1) Requirement engineerin:

Will take all required information from the online book reselling about how they want the website to function, look and what kind of features it will need.

2) System design

Will design the system UML diagrams using software's like magicdraw to give a prototype and diagrams to show the relationship between different classes of how this website will work and communicate with users

PART 1: Overview & Software Requirements Specification

Introduction:

A) Purpose:

To develop a virtual bookstore that facilitate online functions like browse books, buying books, uploading books, through some modules suitable for a user's role like visitor, customer, administrator.

B) Project scope :

User:

- Enable user to register with basic registration details
- Enable user to login into the system in order to access the system.

- Once user is logged into the, he/she may view all the added books with their details.
- User can surf various book of their choice and buy them.
- System allows user to sell their books online buy added the book name and its details.
- Enable user to upload a book if he/she wants to sell.
- All the purchase history of user will be displayed with details.
- If any buyer is interested in buying a book or anyone buys a book from the user then buyer's details will be displayed.

Admin:

- admin need to login into the system in order to access the system.
- Admin can view all the added books online with their details
- System allows admin to view all the transaction details of buying and selling a book.
- All the registered user details will be displayed to the admin

C) Glossary and Abbreviations :

- Used UML notations

D) List of the System Stakeholders :

- Administrators
- Owners

2) Software Requirement Pattern:

3) Functional USER Requirements:

Register: Allows Account Registration

1- The system shall allow a non-registered user to create a new account.
2- The system shall require the following information from the user: Name, password.
3- The system shall ask the user for a username and password.

4- The system shall confirm the username and password are acceptable (username not taken).
5- The system shall store the information in the database.

Login: Allows Account Login

1. The system shall allow a registered user to log-in to their account.
2. The system shall require a username and password from the user.
3. The system will verify the username and password, and the user will be considered logged-in.

catalouge: Allows user to show

1. The catalouge allow a user to show for books with title, author, price, info.
2. The catalouge results will include a picture of the front cover, along with the title, author, price, availability, and condition of the book.

Update Account Information: Allows users to update account information

1. The system shall allow a user to update their account information.
2. The user shall be allowed to view and change their name, mailing address, billing address, credit card type, credit card number, expiration date, and security code.
3. The user shall be able to change their password by entering the old one once, and a new one twice.

buy: allow users to add books to buy

1- System allows user to buy their books online by adding the book name and its details

2- System allows user to upload a book if he/she wants to buy a book.

Functional ADMIN Requirements:

Login: Administrators (web developers) will be required to login at all times

1. The system shall allow a registered admin to log-in to their account.
2. The system shall require a username and password from the admin.
3. The system will verify the username and password, and the admin will be considered logged-in.

View books: Admin can view all the added books online with their details

View transactions: System allows admin to view all the transaction details of buying and selling a book.

View users: All the registered user details will be displayed to the admin

4) Non Functional Requirements:

It is essential for this system to conform to user's needs and demands. Requirements Analysis produced the following non-functional requirements.

Performance Requirements:

• System login shall take less than max 5 seconds.

• Orders shall be processed within max 7 seconds.

• System shall support approximately 5000 simultaneous users.

Reliability

The average time to failure shall be minimum of 30 days. In the event that a server does crash, a backup server will be up and running within an hour.

Availability

The Online Book Reselling shall be available to users 24 hours a day, 7 days a week, with the exception of being down for maintenance no more than one hour a week. If the system crashes, it should be back up within one hour

Security

Users will be able to access only their own personal information and not that of other users. Purchases will be handled through a secure server to ensure the protection of user's credit card and personal information.

Maintainability

Any updates or defect fixes shall be able to be made on server-side computers only without any patches required by the user.

The System Must be Easy to Use

Users should be able to learn this system very quickly. It should be fairly intuitive. The reports should be clear and precise, and not overly technical.

The System Should not be overly technical

The information produced by the system should be understandable by anyone with general knowledge. Details should be written in a very understandable fashion.

5) Domain Requirements Specification: www.onlinebookreselling.com

6) Design & Implementation Constraints:

1- easy to use

2- attractive interface

3- max response time 7s

4- language can be written

5- skilled programmers

7) System Evolution:

Anticipated changes: total users will increase

How should any anticipated changes in the future: due to users increasing we will need more powerful hardware, continuous maintain and virus defender.

