Final Report

Book Book (Group 12)

The new chapter for secondhand books

Presented to

Assoc. Prof. Taratip Suwannasart

Project Members

6532125821 Peneik Sitthimongkhon

6531337321 Yuatyong Chaichana

6531335021 Puvamin Tanakittanon

6531322921 Pranesh Ingkanunt

6530363421 Wasutha Sukkrasanti

6530313021 Peemdej Chancharoen

6530186721 Tapaneeya Odmung

6530128321 Nutthapong Dissanont

6530105921 Naphon Tangtrongpairoj

2110322 Database Systems

1st Semester, 2024

Department of Computer Engineering Faculty of Engineering,

Chulalongkorn University

Table of Contents

1. Introduction

1.1 Introduction

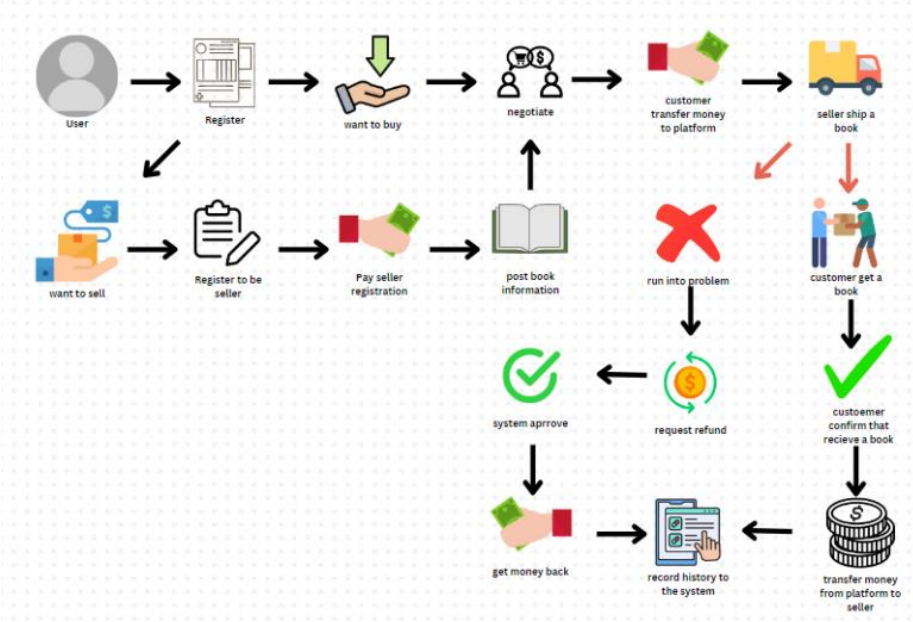
The two most popular channels for trading secondhand books in Thailand are Facebook Group Platform and Shopee E-Commerce. Currently, both present significant challenges. Facebook Group Platform offers flexibility allowing sellers and buyers to interact freely without intermediaries, but they lack security and structure. Shopee e-Commerce provides an intermediary system for safer transactions but is not specifically designed for secondhand book trading which limits its flexibility and user experience. Our goal is to create a dedicated platform that combines the strengths of both systems. By incorporating the security of an intermediary system with features specifically designed for secondhand books we aim to reduce trading risks, improve customer satisfaction and establish sustainable revenue streams for long-term success.

1.2 Organization Background

Twelfth Solution (NakornTaraTip) was founded in 2024 by a group of students and software developers. The company started as a small startup focused on helping local businesses in Thailand become more accessible to modern technology. Over the years, we have grown into a leading software development company, serving clients worldwide. Our key mission is to deliver innovative and reliable software solutions that drive business success. We value integrity, quality, and creativity in software development to ensure cutting-edge products for our clients. Our team consists of over 40 skilled engineers structured into several specialized teams: development team, business analysis team, project management team, and quality assurance team.

Together with Assoc. Prof. Nakornthip Prompoon (CEO. Twelve Solution),we aim to create business, and society impacts by improving Thai readers community, while also, being able to generate sustain revenue streams in the long run

1.3 Overview of the to-be system context



1. When opening the platform, the users will have to log in to the system with their username and password, otherwise they have to sign up with some basic information.

1.1 In case that user wants to sign up as seller, they need to go through extra step by registering their mobile number through OTP and sending ID card pictures to the system then the admin will approve their seller account to join.

2. There are multiple actions for the users to do in the system.

2.1 users can buy secondhand books through searching or browsing in recommendation feeds. Users can select tags to sort the type of the book, such as genre, number of pages, book condition, or special feature.

2.1.1 In case that there in no current selling book that the user is interested they can make a request post with the book detail that use the same description system as in selling post(2.2) System/Software requirements specification: Book Book The new chapter for second hand books 7

2.2 User that register as seller can sell their book. by create a post to sell the book. The post shall contain the book name, author name, and book condition. Users can add a special tag to identify the type of book to make their book easy to find. If that book is already in the system database. The system will add further info about the book and show the firsthand book price to be a tools for secondhand price estimation with current book condition. Otherwise, the system will request more information, and system administrators will check the validity of the book and add it to the database.

3. If the requirement is a match. Users can chat with each other through the chat box. Users can ask for further information, negotiate the price, or talk about shipping methods. The system can recommend shipping methods with different prices but can top up with extra packaging prices from seller. The price for the book including the shipment is up to both party dealing.

4. After negotiation customers will have to transfer money to the system and the book owner will deliver the book. If delivery is through the standardize method that can be access by the platform, The application will provide customer with tracking service. If not, seller need to provide the buyer with the shipment tracking themselves.

5. After book owners ship the book there will be 3 possible cases

5.1.The customer receives the book. Ensure that the book's condition on the website is correct. Customers will have to confirm in the system, then the payment will be forward to seller accounts and record that transaction as completed.

5.2.The customer receives the book, but the books have a problem as state in policy, for instance, the book's condition is not in the same state as declared by seller. Customers will have to report the problem within the time limit after receiving the book to the system. Afterward the platform will approve return request sending the book back to the owner within five calendar days. And the payment would be refund back to the customer accounts. If the inspection said that it is seller false. Seller will receive penalty.

5.3. If the customer did not receive the book. system will transfer money back to the customer. And the investigation will be launch to determine the cause of the undelivered book and may result in refund and punishment to the seller if not provide with reasonable cause.

