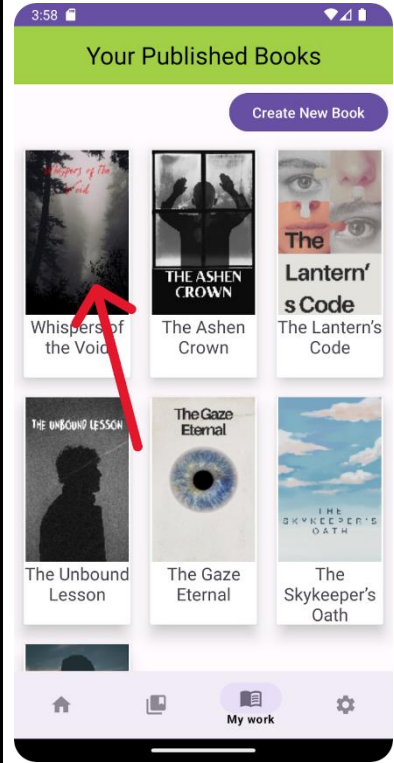
Case 2: If two or more ebooks match the uploaded ebook, but they do not have the same author address and ebook ID, report the case to the platform. Validation fails, and royalties may be lost in this case.

Case 3: If only one ebook matches the uploaded ebook, record the ebook ID_B and the author address ADD_B of the searched ebook.

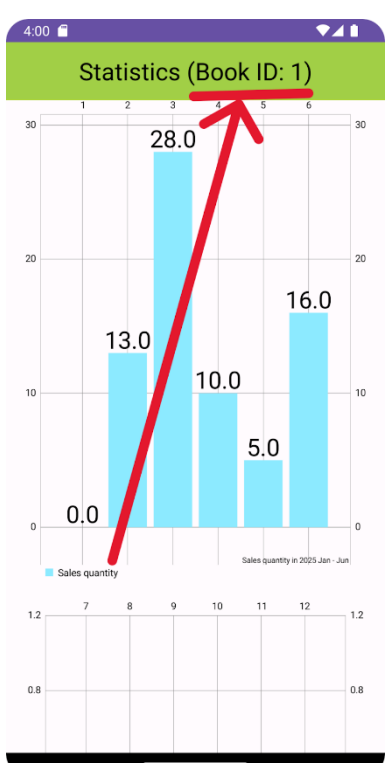
Continue to case 3, search the uploaded ebook using ID_B , and record the ebook ID_C and the author address ADD_C of the searched ebook. If $ADD_A = ADD_B = ADD_C$ and $ID_A = ID_B = ID_C$, the test of the validation passes on the author side. Otherwise, there may be a risk of missing royalties.

When a reader wants to purchase an ebook with author's address ADD_D and the ebook ID_D , the reader should search the ebook by ID_D and obtain ID_E and ADD_E from the result. If $ADD_D = ADD_E$, $ID_D = ID_E$, and no other distinct information about the two ebooks, then the reader can sign the transaction with the address, ID, and other correct inputs. **Figure 12** shows some screenshots of the app and the descriptions for transaction validation.

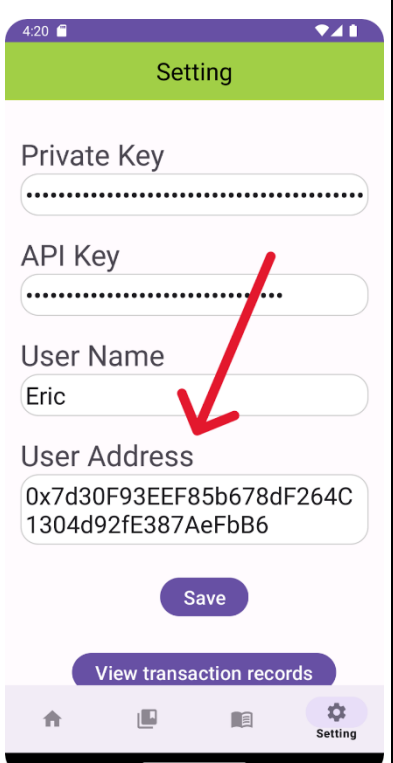
“My work” shows all the published ebooks of the user