8) requirements discovery: all Design & Implementation Constraints

9) requirements validation techniques:

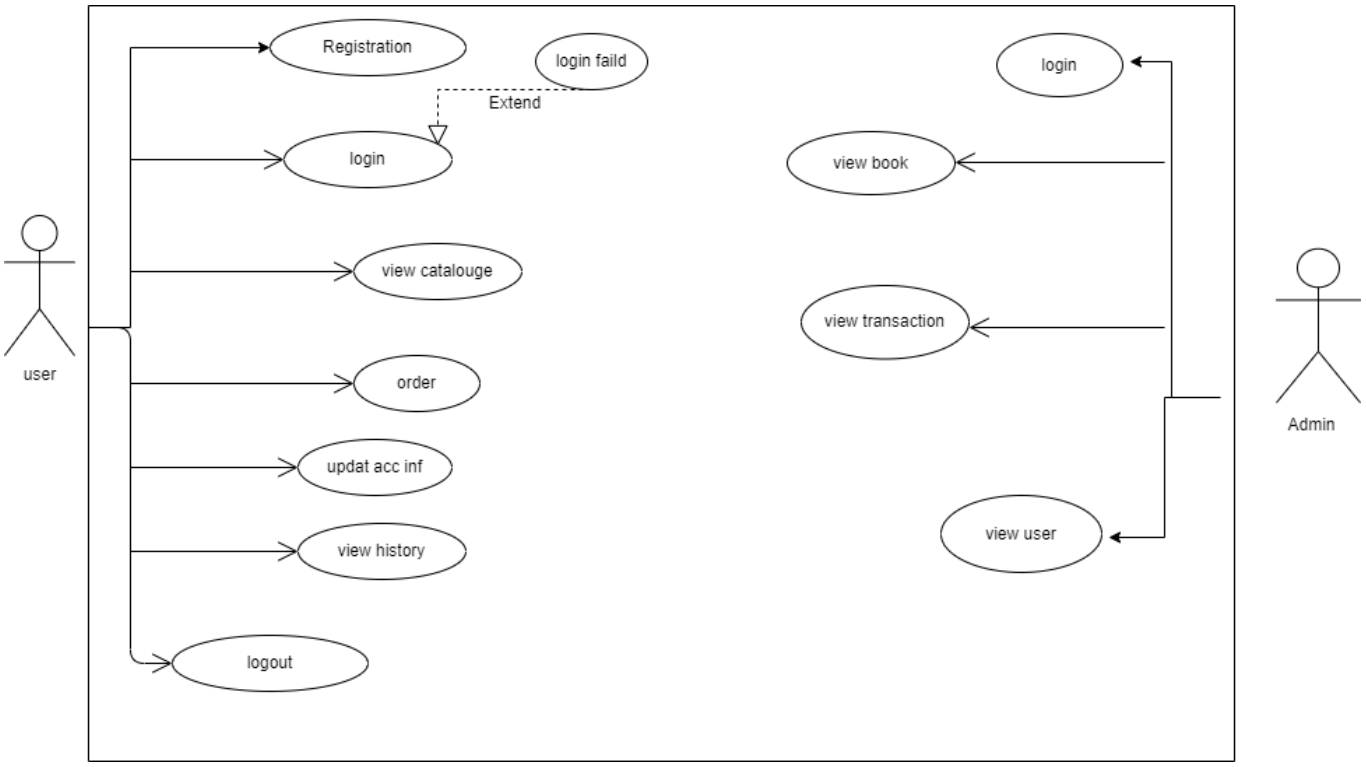
Test case generation: writing SQL test cases for verifying and testing database functionalities

Prototyping: the form of the product's interface, front-end design.

PART 2: System Design & Models:

10) Functional Diagrams:

A) Use-Case Diagram:



B) Detailed USER Use-Cases Description:

1) Register

Purpose	If the user doesn't have an account then he will be asked to register
Actor	User
Input	: The user will enter details in the registration form according to the required fields. The fields include: 1. Username 2. Password 3. confirm password 4.Email
Output	After registration the user will be directed to the main home page

2) Login (USER)

Purpose	If the user wants to get access to all the functionalities of Online Book Store he should login using his username and password.
Actor	User
Input	The user will enter his username and password
Output	If it is a successful login the user will be directed to the main home page. Else if the user enters invalid information he will be asked to re enter information

3) Login failed

Purpose	If there are any problem in logging return massage
Actor	User
Input	Username and password
Output	Incorrect username or password

4) catalouge

Purpose	A user can search for a book of his choice by selecting category and title. Then a select query is used to retrieve data from the database and display the selected information.
Actor	User
Input	The user will select a category and enter title in a text box provided
Output	The system will display the books which matches the selected search criteria

5) Order

Purpose	Start buying books process
Actor	User
Input	Order request
Output	the new book will be inserted in the website

6) Update Acc info

Purpose	If the user wants to change his personal account information then he can update his/her selected fields and the entire data will be updated.
Actor	User
Input	The user will update his account information
Output	The system will update the entered information

7) Add Book

Purpose	If the user wants to add a book then he/she can upload a book
Actor	User
Input	If user wants to add a book the he/she should click the add book button and enter book details
Output	the new book will be inserted in the website

8) View history

Purpose	List transaction history
Actor	User
Input	Request history action
Output	Historical transaction

9) Clear session

Purpose	Secure system
Actor	User

Input	Logout event
Output	Cleared session

Detailed ADMIN Use-Cases Description:

1) Login

Purpose	If the user wants to end his session and sign out of the website then he can use the logout option
Actor	Admin
Input	The user will click the logout button
Output	The user's account session comes to an end and he should login again if he wants to enter into the website.