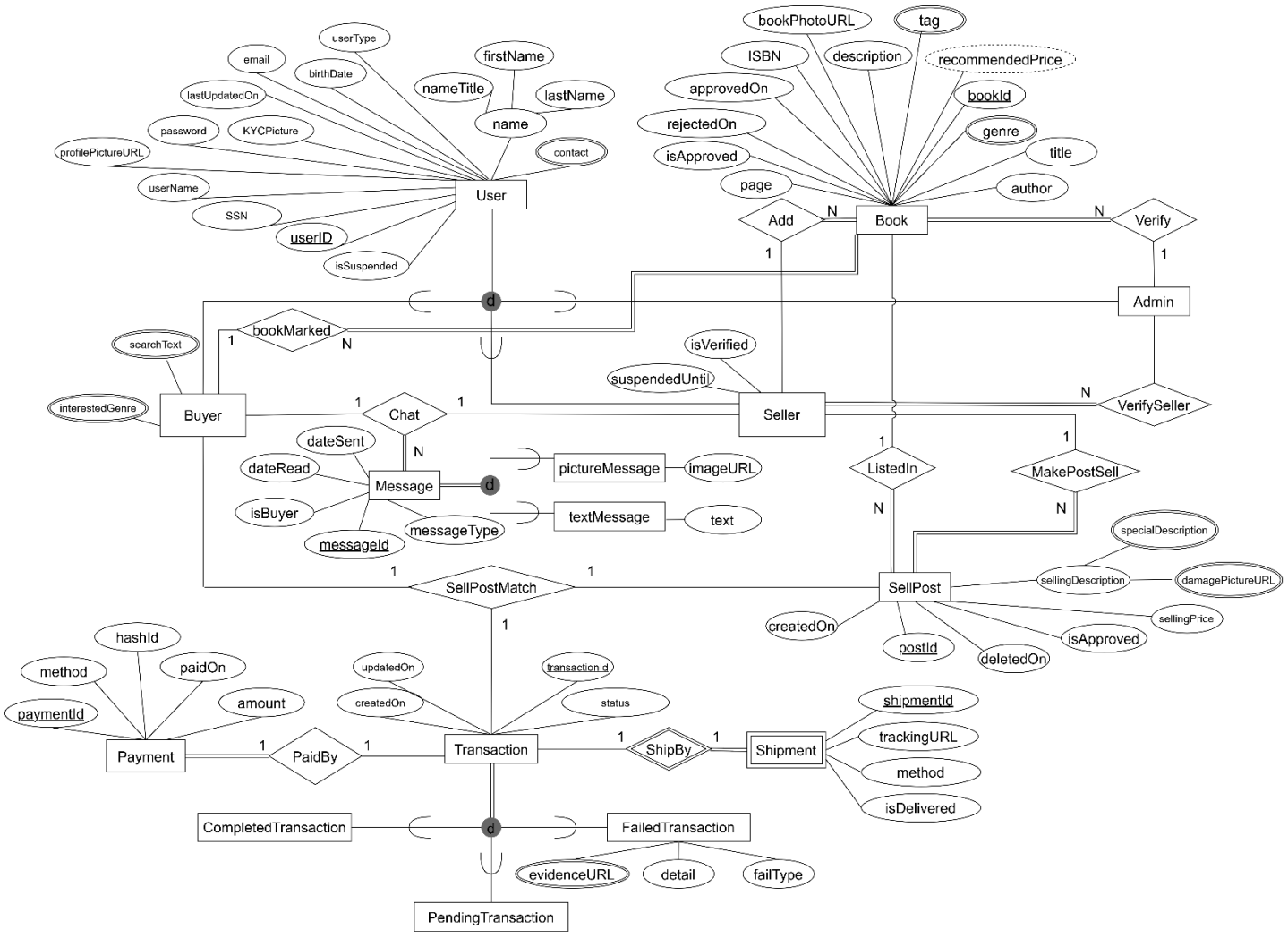
2. Project's Objective

The objective is to develop web-based applications for matching people who want to buy and sell the same book. This application provides a system where users can either choose books that they have and create a post to sell them, or they can search and find secondhand books by name or other specification, For instance, author signature, book version, or special book cover. To look for the price and book condition and negotiate with the seller for reasonable price and inquire more information about the book. This application could also prevent buyers from buying overpriced secondhand books and helping sellers sell their books at a reasonable price.

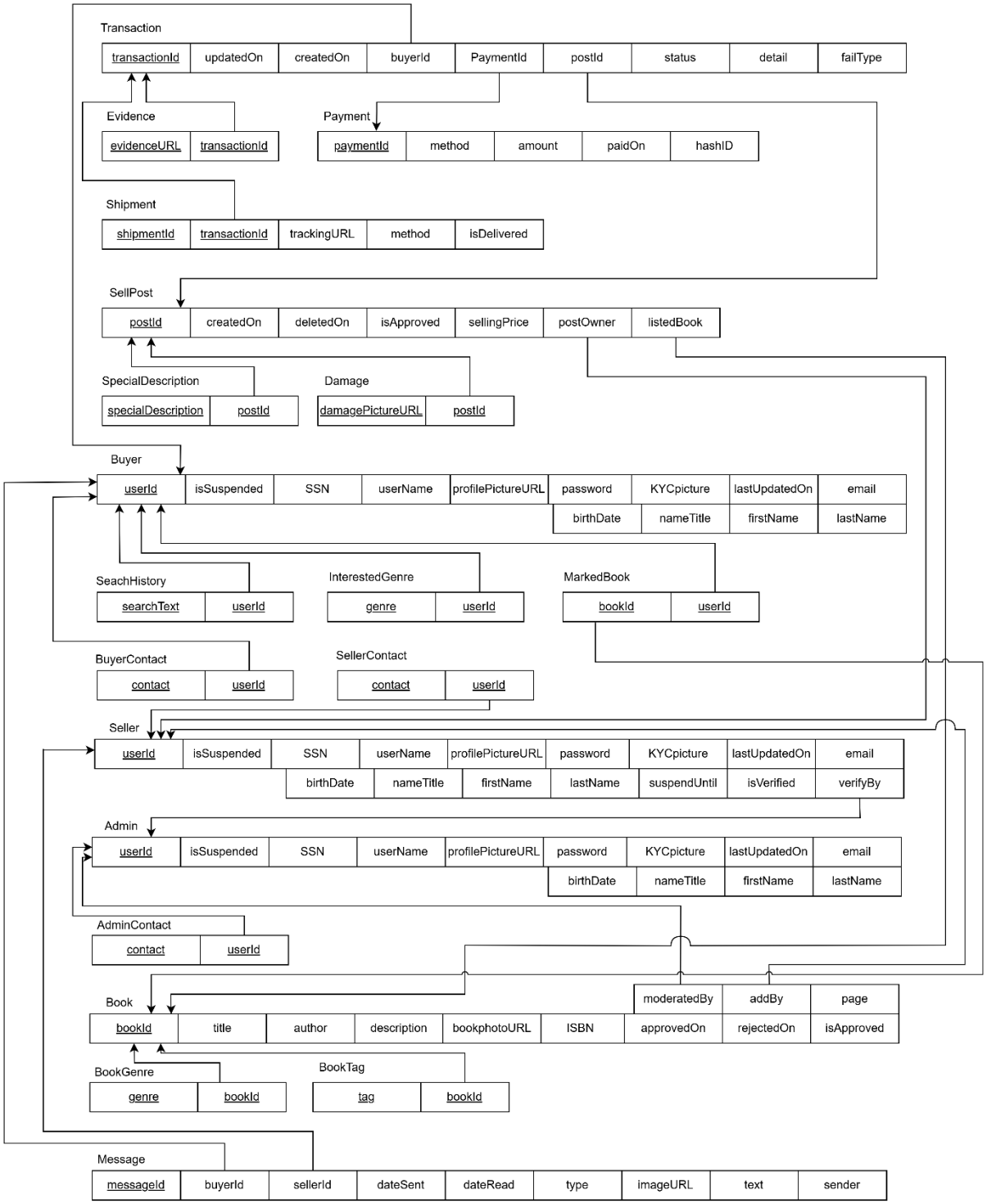
3. System Functionalities

1. **Book Discovery and Details**
    i. The system shall allow Buyers to view detailed information about books available in the database, including descriptions, authors, conditions, and prices.
    ii. The system shall allow Buyers to post requests for specific books they are looking for, including details such as title, author , and special requirements.
    iii. The system shall allow Buyers to search for books based on criteria like author's signature, limited editions, or unique features for book collectors.
2. **Communication and Transactions**
    i. The system shall allow Buyers to communicate with Sellers through a messaging or chat feature to negotiate terms, ask questions, or discuss details.
    ii. The system shall handle payment transactions for books bought by Buyers, supporting various payment methods (e.g., credit card, digital wallets).
    iii. The system shall allow Buyers to view a history of their past transactions, including purchase details and transaction statuses.
3. **Order Management**
    i. The system shall allow Buyers to track the status of their orders, including shipment tracking and delivery updates.
    ii. The system shall allow Buyers to save or bookmark books they are interested in for easy access later.
4. **Personalized Experience**
    i. The system shall provide recommendations for books based on Buyers' search history, preferences, and past purchases.
    ii. The system shall allow Buyers to leave reviews and ratings for purchased books to assist other Buyers in decision-making.