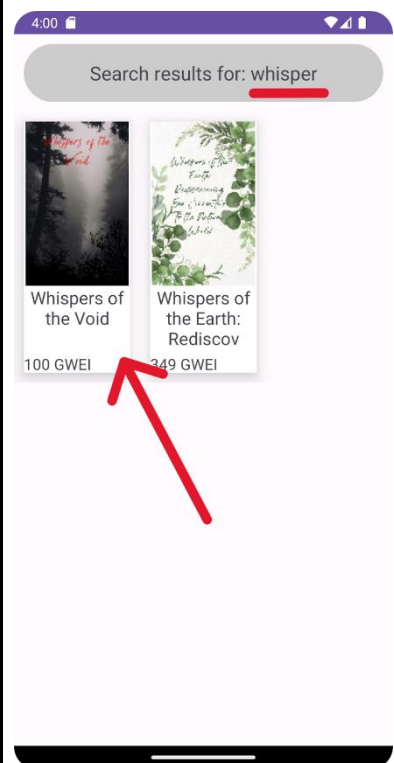
Click the ebook to view the ebook ID



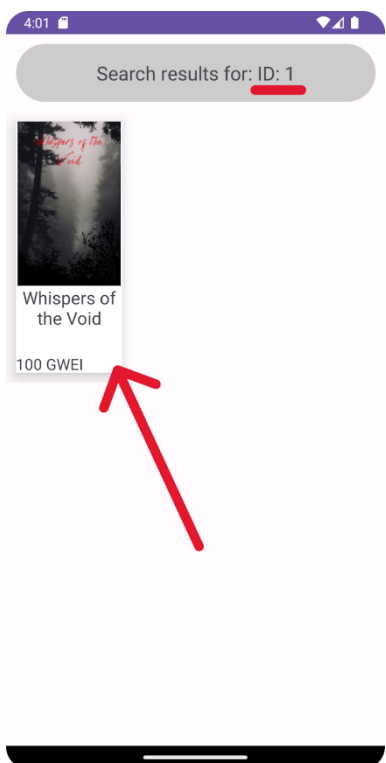
View the address of the user for receiving royalties



Search by prompt



Search by ID



All data should be identical, obtained from any place

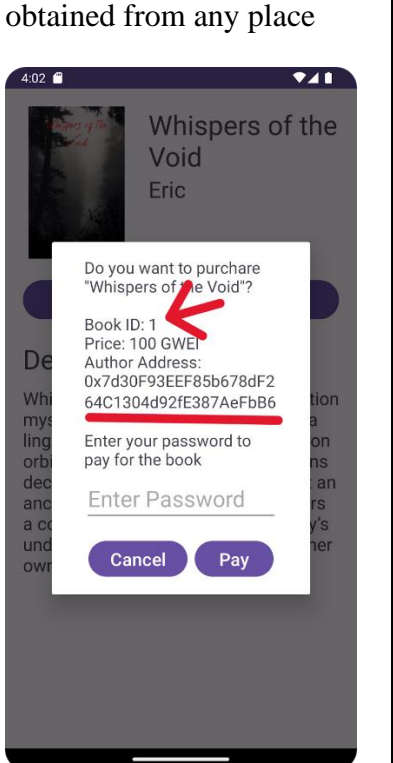


Figure 12 Screenshots of the app in transaction validation

In the implemented system, the contract addresses of the frameworks are hard-coded in the app as they are not expected to change. In addition, the app is open-source, so it allows users to ensure that the inputs being used to sign are displayed on the screen correctly. As the app allows readers to sign and deploy contracts directly, it allows provide a method for them to validate that the private key is stored safely in the local storage and will be used correctly.

3.7.5 Use cases

In this section, the use cases of the system will be described. **Figure 13** is a use case diagram showing what the readers and authors want to do with the system. The implemented system does not provide any functionality for administrators to manage the system.