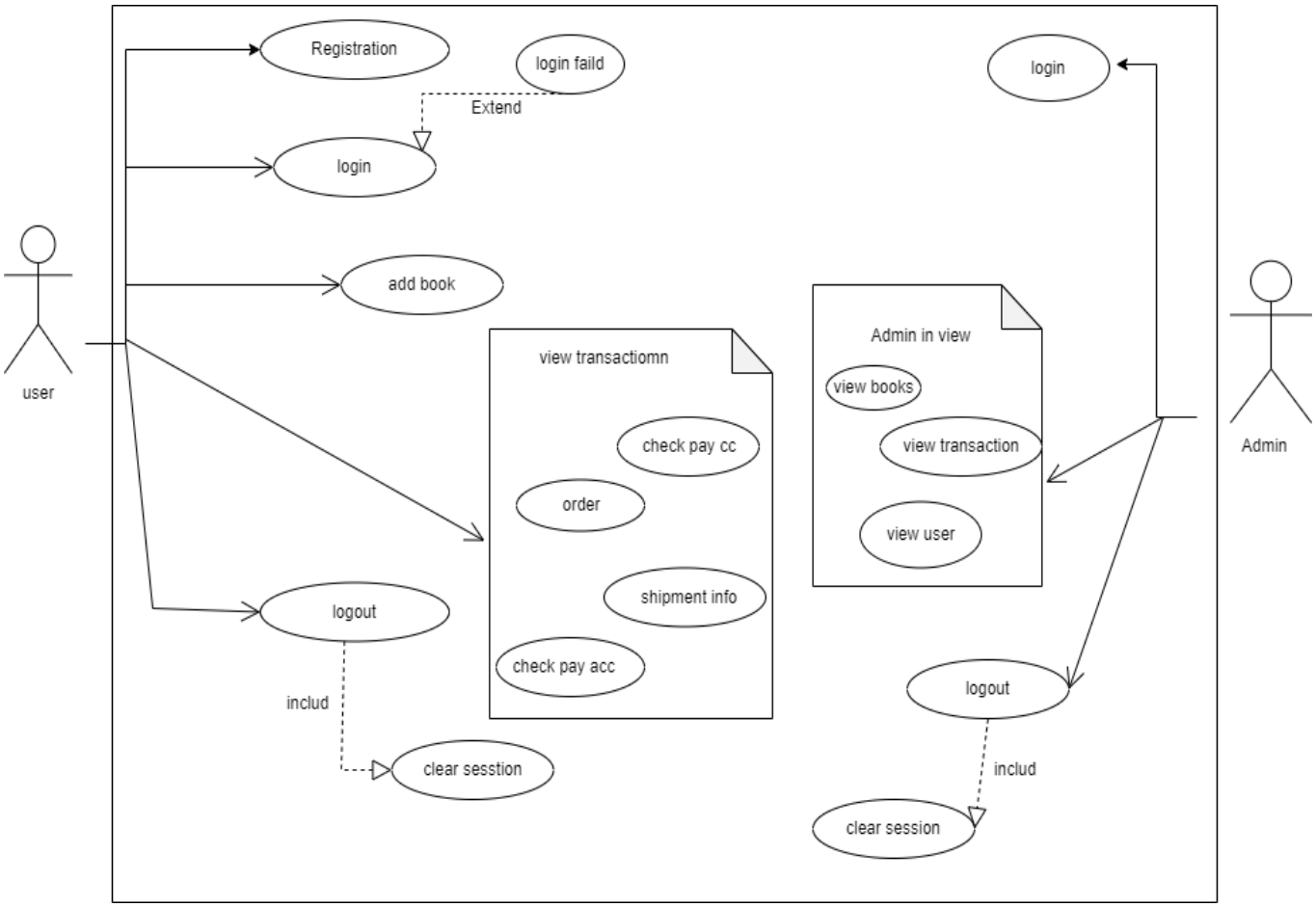
2) View books

Purpose	If the user wants to end his session and sign out of the website then he can use the logout option
Actor	Admin
Input	The user will click the logout button
Output	The user's account session comes to an end and he should login again if he wants to enter into the website.