4. ER Diagram

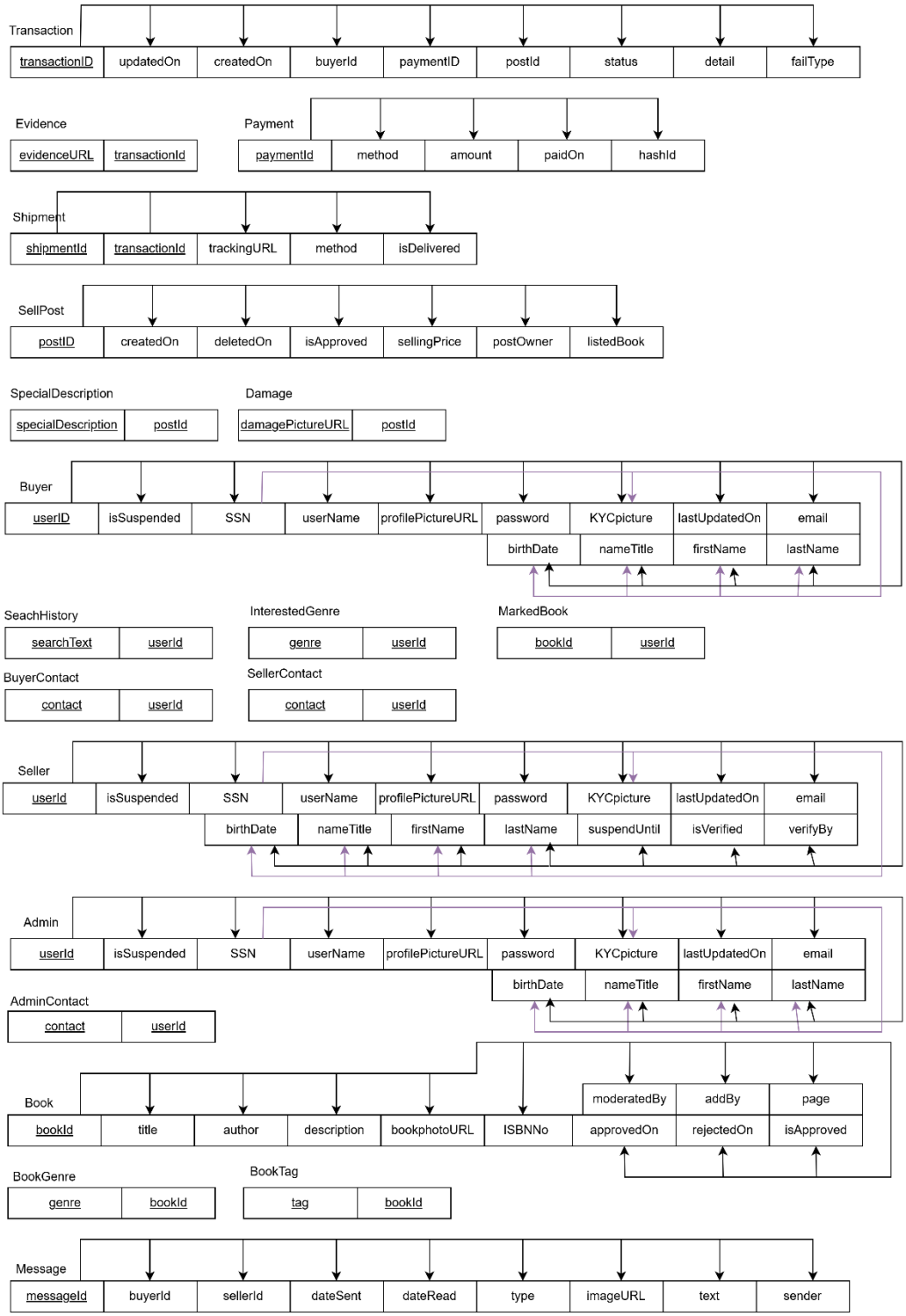

User
userType
email
birthDate
firstName
nameTitle
lastName
name
lastUpdatedOn
password
KYCPicture
contact
profilePictureURL
userName
SSN
userID
isSuspended
Book
bookPhotoURL
tag
ISBN
description
recommendedPrice
approvedOn
bookId
rejectedOn
genre
isApproved
title
page
author
Add
Verify
Admin
bookMarked
searchText
interestedGenre
Buyer
Chat
Seller
isVerified
suspendedUntil
VerifySeller
dateSent
dateRead
Message
isBuyer
messageId
messageType
pictureMessage
imageURL
textMessage
text
ListedIn
MakePostSell
SellPostMatch
SellPost
specialDescription
sellingDescription
damagePictureURL
sellingPrice
createdOn
postId
deletedOn
isApproved
hashId
method
paidOn
paymentId
amount
Payment
PaidBy
updatedOn
createdOn
transactionId
status
Transaction
ShipBy
Shipment
shipmentId
trackingURL
method
isDelivered
CompletedTransaction
FailedTransaction
evidenceURL
detail
failType
PendingTransaction