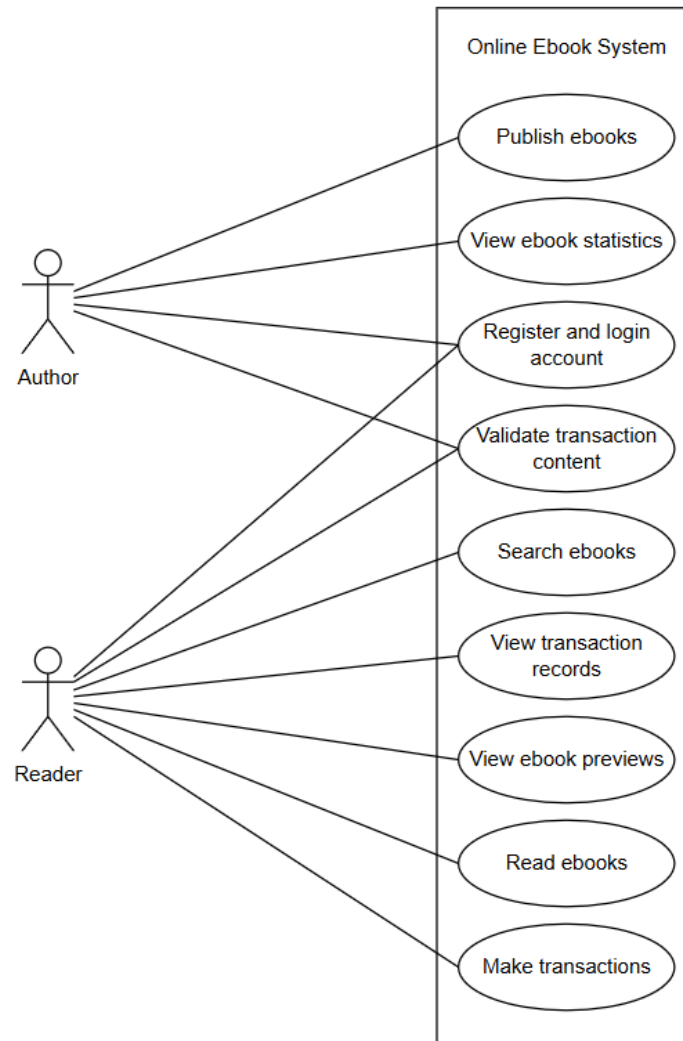


Figure 13 Use case diagram

Publish ebooks:

Authors can seamlessly publish their ebooks on the platform by submitting essential materials and data. This includes uploading a PNG image to serve as the ebook cover, providing a clear and engaging ebook title, writing a concise description to attract potential readers, supplying a text file formatted according to the platform's fixed structure for the ebook's contents, and setting a list price to determine its market value.

View ebook statistics:

The platform enables authors to monitor the performance of their ebooks through detailed statistics. For instance, authors can access data on sales volume for each month, allowing

them to analyze trends, assess market reception, and make informed decisions about their publishing strategies.

Register and login account:

To ensure secure access, the platform supports user account registration and login. During registration, a verification code is sent via email to confirm the user's identity, providing a straightforward and secure process to create and access an account for managing ebook-related activities.

Validate transaction content:

As outlined in previous discussions, the platform includes a mechanism to validate transaction content. This process ensures that all transaction details are accurate and secure.

Search ebooks:

Users can efficiently locate ebooks on the platform through a search functionality that supports queries by ebook ID or specific prompts. This feature simplifies the process of finding relevant ebooks, enhancing user experience and accessibility.

View transaction records:

The platform allows users to review their transaction history, providing visibility into the status of each transaction. Users can check whether a transaction is pending, activating, refunded, or expired, ensuring transparency and control over their financial activities.

View ebook previews:

To aid in purchasing decisions, the platform offers ebook previews that display key details, including the cover image, book title, author name, a brief description, and the book's price. This feature helps users evaluate ebooks before committing to a purchase.

Read ebooks:

The ebook reading experience is designed to be user-friendly, offering intuitive navigation. Users can move to the next or previous page with a single click or directly enter a page number to jump to a specific section, ensuring a smooth and enjoyable reading process.

Make transactions:

The platform streamlines the transaction process to require only a few steps, prioritizing both efficiency and security. Before finalizing a transaction, users must enter their account password to sign the transaction using a private key stored locally, adding a layer of protection to financial interactions. The app is able to get the current level of the gas price and provide a reasonable gas price for the transaction.

3.7.6 Database

Figure 14 shows the ER diagram of the database. Key features of the system design related to the database:

- After login, a session token is sent to the client and included in subsequent requests to verify, excluding the search functions.
- The critical information about the transactions is stored in the database for handling the relevant requests and preventing third-party apps from failing to connect with the blockchain.
- After receiving a new ebook, the content of the ebook will be divided into chapters and pages based on the format of the file and the standard respectively.