3) View users

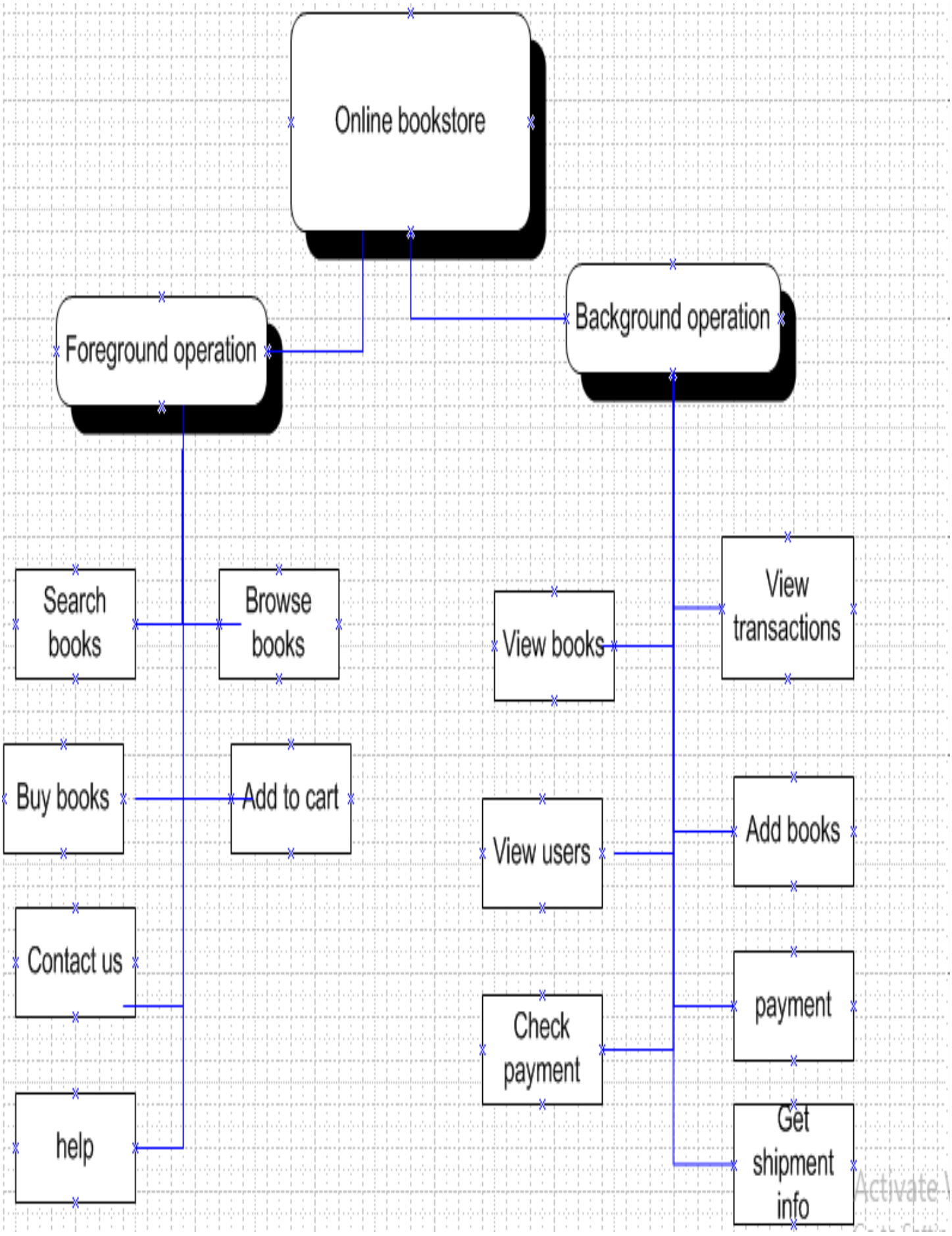
Purpose	If the user wants to end his session and sign out of the website then he can use the logout option
Actor	Admin
Input	The user will click the logout button
Output	The user's account session comes to an end and he should login again if he wants to enter into the website.

c) Package Diagram grouping relevant Use-Cases into Packages:

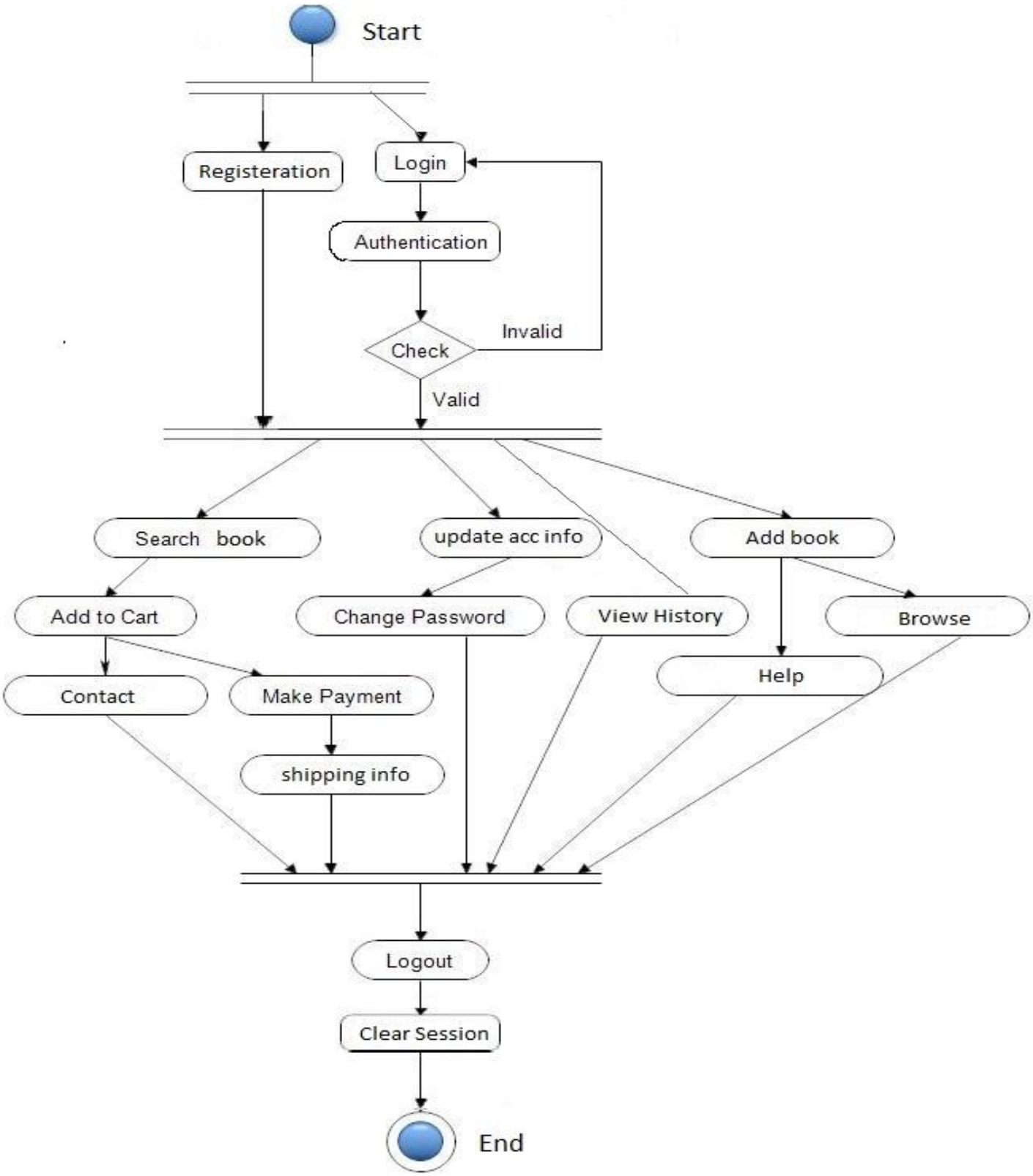


11)Structural & Behavioural Diagrams:

a)system Architecture:



b) Activity Diagrams:



Design Pattern:

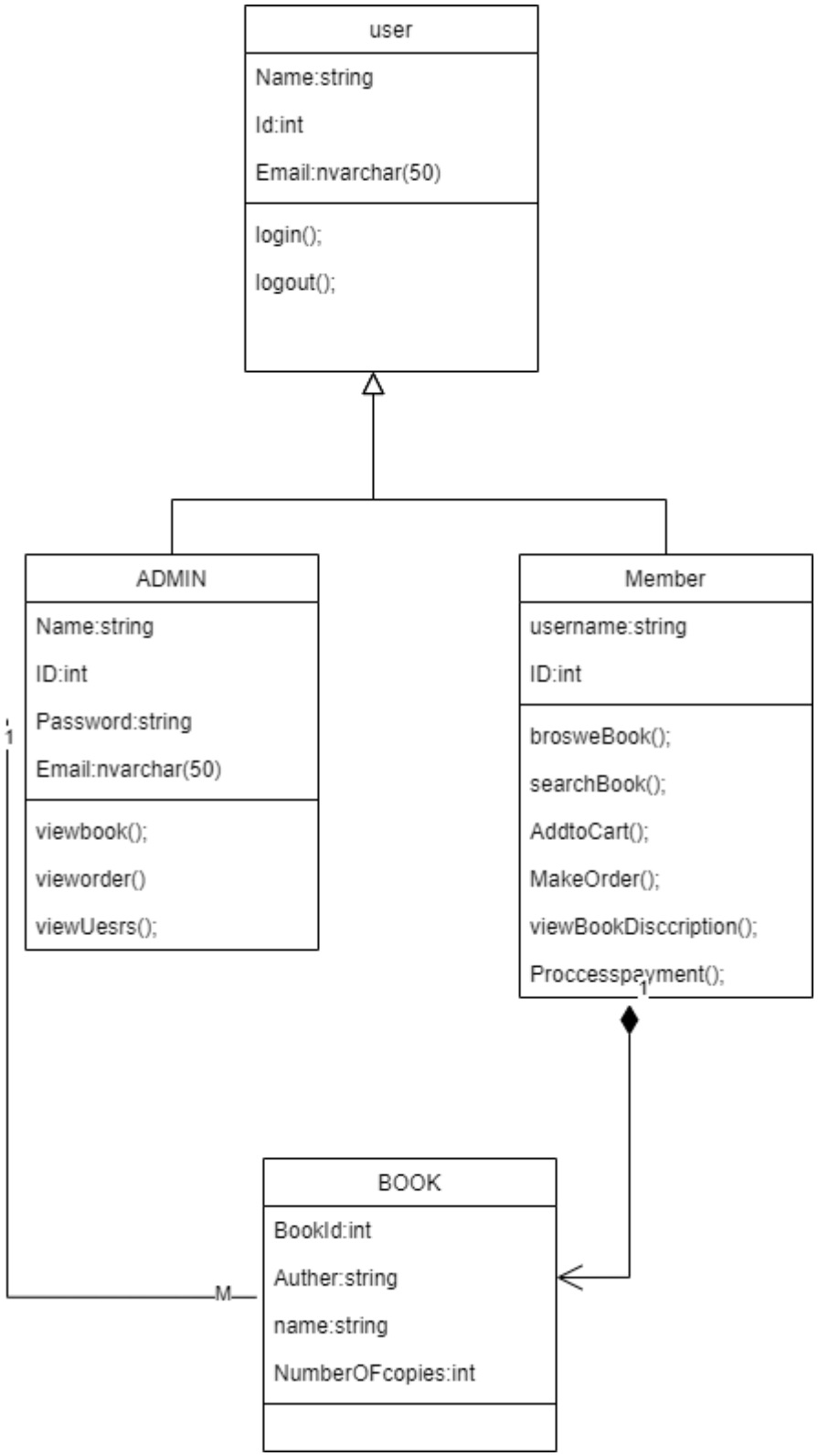
MVC pattern architecture gives us the idea of separation of concern, it helps us to implement the separation of concern among the model, view and controller classes within applications.

Separation of concern makes it easy for us to test our application as relation among different components of application is clearer and coherent. MVC help us to implement a test-driven development approach, in which we implement automated test cases before we write the code. These unit test cases help us predefine and verify requirements of new code before writing it.