5. Schema Diagram



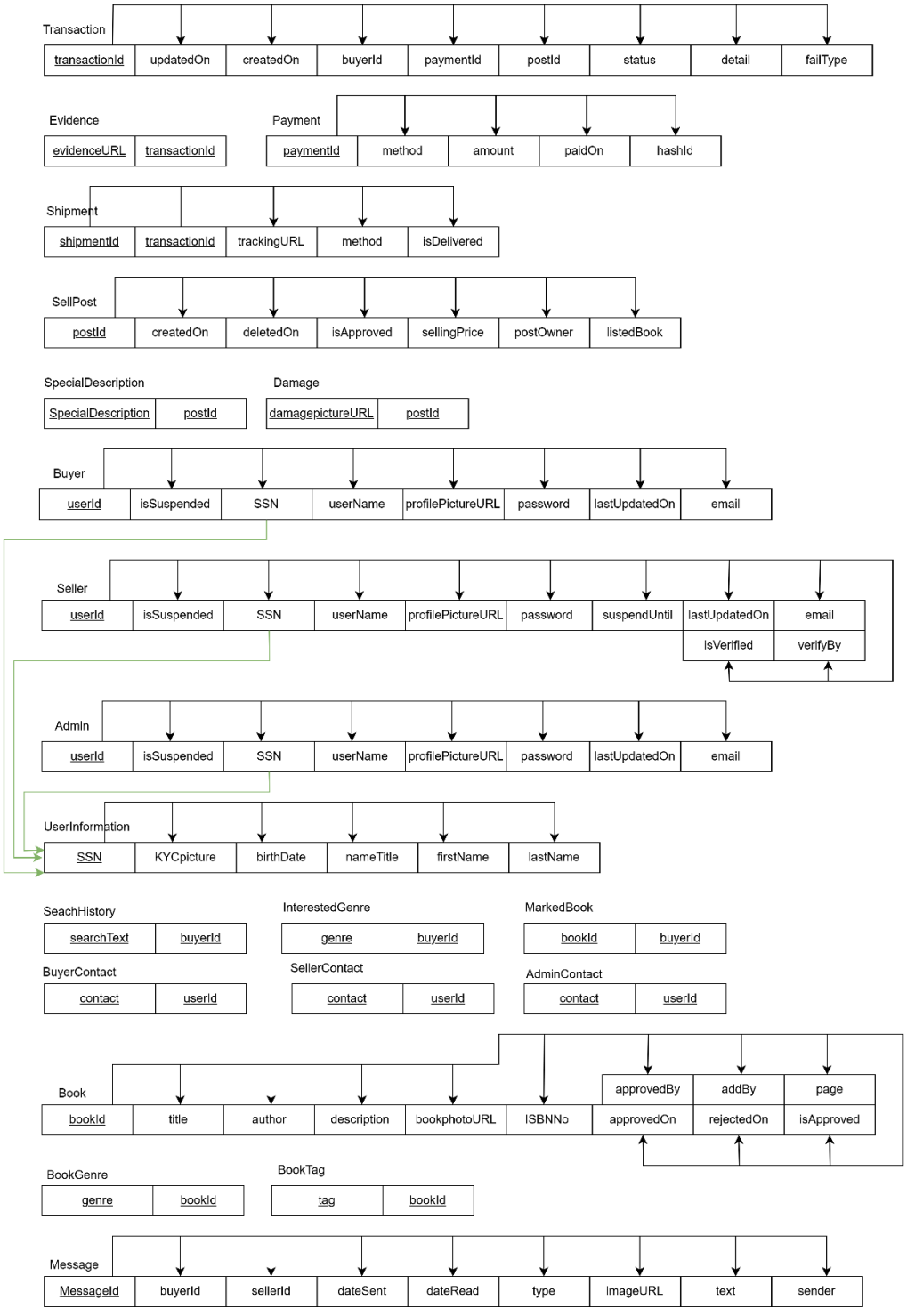
6. Normalization of Relation (3NF)

From the schema we can define FD(Functional Dependency)of all attribute in our schema as

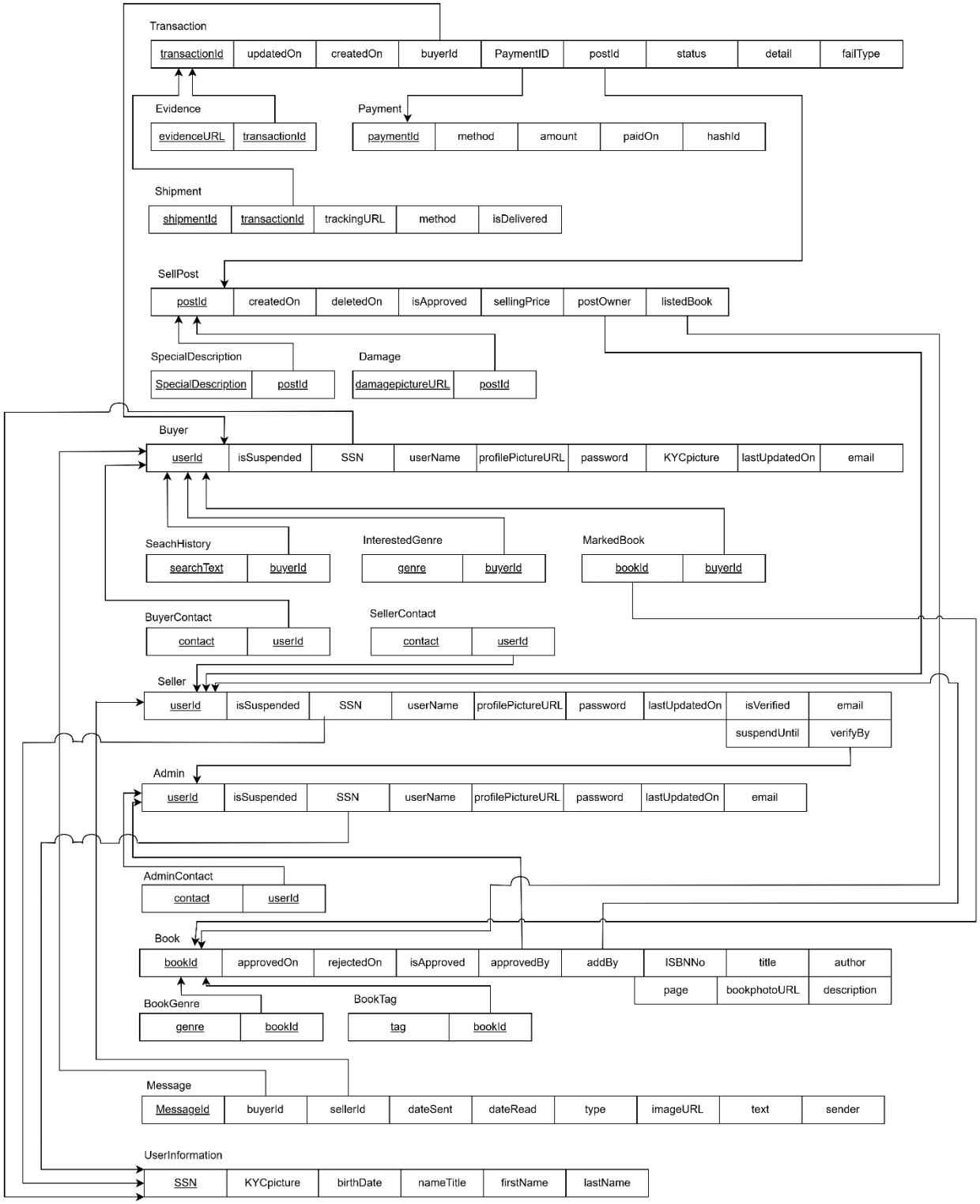


From the schema with FD, There are still FDs that violate the principles of 3NF (Third Normal Form) as some attributes have dependency on non-primary attributes. In Seller, Buyer and Admin, The non-primary attribute SSN can imply some of the formal user information.