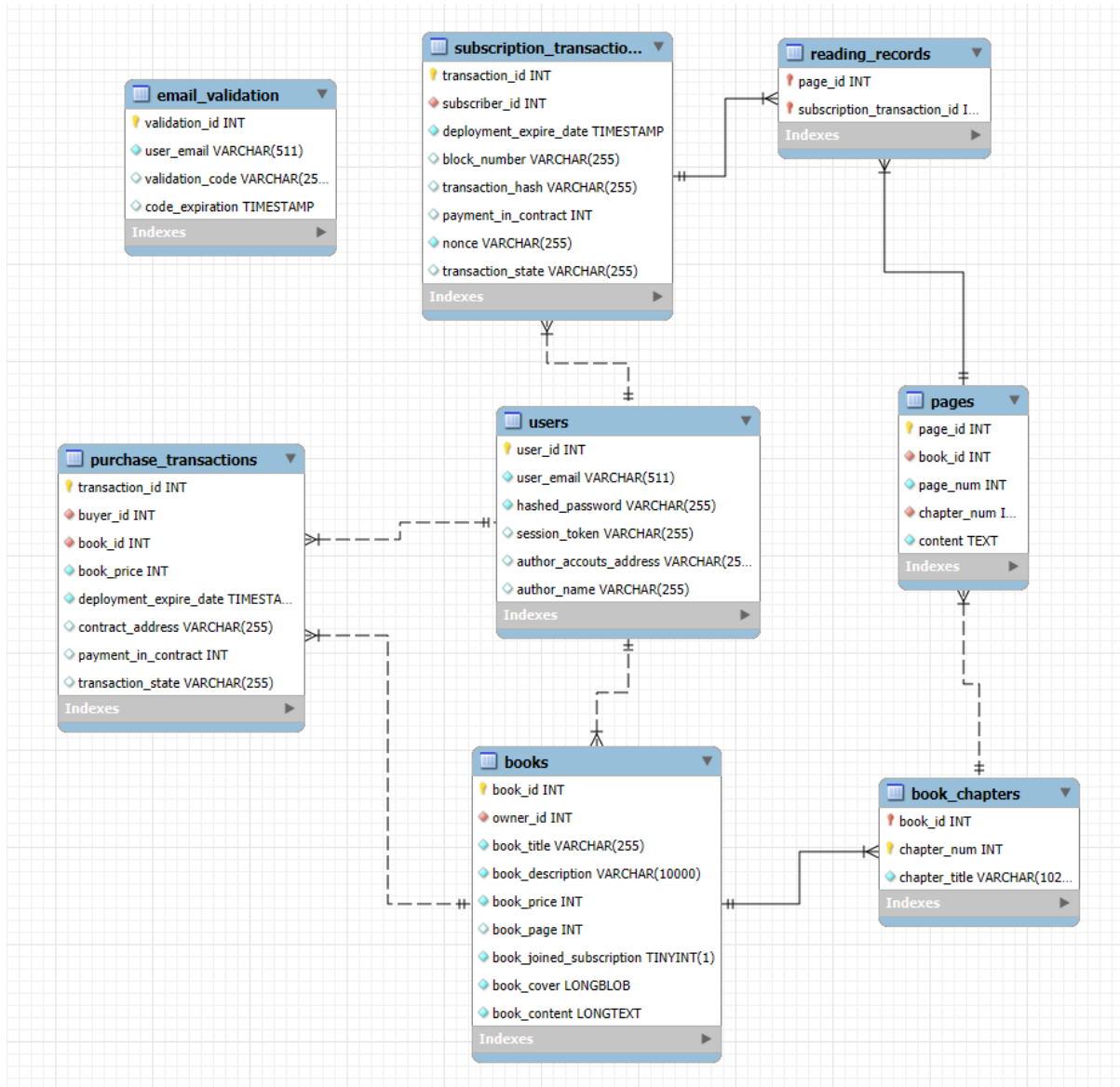


Figure 14 ER diagram of the backend database

4 Discussion

In this section, we elaborate and analyze the performance of the proposed blockchain-based ebook transaction system, focusing on transaction time, throughput and costs, privacy, scalability and flexibility, correctness of royalty validation, and limitations of Ethereum.

Correctness of Royalty Validation of the Proposed System

The proposed system ensures accurate royalty validation by leveraging blockchain transparency, smart contract automation, and decentralized storage via IPFS. For ebook purchases, smart contracts automatically distribute funds based on predefined royalty rates, with transaction states ("deposited," "refunded," "distributed," or "punitively distributed") permanently recorded on the blockchain. This allows authors to independently verify transaction statuses and royalty payments. A punitive distribution feature motivates timely fund allocation by the platform, enabling authors to claim full payments if delays occur.

In the subscription model, IPFS stores reading summaries linked to hashed record IDs, balancing privacy and validation. Authors can compute expected royalties from these records and compare them to received payments. However, this process depends on readers validating their reading records, introducing a reliance on reader participation that could be enhanced with platform incentives, though their impact needs further study. Overall, the integration of blockchain and IPFS provides authors with trustworthy, tamper-proof data to confirm royalties, effectively tackling underpayment issues, provided smart contracts and reading records are accurately implemented.

Limitations of Ethereum

A key drawback of the Ethereum-based system is that readers must pay gas fees to deploy transactions for ebook purchases or subscriptions. Unlike traditional ebook transactions, where only the list price is paid, these additional fees, unpredictable due to network congestion, may discourage users. In Ethereum, gas fees compensate miners for processing transactions and must be paid by the transaction initiator (the reader) using cryptocurrency from their own address. This design prevents the platform from directly absorbing the fee, as only the caller's address can cover the cost, and readers must explicitly authorize each transaction on the blockchain.