After fixing the schema to compile with the principles of 3NF, We got schema with all FDs and new foreign key indications as



And the new main schema as



7. Data Dictionary of Relation

Tables		
No	Name	Description
1	Admin	A user who is managing the platform
2	AdminContact	Admin Contact Information
3	Book	A book in the database
4	BookGenre	Genre of the book
5	BookTag	Tag of the book
6	Buyer	A user who is buying books
7	BuyerContact	Buyer Contact Information
8	Chat	A chat between a seller and a buyer
9	Damage	Damage Information of Selling Book
10	Evidence	Evidence of Fail Transaction
11	InterestedGenre	Buyer Interested Genre Information
12	MarkedBook	Buyer Marked Book Information
13	Payment	Payment information
14	SearchHistory	Buyer Search History Information
15	SellPost	A post created by a seller to sell their book
16	Seller	A user who is selling books
17	SellerContact	Seller Contact Information
18	Shipment	Shipping information
19	SpecialDescription	Special Description of Selling Book
20	Transaction	Transaction information
21	UserInformation	Information of the user

Table Name: Admin					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	userId	SERIAL	Buyer ID	Number	FALSE
2	isSuspended	BOOLEAN	Account Status	TRUE/FALSE	FALSE
3	SSN	VARCHAR(20)	SSN or Passport ID	Valid SSN or PassportID Format	FALSE
4	userName	VARCHAR(50)	Username	Alphanumeric	FALSE
5	profilePictureURL	TEXT	Profile Picture URL	Valid URL	TRUE
6	password	VARCHAR(255)	User Password	Encrypted String	FALSE
7	lastUpdatedOn	TIMESTAMP	Last Updated Timestamp	Valid Date/Time Format	FALSE
8	email	VARCHAR(100)	Email Address	Valid Email Format	FALSE

Table Name: AdminContact					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	contact	VARCHAR(100)	Contacts	Alphanumeric	FALSE
2	userId	INTEGER	Admin ID	Number	FALSE

Table Name: Book					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	bookId	SERIAL	Book ID	Numeric	FALSE
2	approvedOn	TIMESTAMP	Approval Timestamp	Valid Date/Time Format	TRUE
3	rejectedOn	TIMESTAMP	Rejection Timestamp	Valid Date/Time Format	TRUE
4	isApproved	BOOLEAN	Book Status	TRUE/FALSE	FALSE
5	moderatedBy	INTEGER	Admin ID	Number	TRUE
6	addBy	INTEGER	Seller ID	Number	TRUE
7	ISBN	VARCHAR(13)	ISBN Number	Standard ISBN Format	TRUE
8	title	VARCHAR(200)	Book Title	Alphanumeric	FALSE
9	author	VARCHAR(100)	Author Name	Alphanumeric	TRUE
10	page	INTEGER	Number of Pages	Positive Integer	TRUE
11	bookPhotoURL	TEXT	Book Photo URL	Valid URL	FALSE
12	description	TEXT	Book Description	Alphanumeric	TRUE

Table Name: BookGenre					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	genre	ENUM	Genres	'Fiction', 'Non-Fiction', 'Mystery', 'Thriller', 'Romance', 'Science Fiction', 'Fantasy', 'Historical Fiction', 'Horror', 'Biography', 'Memoir', 'Self-Help', 'Health & Wellness', 'Psychology', 'Poetry', 'Drama', 'Adventure', 'Children's Literature', 'Young Adult', 'Graphic Novels / Comics', 'Crime', 'True Crime', 'Dystopian', 'Contemporary', 'Religious / Spiritual'	FALSE
2	bookId	INTEGER	book ID	Number	FALSE

Table Name: BookTag					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	tag	ENUM	Tags	'Bestseller', 'New Release', 'Classic', 'Award-Winning', 'Must-Read', 'Highly Recommended', 'Inspirational', 'Coming of Age', 'Family Saga', 'Historical', 'Dark Fantasy', 'Detective', 'LGBTQ+', 'Young Adult', 'Children's Book', 'Short Stories', 'Mystery', 'Self-Help', 'Thriller', 'Romantic Comedy'	FALSE
2	bookId	INTEGER	book ID	Number	FALSE

Table Name: Buyer					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	userId	SERIAL	Buyer ID	Number	FALSE
2	isSuspended	BOOLEAN	Account Status	TRUE/FALSE	FALSE
3	SSN	VARCHAR(20)	SSN or Passport ID	Valid SSN or PassportID Format	FALSE
4	userName	VARCHAR(50)	Username	Alphanumeric	FALSE
5	profilePictureURL	TEXT	Profile Picture URL	Valid URL	TRUE
6	password	VARCHAR(255)	User Password	Encrypted String	FALSE
7	lastUpdatedOn	TIMESTAMP	Last Updated Timestamp	Valid Date/Time Format	FALSE
8	email	VARCHAR(100)	Email Address	Valid Email Format	FALSE

Table Name: BuyerContact					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	contact	VARCHAR(100)	Contacts	Alphanumeric	FALSE
2	userId	INTEGER	Buyer ID	Number	FALSE

Table Name: Chat					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	messageId	SERIAL	Message ID	Number	FALSE
2	buyerId	INTEGER	Buyer ID	Number	FALSE
3	sellerId	INTEGER	Seller ID	Number	FALSE
4	dateSent	TIMESTAMP	Date Sent	Valid Date/Time Format	FALSE
5	dateRead	TIMESTAMP	Date Read	Valid Date/Time Format	TRUE
6	type	ENUM	Message Type	'Text', 'Image'	FALSE
7	imageURL	TEXT	Picture Message URL	Valid URL	TRUE
8	text	TEXT	Text Message Content	Text	TRUE
9	sender	ENUM	is Message Send by Buyer	'Buyer', 'Seller'	FALSE

Table Name: Damage					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	damagePictureURL	TEXT	Damage Information or Damage Picture URL	Text or Valid URL	FALSE
2	postId	INTEGER	Sell Post ID	Number	FALSE

Table Name: Evidence					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	transactionId	INTEGER	Transaction ID	Number	FALSE
2	evidenceURL	TEXT	Evidence picture URL	Valid URL	FALSE

Table Name: InterestedGenre					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	genre	ENUM	Interested Genres	'Fiction', 'Non-Fiction', 'Mystery', 'Thriller', 'Romance', 'Science Fiction', 'Fantasy', 'Historical Fiction', 'Horror', 'Biography', 'Memoir', 'Self-Help', 'Health & Wellness', 'Psychology', 'Poetry', 'Drama', 'Adventure', 'Children's Literature', 'Young Adult', 'Graphic Novels / Comics', 'Crime', 'True Crime', 'Dystopian', 'Contemporary', 'Religious / Spiritual'	FALSE
2	userId	INTEGER	Buyer ID	Number	FALSE

Table Name: MarkedBook					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	bookId	INTEGER	Marked Book IDs	Positive Integers	FALSE
2	userId	INTEGER	Buyer ID	Number	FALSE

Table Name: Payment					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	paymentId	SERIAL	Payment ID	Number	TRUE
2	method	ENUM	Payment Method	Credit Card', 'Debit Card', 'PayPal', 'Bank Transfer', 'Cash on Delivery', 'Apple Pay', 'Google Pay', 'Amazon Pay', 'Cryptocurrency', 'Gift Card', 'Alipay', 'WeChat Pay', 'Stripe', 'Venmo', 'Samsung Pay', 'Afterpay', 'Klarna', 'Payoneer', 'Mobile Money', 'CashApp'	FALSE
3	amount	DECIMAL(10,2)	Payment Amount	Positive Decimal	FALSE
4	paidOn	TIMESTAMP	Payment Timestamp	Valid Date/Time Format	TRUE
5	hashId	VARCHAR(100)	Payment Hash ID	Unique Hash Values	TRUE

Table Name: SearchHistory					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	searchText	VARCHAR(100)	Search History	Text	FALSE
2	userId	INTEGER	Buyer ID	Number	FALSE

Table Name: SellPost					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	postId	SERIAL	Sell Post ID	Number	FALSE
2	createdOn	TIMESTAMP	Creation Timestamp	Valid Date/Time Format	FALSE
3	deletedOn	TIMESTAMP	Deletion Timestamp	Valid Date/Time Format	TRUE
4	isApproved	BOOLEAN	Post Status	TRUE/FALSE	FALSE
5	sellingPrice	DECIMAL(10,2)	Selling Price	Positive Decimal	FALSE
6	postOwner	INTEGER	Seller ID	Number	FALSE
7	listedBook	INTEGER	Book ID	Number	FALSE

Table Name: Seller					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	userId	SERIAL	Buyer ID	Number	FALSE
2	isSuspended	BOOLEAN	Account Status	TRUE/FALSE	FALSE
3	SSN	VARCHAR(20)	SSN or Passport ID	Valid SSN or PassportID Format	FALSE
4	userName	VARCHAR(50)	Username	Alphanumeric	FALSE
5	profilePictureURL	TEXT	Profile Picture URL	Valid URL	TRUE
6	password	VARCHAR(255)	User Password	Encrypted String	FALSE
7	lastUpdatedOn	TIMESTAMP	Last Updated Timestamp	Valid Date/Time Format	FALSE
8	email	VARCHAR(100)	Email Address	Valid Email Format	FALSE
9	isVerified	BOOLEAN	Verification Status	TRUE/FALSE	FALSE
10	suspendedUntil	TIMESTAMP	Suspension End Timestamp	Valid Date/Time Format	TRUE
11	verifiedBy	INTEGER	Admin ID	Number	TRUE

Table Name: SellerContact					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	contact	VARCHAR(100)	Contacts	Alphanumeric	FALSE
2	userId	INTEGER	Seller ID	Number	FALSE

Table Name: Shipment					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	shipmentId	SERIAL	Shipment ID	Number	FALSE
2	transactionId	INTEGER	Transaction ID	Number	FALSE
3	trackingURL	TEXT	Tracking URL	Valid URL	TRUE
4	method	ENUM	Shipping Method	'Standard', 'Express'	FALSE
5	isDelivered	BOOLEAN	Shipment Status	TRUE/FALSE	FALSE

Table Name: SpecialDescription					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	specialDescription	ENUM	Special Description	'Author Signature', 'Limited Edition', 'First Edition', 'Special Cover Art', 'Illustrated Edition', 'Collector's Edition', 'Slipcase Edition', 'Leather-Bound', 'Gilded Edges', 'Deckle Edges', 'Pop-Up Elements', 'Fold-Out Pages', 'Handwritten Notes by Author', 'Personalized Message', 'Numbered Edition', 'Exclusive Artwork', 'Embossed Cover', 'Gold Foil Stamping', 'Box Set', 'Anniversary Edition', 'Hardcover with Dust Jacket', 'Transparent Cover', 'Annotated Edition', 'Signed by Illustrator', 'Map Insert', 'Supplementary Materials', 'Exclusive Content', 'Fan Art Edition', 'Interactive Elements (e.g., Augmented Reality)', 'Bilingual Edition'	FALSE
2	postId	INTEGER	Sell Post ID	Number	FALSE

Table Name: Transaction					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	transactionId	SERIAL	Transaction ID	Number	FALSE
2	updatedOn	TIMESTAMP	Last Update Timestamp	Valid Date/Time Format	TRUE
3	createdOn	TIMESTAMP	Creation Timestamp	Valid Date/Time Format	FALSE
4	buyerId	INTEGER	Buyer ID	Number	FALSE
5	paymentId	INTEGER	Payment ID	Number	FALSE
6	postId	INTEGER	Post ID	Number	FALSE
7	status	ENUM	Transaction Status	'Failed', 'Pending', 'Completed'	FALSE
8	detail	TEXT	Fail Transaction Detail	Alphanumeric	TRUE
9	failType	ENUM	Message Type	'Buyer', 'Shipping', 'Other'	TRUE

Table Name: UserInformation					
No	Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
1	SSN	VARCHAR(20)	SSN or Passport ID	Valid SSN or PassportID Format	FALSE
2	KYCpicture	TEXT	KYC Picture URL	Valid URL	FALSE
3	birthDate	DATE	Birth Date	Valid Date Format	FALSE
4	nameTitle	ENUM	Name Title (Mr., Ms., Mrs.)	'Mr.', 'Ms.', 'Mrs.'	FALSE
5	firstName	VARCHAR(50)	First Name	Alphabetic	FALSE
6	lastName	VARCHAR(50)	Last Name	Alphabetic	FALSE

8. SQL Queries

8.1. INSERT

Posting a new book can be achieved by running this query.

```
INSERT INTO sellpost (postOwner, listedBook, sellingPrice)
VALUES (2, 2, 50);
```

A wrong example where referential integrity is violated. Given that postowner id of 1000 does not exist.

```
INSERT INTO sellpost (postOwner, listedBook, sellingPrice)
VALUES (1000, 2, 10);
```

Another wrong example where the constraint is violated. See that sellingprice must be strictly positive.

```
INSERT INTO sellpost (postOwner, listedBook, sellingPrice)
VALUES (2, 2, -10);
```

8.2. DELETE

First, we insert new row into the seller table just for the clarity of explanation.

```
INSERT INTO seller (email, password, ssn, username, profilepictureurl)
VALUES (
   'tontoeylnwzasellbook@gmail.com',
   '78590657fe72bb2e9d87a3f4f89d47e48ecd28dd42660c0c1c7ddb3ac286825f',
   '1112223334446',
   'tontoeysellbook',
   'https://cdn.anime-planet.com/characters/primary/hamtaro-1.jpg?t=1625777064'
);
```

Deleting a seller from can be achieved by running this query.

```
DELETE FROM seller WHERE userid = 5;
SELECT * FROM seller:
```

A wrong example where referential integrity is violated. Provided that seller with userId of 1 is still referenced by some other tables.

```
DELETE FROM seller WHERE userid = 1;
```

8.3. UPDATE

Buyer username can be changed by running this query.

```
UPDATE buyer SET username = 'bo0k'
WHERE userId = 2;
```

A wrong example where referential integrity is violated. Provided that SSN of '1110000004444' does not exist in userinformation.

```
UPDATE buyer SET SSN = '1110000004444'
WHERE userId = 2;
```

A wrong example where the constraint is violated. In this case, buyer username must be unique. Given that buyer with username of 'farofaro' already exists.

```
UPDATE buyer SET username = 'farofaro'
WHERE userId = 2;
```

9. Indexing

According the functional requirement, buyers should be able to view seller profile and their posts. Therefore, we frequently have to filter posts by postowner as demonstrated below. We decided to create a B+ tree index on postowner field in sellpost table.

```
SELECT * FROM sellpost WHERE postowner = 1;
```

```
CREATE INDEX sellpost_postowner_idx
ON sellpost USING btree (postowner);
```

Also, buyers should be able to filter posts by listedbook so that they can compare price and conditions. Therefore, we frequently have to filter posts by listedbook as demonstrated below. We decided to create a B+ tree index on listedbook field in sellpost table.

```
SELECT * FROM sellpost WHERE listedbook = 7;
```

```
CREATE INDEX sellpost_listedbook_idx
ON sellpost USING btree (listedbook);
```

10. Stored Procedures

In case we have to permanently suspend seller account, we also have to disapprove all of their existing posts. This stored procedure will be handy to accomplish such task.

```
CREATE PROCEDURE perm_suspend_seller(sellerid INT)
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE seller
    SET issuspended = true
    WHERE userid = sellerid;
    UPDATE sellpost
    SET isapproved = false
    WHERE postowner = sellerid;
END;
$$;
```