While the platform could reimburse readers, this would add complexity and potential delays. Ethereum lacks native support for mechanisms like multi-signature transactions, where one party signs and another pays the fee, limiting fee delegation options. However, other blockchains offer alternatives, such as meta-transactions or dual-signature systems, allowing the platform to cover fees on behalf of readers. Adopting such a blockchain could enhance user experience but would require redesigning the system to ensure compatibility and maintain security and decentralization.

Transaction Time

The transaction time, defined as the duration from user payment confirmation to granting access to the ebook, is a key determinant of user experience in the proposed system. Testing conducted on the Ethereum Sepolia testnet demonstrated that this process takes less than 30 seconds when utilizing the gas price recommended by the application, which adjusts based on current network conditions. This duration is deemed acceptable for ebook transactions, as it meets typical user expectations for rapid access to digital content. However, the primary bottleneck is Ethereum's block time, averaging approximately 12 seconds per block. This interval dictates how frequently new transactions are added to the blockchain, inherently limiting the speed of transaction processing.

While additional block confirmations are often required to ensure transaction security and finality, particularly in high-value scenarios, the system can tolerate providing ebook access prior to full confirmation. This approach is viable due to two factors: (1) ebook purchases are small-amount transactions, reducing the financial risk of reversal, and (2) the system retains the ability to revoke reading rights if a transaction becomes invalid, such as during a blockchain reorganization. Although such invalidations are rare, this design choice effectively minimizes transaction time, enhancing user satisfaction without significantly compromising security.

Throughput and Costs

The throughput of the system, or its capacity to process multiple transactions concurrently, is sufficient for the current scope of ebook transactions, which do not demand the high-frequency processing seen in domains like high-frequency trading (HFT). Unlike HFT, where thousands of transactions per second are common, ebook purchases and subscriptions occur at a lower rate, suggesting that the system could handle its present load. However, as the platform scales with increased user adoption, the volume of transactions may approach the limits of Ethereum's throughput, constrained by its block size and 12-second block time. Further research is needed to determine whether ebook transaction demands could overload the blockchain under peak conditions.

Each transaction on Ethereum incurs gas fees, payable to miners for processing, which vary with network congestion. These costs could rise significantly during periods of high demand, impacting users and the platform's operational expenses. One potential mitigation is the

adoption of permissioned blockchains, which typically offer higher throughput and lower costs compared to public blockchains like Ethereum or Bitcoin. However, this solution poses a challenge to the correctness of royalty validation. Permissioned blockchains, being controlled by specific entities, lack the full decentralization and transparency of public blockchains, potentially enabling manipulation of transaction data and undermining the trustless verification that authors rely upon for accurate royalty payments.

Privacy

Privacy considerations arise from the transparent nature of blockchain technology employed in the system. For ebook purchases, a reader's address is linked to the purchased ebook via the smart contract. If an external party knows the owner of that address, they can deduce the reader's book choices, potentially exposing personal reading preferences. In contrast, the subscription model offers improved privacy for reading records. These records are stored on IPFS with unique hashed IDs, ensuring that only the reader knows which record corresponds to their activity, thus shielding individual reading histories from public view.

Additionally, the system's design makes sales figures for each book, as well as the profits of authors and the platform, publicly accessible on the blockchain. While this transparency supports royalty validation by allowing authors to verify earnings, it may be undesirable for authors and the platform who prefer to keep financial details confidential. Balancing transparency for validation with privacy protection remains a notable challenge in the system's implementation.

Scalability and Flexibility

The proposed system demonstrates flexibility in supporting complex royalty distribution models. For instance, in ebook purchases, the smart contracts can incorporate factors such as the country of readers and authors into the royalty distribution rules. This adaptability allows for tailored royalty calculations, potentially reflecting regional pricing or tax variations, but it requires including additional user data in the contracts, which could leak further information about readers and authors. Such enhancements are feasible because the royalty distribution logic is coded within the immutable smart contracts, ensuring consistency and verifiability.

Changes to distribution rules are also accommodated by deploying new framework contracts or modifying existing ones, provided the original contracts are designed to permit updates.

This flexibility enables the system to evolve with new business requirements or market conditions. However, scalability, the system's ability to handle growing numbers of users and transactions, is constrained by the underlying blockchain's limitations, particularly Ethereum's throughput. As discussed, increased transaction volumes could exceed current capacity, necessitating careful planning to ensure the system remains performant as adoption grows.