For instance, we want to permanently suspend seller with id of 4 and disapprove all of seller existing posts.

```
CALL perm_suspend_seller(4);
```

In the present day, many gambling websites promote their sites on multiple platforms by posting links, sometime malicious, on the platforms. To be able to counter with this type of spam, we should be able to remove buyer, seller or admin contacts which contain certain text keyword.

```
CREATE PROCEDURE remove_contact_with_keyword(keyword VARCHAR)
LANGUAGE plpgsql
AS $$
   DECLARE
      deleted_seller INT := 0;
      deleted_buyer INT := 0;
      deleted_admin INT := 0;
   BEGIN
      DELETE FROM sellercontact
      WHERE contact LIKE '%' || keyword || '%';
      GET DIAGNOSTICS deleted_seller = ROW_COUNT;

      DELETE FROM buyercontact
      WHERE contact LIKE '%' || keyword || '%';
      GET DIAGNOSTICS deleted_buyer = ROW_COUNT;

      DELETE FROM admincontact
      WHERE contact LIKE '%' || keyword || '%';
      GET DIAGNOSTICS deleted_admin = ROW_COUNT;

      RAISE NOTICE 'Deleted % row(s) from sellercontact, % row(s) from buyercontact, and %
row(s) from admincontact where contact contains "%".',
      deleted_seller, deleted_buyer, deleted_admin, keyword;
   END;
$$;
```

For instance, we want to remove contact with “mark888.com “ URL text to block gambling site promotion

```
CALL remove_contact_with_keyword('mark888.com');
```

11. Stored Functions

Sometime we want to classify buyers into tier based on their spendings. This stored functions will become handy as executives can easily change the tier criteria.

```
CREATE FUNCTION buyer_tier(total_spent FLOAT)
   RETURNS VARCHAR(10)
   LANGUAGE plpgsql
   AS
$$
   DECLARE
      tier VARCHAR(10);
   BEGIN
      IF (total_spent = 0) THEN
         tier := 'Normal';
      ELSEIF (total_spent > 0 AND total_spent <= 100) THEN
         tier := 'Bronze';
      ELSEIF (total_spent > 100 AND total_spent <= 1000) THEN
         tier := 'Silver';
      ELSEIF (total_spent > 1000) THEN
         tier := 'Gold';
      END IF;
      RETURN (tier);
   END;
$$
```

The example use case is we sum each buyer spending. Then, use the stored function to classify their spendings into tier as demonstrated below.

```
SELECT userid, COALESCE(sum(amount), 0) total_spending,
buyer_tier(COALESCE(sum(amount), 0))
FROM buyer
   LEFT JOIN transaction AS t
   ON t.buyerid = buyer.userid
   LEFT JOIN payment AS p
   ON t.paymentid = p.paymentid
GROUP BY userid
ORDER BY total_spending DESC
```

Another mean to block gambling site promotion is to censor certain text keyword. This stored function will play a role into detecting if the given text string contain restricted keywords or not. It will return TRUE if it does, and return FALSE otherwise.

```
CREATE FUNCTION gamble_inapprop_text(input_string text)
   RETURNS BOOLEAN
   LANGUAGE plpgsql
   AS
$$
DECLARE
   word_list text[] := ARRAY['พนัน', 'บาคาร่า', 'โอนไว', 'mark888'];
   word text;
BEGIN
   FOREACH word IN ARRAY word_list LOOP
      IF STRPOS(input_string, word) > 0 THEN
         RETURN TRUE;
      END IF;
   END LOOP;
   RETURN FALSE;
END;
$$
```

For example, we may want to list sellers who promote gambling site on the book adding feature.

```
SELECT DISTINCT addedBy AS userid, username
FROM
   (SELECT *, gamble_inapprop_text(title) check1,
   gamble_inapprop_text(description) check2
   FROM book) AS temp
JOIN seller
ON seller.userid = temp.addedby
WHERE (check1 OR check2)
```

12. Triggers

On the lastupdatedon field in some tables, it should update automatically when other fields value are altered. We then create trigger to accomplish such task.

```
CREATE FUNCTION update_lastupdatedon()
RETURNS TRIGGER
LANGUAGE plpgsql
AS
$$
   BEGIN
      IF (NEW.username <> OLD.username) OR (NEW.profilepictureurl <> OLD.profilepictureurl)
THEN
         NEW.lastupdatedon := NOW()::timestamp(0);
      END IF;
      RETURN NEW;
   END;
$$;
```

```
CREATE TRIGGER update_lastupdatedon_trigger
BEFORE UPDATE ON buyer
FOR EACH ROW
EXECUTE FUNCTION update_lastupdatedon();
```

Now that if buyer change username, the lastupdatedon field will be updated automatically.

```
UPDATE buyer SET username = 'tontoeyngaeee'
WHERE userid = 3;

SELECT * FROM buyer;
```

Another case where the usage of trigger might be useful is when shipment is delivered. After shipment is delivered, its transaction status should be automatically set to be 'Completed'.

```
CREATE OR REPLACE FUNCTION update_transaction_status()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   IF NEW.isdelivered = TRUE AND OLD.isdelivered = FALSE THEN
      UPDATE transaction
      SET status = 'Completed', failtype = null
      WHERE transactionid = NEW.transactionid;
   END IF;
   RETURN NEW;
END;
$$
```

```
CREATE TRIGGER trigger_update_trans_status
AFTER UPDATE OF isdelivered ON shipment
FOR EACH ROW
WHEN (NEW.isdelivered = TRUE)
EXECUTE FUNCTION update_transaction_status();
```

For instance, after certain shipment is delivered, its transaction status will be automatically set to be 'Completed'.

```
UPDATE shipment
SET isdelivered = true
WHERE shipmentid = 1;
```

13. Execution Path

In this case we want to list all users who are not suspended. The first query Is selected over the second one due to its efficiency. See that the first query only performs one sequential scan. Further details and execution path of each query are provided below.

13.1. Query I

```
SELECT username, ssn
FROM seller
WHERE NOT isSuspended;
```

Statistics per Node Type

Node type	Count
Seq Scan	1

Statistics per Relation

Relation name	Scan count
Node type	Count
seller	1
Seq Scan	1

	Node
1.	→ Seq Scan on seller as seller Filter: (NOT issuspended)

13.2. Query II

```
SELECT username, ssn
FROM seller
WHERE userid IN
(SELECT userid FROM seller
WHERE NOT isSuspended);
```

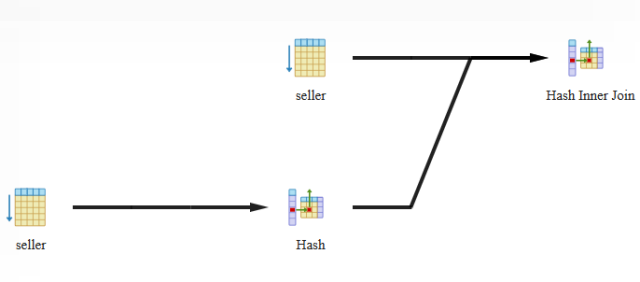
Statistics per Node Type

Node type	Count
Hash	1
Hash Inner Join	1
Seq Scan	2

Statistics per Relation

Relation name	Scan count
Node type	**Count**
seller	2
Seq Scan	2

#	Node
1.	→ Hash Inner Join **Hash Cond**: (seller.userid = seller_1.userid)
2.	→ Seq Scan on seller as seller
3.	→ Hash
4.	→ Seq Scan on seller as seller_1 **Filter**: (NOT issuspended)