5 Conclusion

This report details the development and implementation of a blockchain-based ebook transaction system aimed at resolving royalty validation challenges within the ebook industry. Leveraging Ethereum smart contracts and the InterPlanetary File System (IPFS), the system provides a decentralized solution that enhances transparency and empowers authors to independently verify their royalty payments. The prototype includes an Android mobile application and a backend server, successfully demonstrating support for ebook purchases and subscriptions. For purchases, smart contracts automate royalty distribution with predefined rates and punitive measures to ensure timely payments. In the subscription model, IPFS-stored reading records enable royalty calculations, though validation hinges on reader participation. The immutability and accessibility of blockchain data eliminate reliance on centralized intermediaries, offering a robust mechanism to combat underpayment.

The system's design and underlying principles extend beyond ebooks, offering potential applications in similar digital content industries such as music and video streaming. These sectors face comparable issues with opaque royalty distribution and underpayment risks, where adopting this decentralized approach, through tailored smart contracts and IPFS integration, could enhance trust and accountability. This adaptability positions the project as a valuable framework for improving creator compensation across digital platforms.

Despite its strengths, the system has limitations. Ethereum's gas fees, borne by readers, add costs not present in traditional ebook transactions, potentially reducing user adoption. The subscription model's dependence on reader validation introduces variability, as inconsistent engagement could undermine reliability. Additionally, the public nature of blockchain data

raises privacy concerns, exposing reader preferences, author earnings, and platform finances. Future research could address these by exploring blockchains with lower fees or fee delegation, implementing privacy-enhancing techniques like developing incentives to boost reader participation. Such advancements would refine the system and broaden its impact, fostering a fairer digital content ecosystem.

References

- AFP News Desk. (2024, May 20). Spotify sued over alleged unpaid royalties. *The Jakarta Post*. <https://www.thejakartapost.com/culture/2024/05/20/spotify-sued-over-alleged-unpaid-royalties.html>
- Ali, V., Norman, A. A., & Azzuhri, S. R. B. (2023). Characteristics of blockchain and its relationship with trust. *IEEE Access*, *11*, 15364–15374. <https://doi.org/10.1109/access.2023.3243700>
- Battersby, M. (2024, September 5). *Amazon Kindle Unlimited reveals most-read books of the past decade*. Retrieved April 17, 2025, from <https://www.thebookseller.com/news/amazon-kindle-unlimited-reveals-most-read-books-of-the-past-decade>
- Brittain, B. (2024, May 17). Spotify sued over millions in allegedly unpaid music royalties. *Reuters*. <https://www.reuters.com/legal/litigation/spotify-sued-over-millions-allegedly-unpaid-music-royalties-2024-05-17/>
- Chavan, S., Warke, P., Ghuge, S., & Deolekar, R. V. (2019). Music Streaming Application using Blockchain. *International Conference on Computing for Sustainable Global Development*, 1035–1040. <https://ieeexplore.ieee.org/document/8991275>
- Chen, C. (2023, November 21). *Answers to all your questions about the Kindle Unlimited Reading subscription*. Retrieved April 17, 2025, from <https://www.aboutamazon.com/news/devices/what-is-kindle-unlimited>
- Chi, J., Lee, J., Kim, N., Choi, J., & Park, S. (2020). Secure and reliable blockchain-based eBook transaction system for self-published eBook trading. *PLoS ONE*, *15*(2), e0228418. <https://doi.org/10.1371/journal.pone.0228418>

- Cislo, R., Conforti, E., & McHale-Adams, M. (2024, January 31). *Royalty audits & licensing agreements: Ensure accurate reporting & payments*. Retrieved April 17, 2025, from <https://www.plantemoran.com/explore-our-thinking/insight/2024/01/royalty-audits-and-licensing-agreement>
- Errera, R. (2024, November 12). Printed Books vs eBooks Statistics, Trends and Facts [2025]. *Toner Buzz*. Retrieved April 16, 2025, from <https://www.tonerbuzz.com/blog/paper-books-vs-ebooks-statistics/>
- Haines, D. (2023, August 28). *Does Kindle Unlimited Pay per page pay authors fairly?* Retrieved April 18, 2025, from <https://justpublishingadvice.com/is-kindle-unlimited-pay-per-page-read-fair-for-authors/>
- Hill, J. (2021, June 8). *The most popular Self-Publishing Platforms: Pros & Cons*. Retrieved April 17, 2025, from <https://www.launchmybook.com/the-most-popular-self-publishing-platforms-pros-cons/>
- Huang, D., Liu, L., Deng, Y., & Chen, C. (2022). A Digital Media Subscription Management System Combined with Blockchain and Proxy Re-Encryption Mechanisms. *Symmetry*, 14(10), 2167. <https://doi.org/10.3390/sym14102167>
- Kaur, G. (2025, February 12). *Bitcoin transaction confirmations: What to know about receiving BTC*. Retrieved April 18, 2025, from <https://cointelegraph.com/learn/articles/bitcoin-transaction-confirmations>
- Kumar, R., & Tripathi, R. (2020). Blockchain-Based framework for data storage in Peer-to-Peer scheme using interplanetary file system. In *Elsevier eBooks* (pp. 35–59). <https://doi.org/10.1016/b978-0-12-819816-2.00002-2>
- Larson, C. (2022). Streaming books: confluencers, Kindle Unlimited and the platform imaginary. *Communication Culture and Critique*, 15(4), 520–530. <https://doi.org/10.1093/ccc/tcac032>
- Mohanta, B. K., Panda, S. S., & Jena, D. (2018). An overview of smart contract and use cases in blockchain technology. *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–4. <https://doi.org/10.1109/icccnt.2018.8494045>

- Naz, M., Al-Zahrani, F. A., Khalid, R., Javaid, N., Qamar, A. M., Afzal, M. K., & Shafiq, M. (2019). A secure data sharing platform using blockchain and interplanetary file system. *Sustainability*, 11(24), 7054. <https://doi.org/10.3390/su11247054>
- Nizamuddin, N., Hasan, H., Salah, K., & Iqbal, R. (2019). Blockchain-Based Framework for protecting author royalty of Digital Assets. *Arabian Journal for Science and Engineering*, 44(4), 3849–3866. <https://doi.org/10.1007/s13369-018-03715-4>
- Polanka, S. (2013). Ebook Access: Business models for subscription services. *Online Searcher*, 37(2), 65–67. https://corescholar.libraries.wright.edu/cgi/viewcontent.cgi?article=1127&context=ul_public
- Psaras, Y., & Dias, D. (2020). The InterPlanetary file System and the FileCoin network. *IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume*. <https://doi.org/10.1109/dsn-s50200.2020.00043>
- Revankar, S. (2025, February 12). *eBooks Statistics By Revenue, User, Country, Sales, Genre and Facts* (R. Jambhale, Ed.). Retrieved April 16, 2025, from <https://www.coollest-gadgets.com/ebooks-statistics/>
- Shannon, S. (2015, June 25). *Amazon's "pay-per-page" plan could alter writing as well as royalties*. Retrieved April 17, 2025, from <https://www.theguardian.com/books/booksblog/2015/jun/25/amazons-pay-per-page-plan-writing-royalties-ebook-kindle>
- Sherris, J. (2023, October 9). *How does Kindle Unlimited work for authors?* Retrieved April 18, 2025, from <https://printbindship.com/how-does-kindle-unlimited-work-for-authors/>
- Staiger, J. (2012). How e-books are used. *Reference & User Services Quarterly*, 51(4), 355–365. <https://doi.org/10.5860/rusq.51n4.355>
- Stewart Rose, D., & Anton, K. (2023). *Is my licensee in compliance? Tips to keep them on track*. InvotexIP. Retrieved April 17, 2025, from <https://static1.squarespace.com/static/59ef86992ae8ba5803ae8cc43/t/6481eb8458c2117ee56b8eab/1686236036551/ARTICLE+-+Is+My+Licensee+in+Compliance+Feb+2023.pdf>
- Taherdoost, H. (2023). Smart contracts in Blockchain Technology: A critical review. *Information*, 14(2). <https://doi.org/10.3390/info14020117>

- Viriyasitavat, W., & Hoonsoapon, D. (2018). Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration*, 13, 32–39. <https://doi.org/10.1016/j.jii.2018.07.004>
- Wang, S., Yuan, Y., Wang, X., Li, J., Qin, R., & Wang, F. (2018). An overview of Smart Contract: architecture, applications, and future trends. *2022 IEEE Intelligent Vehicles Symposium (IV)*, 108–113. <https://doi.org/10.1109/ivs.2018.8500488>
- Wogahn, D. (2024, May 6). *Back to Learning Center The 2024 guide to Amazon fees and royalties for Kindle eBooks and KDP Print*. Retrieved April 17, 2025, from <https://www.authorimprints.com/amazon-kdp-royalty-pricing/>
- Xinyi, Y., Yi, Z., & He, Y. (2018). Technical characteristics and model of blockchain. *International Conference on Communication Software and Networks*, 562–566. <https://doi.org/10.1109/iccsn.2018.8488289>
- Yoo, S. (2024, February 19). *The Pros and Cons of Self-Publishing on Amazon*. Retrieved April 17, 2025, from <https://publishdrive.com/the-pros-and-cons-of-self-publishing-on-amazon.html>
- Zou, W., Lo, D., Kochhar, P. S., Le, X. D., Xia, X., Feng, Y., Chen, Z., & Xu, B. (2019). Smart contract development: challenges and opportunities. *IEEE Transactions on Software Engineering*, 47(10), 2084–2106. <https://doi.org/10.1109/tse.2019.2942301>