14. Complex Query

For the first scenario, we may want to list buyer user (who has at least 1 payment) by their average discount percentage givens.

```
SELECT buyerid, f firstname, l lastname,
ROUND(AVG(((1 - c/b))), 3) * 100 avg_percent_discount
FROM
   (
   SELECT transaction.buyerid AS buyerid, u.firstname f, u.lastname l, sellingprice b, amount c
FROM transaction
   JOIN sellpost
   ON transaction.postid = sellpost.postid
   JOIN payment
   ON transaction.paymentid = payment.paymentid
   JOIN buyer
   ON transaction.buyerid = buyer.userid
   JOIN userinformation AS u
   ON buyer.ssn = u.ssn
   WHERE transaction.paymentid IS NOT NULL
   ) AS temp
GROUP BY buyerid, f, l
ORDER BY SUM(ROUND((1 - c/b),4) * 100) DESC;
```

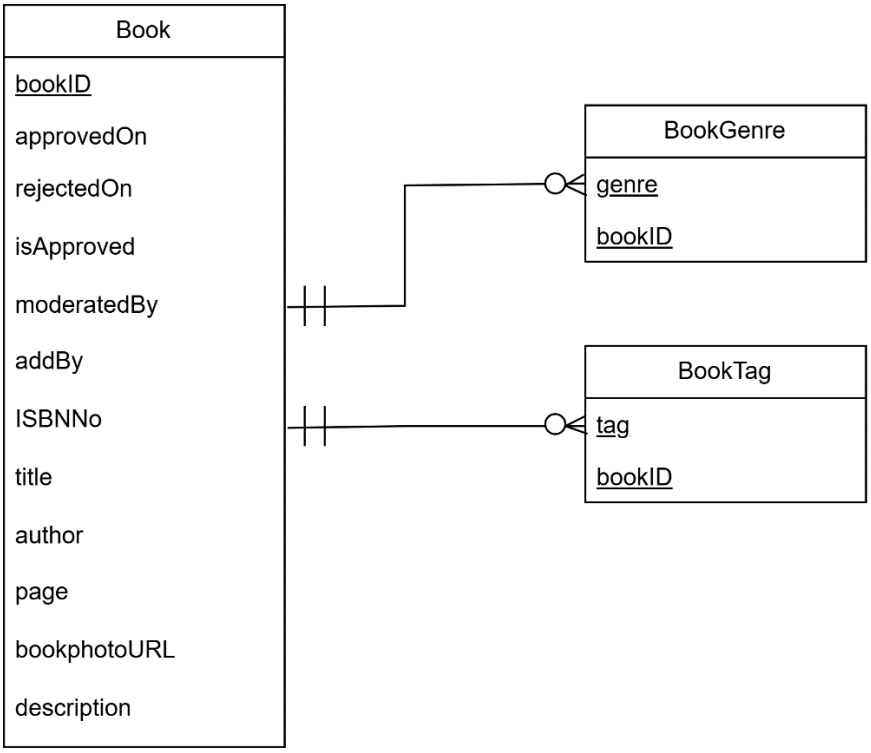
For the second scenario, we may want to find gambling promotion contents by sellers using any features on the platform.

```
(SELECT sellerid, 'Message' feature, text
FROM message
WHERE sender = 'Seller' AND gamble_inapprop_text(text))
UNION
(SELECT addedby sellerid, 'Book Description' feature, description
FROM book
WHERE gamble_inapprop_text(description))
UNION
(SELECT userid, 'Contact' feature, contact
FROM sellercontact
WHERE gamble_inapprop_text(contact))
ORDER BY sellerid;
```

For the third scenario, we may want to find books with genre that interest buyers the most. Note that there can be more than one book.

```
SELECT title, genre
FROM book
JOIN bookgenre
ON book.bookid = bookgenre.bookid
WHERE genre
IN
(
   SELECT genre
   FROM interestedgenre
   GROUP BY genre
   HAVING COUNT(*) = (
      SELECT MAX(genre_count)
      FROM (
         SELECT COUNT(*) AS genre_count
         FROM interestedgenre
         GROUP BY genre
      ) AS temp
   )
)
```

15. Document-based Design Schema, Multi-values Example



The entity Book is having the one-to-few relation with both BookGenre and BookTag so using embedding to define those relations is reasonable.

The document based design schema of the definition above is provided below.

```
{
  "title": "Book",
  "bsonType" : "object",
  "required": ["bookId", "title", "bookPhotoURL"],
  "properties": {
    "_id": {"bsonType": "objectId"},
    "bookid": { "bsonType": "int" , "auto_increment" : true},
    "title": { "bsonType": "string", "maxLength": 200},
    "author": { "bsonType": "string", "maxLength": 100 },
    "description": { "bsonType": "string" },
    "bookPhotoURL": { "bsonType": "string" },
    "ISBN": { "bsonType": "string", "maxLength": 13 },
    "isApproved": { "bsonType": "bool", "default": false },
    "approvedOn": { "bsonType": "date" },
    "rejectedOn": { "bsonType": "date" },
    "moderatedBy": { "bsonType": "int" },
    "addedBy": { "bsonType": "int" },
    "page": { "bsonType": "int", "min": 1 },
    "genres": {
      "bsonType": "array",
      "uniqueItems" :  true,
      "items": {
        "bsonType": "string",
        "enum": [
            "Fiction", "Non-Fiction", "Mystery", "Thriller", "Romance", "Science Fiction",
            "Fantasy", "Historical Fiction", "Horror", "Biography", "Memoir", "Self-Help",
            "Health & Wellness", "Psychology", "Poetry", "Drama", "Adventure", "Children's Literature",
            "Young Adult", "Graphic Novels/Comics", "Crime", "True Crime", "Dystopian",
            "Contemporary", "Religious/Spiritual"
        ]
      }
    },
    "tags": {
      "bsonType": "array",
      "uniqueItems" :  true,
      "items": {
        "bsonType": "string",
        "enum": [
          "Bestseller", "New Release", "Classic", "Award-Winning", "Must-Read",
          "Highly Recommended", "Inspirational", "Coming of Age", "Family Saga",
          "Historical", "Dark Fantasy", "Detective", "LGBTQ+", "Young Adult",
          "Children's Book", "Short Stories", "Mystery", "Self-Help",
          "Thriller", "Romantic Comedy"
        ]
      }
    }
  }
}
```

16. Basic NoSQL Function Example

Using the Book collection, apply an aggregate function to retrieve all authors, sorted by the total number of books each has contributed, from the highest to the lowest.

```
db.Book.aggregate([
  {
    // Filter to include only approved books
    $match: {
      isApproved: true
    }
  },
  {
    // Group by author and count the number of books
    $group: {
      _id: "$author",
      count: {
        $sum: 1
      }
    }
  },
  {
    // Sort by order of the author's books amount
    $sort:
      {
        count: -1,
        _id: 1
      }
  },
  {
    // Rename all field to author for output consistency
    $project: {
      _id: 0,
      author: "$_id",
      count: "$count"
    }
```