Appendices

Appendix A: pseudo-code of the framework contract for ebook purchases

```

Contract PurchaseContract {
    // State variables
    constant PLATFORM_ADDRESS = fixed platform address
    (0xdBB7747B192B9aF10ce98eFa34F5563fe0E20b07)
    readerAccountAddress = address of the reader who creates the contract
    authorAccountAddress = address of the author receiving royalties
    purchaseTimestamp = timestamp of contract creation
    isInteracted = false (indicates if funds are still in contract)
    balance = payment received during creation (in Ether)

    // Event to log actions (for transparency and royalty validation)
    event ActionExecuted(action: string)

    // Constructor: Initialize contract with payment and author address
    constructor(authorAccountAddress, payment) {
        readerAccountAddress = caller address (reader)
        authorAccountAddress = authorAccountAddress
        purchaseTimestamp = current blockchain timestamp
        balance = payment (sent with transaction)
        // Note: Gas costs incurred for deployment
    }

    // Distribute funds: Split payment between platform (30%) and author (70%)
    function distribute() {
        require caller is PLATFORM_ADDRESS
        require isInteracted is false
        require current timestamp is between (purchaseTimestamp + 10 days) and
        (purchaseTimestamp + 20 days)
    }
}

```

```

        isInteracted = true
        platformShare = balance * 30 / 100
        authorShare = balance - platformShare
        transfer platformShare to PLATFORM_ADDRESS
        transfer authorShare to authorAccountAddress
        emit ActionExecuted("FundsDistributed")
        // Note: Gas costs incurred for transfers and execution
    }

    // Refund: Return funds to reader if platform approves
    function refund() {
        require caller is PLATFORM_ADDRESS
        require isInteracted is false
        require current timestamp is less than purchaseTimestamp + 20 days

        isInteracted = true
        transfer balance to readerAccountAddress
        emit ActionExecuted("Refunded")
        // Note: Gas costs incurred for transfer and execution
    }

    // Penalize: Transfer funds to author if platform fails to act
    function penalize() {
        require caller is readerAccountAddress
        require isInteracted is false
        require current timestamp is greater than purchaseTimestamp + 20 days

        isInteracted = true
        transfer balance to authorAccountAddress
        emit ActionExecuted("Penalized")
        // Note: Gas costs incurred for transfer and execution
    }
}

```

```
Contract PurchaseContractFactory {
    // Event to log contract creation (for transparency and tracking)
    event PurchaseContractCreated(
        purchaseContractAddress: address,
        bookId: integer,
        userId: integer,
        authorAddress: address,
        payment: integer
    )

    // Create a new PurchaseContract for an ebook purchase
    function createPurchaseContract(creationDeadline, bookId, userId,
authorAccountAddress, payment) {
        require current timestamp <= creationDeadline
        require payment > 0

        newContract = deploy new PurchaseContract with authorAccountAddress and send
payment to it
        emit PurchaseContractCreated(newContract address, bookId, userId,
authorAccountAddress, payment)
        // Note: Gas costs incurred for contract deployment
    }
}
```

Appendix B: pseudo-code of the framework contract for ebook subscriptions

```

Contract SubscriptionContract {
  // State variables
  constant PLATFORM_ADDRESS = 0xdBB7747B192B9aF10ce98eFa34F5563fe0E20b07
  readerAccountAddress = address of the creator (reader)
  purchaseTimestamp = timestamp of contract creation

  // Events
  event PaymentTransferred()
  event RefundExecuted(refundValue)

  // Constructor: Transfer payment directly to platform
  constructor(payment) {
    readerAccountAddress = caller address (msg.sender)
    purchaseTimestamp = current timestamp
    transfer payment to PLATFORM_ADDRESS
    emit PaymentTransferred()
  }

  // Refund: Transfer funds back to reader if initiated by platform
  function refund(refundAmount) {
    require caller is PLATFORM_ADDRESS
    transfer refundAmount to readerAccountAddress
    emit RefundExecuted(refundAmount)
  }
}

Contract SubscriptionContractFactory {
  // Event
  event SubscriptionContractCreated(subscriptionContractAddress, userId, payment)

  // Create a new SubscriptionContract
  function createSubscriptionContract(creationDeadline, userId, payment) {
    require current timestamp <= creationDeadline
    require payment > 0

    newContract = create new SubscriptionContract with payment
    emit SubscriptionContractCreated(newContract address, userId, payment)
  }
}

```