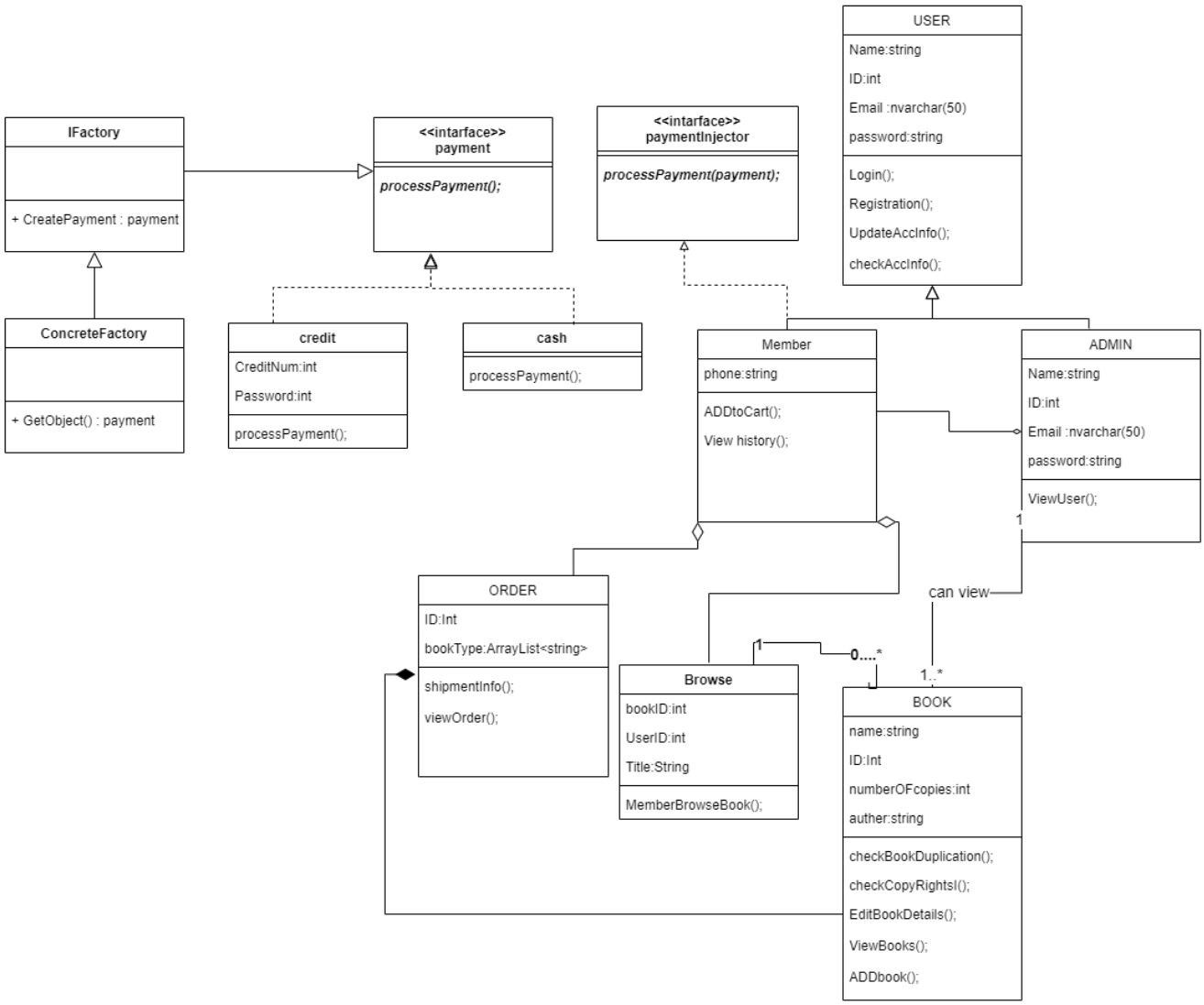
If we are making an application with enough serious stimulating on the client side to refuse to go along with JavaScript alone. If we are developing an application which have a very high lifting on the server side and a little communication on the client side then we should not use the MVC pattern architecture, instead we should use simple setup such as web-based form model

Class diagram:

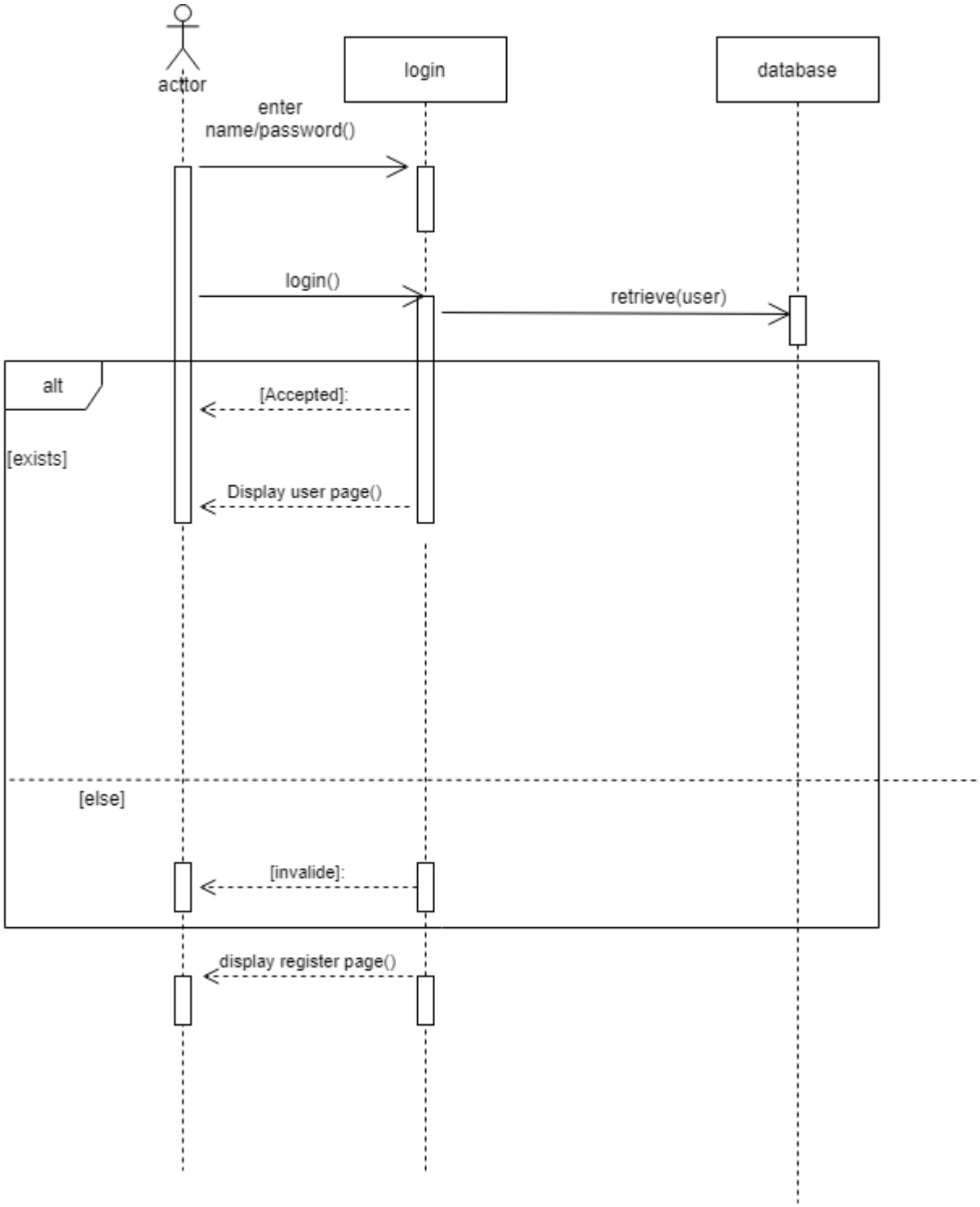
Version1:-

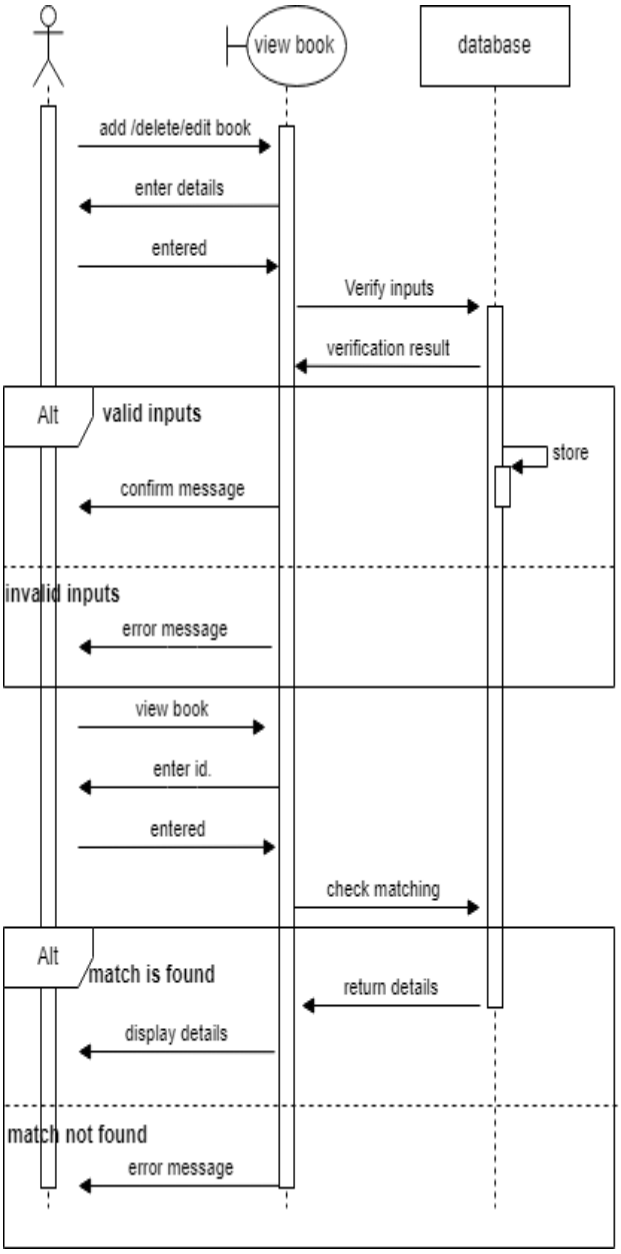

user
Name:string
Id:int
Email:nvarchar(50)
login();
logout();
ADMIN
Name:string
ID:int
Password:string
Email:nvarchar(50)
viewbook();
vieworder()
viewUesrs();
Member
username:string
ID:int
brosweBook();
searchBook();
AddtoCart();
MakeOrder();
viewBookDisccription();
Proccesspayment();
1
1
M
BOOK
BookId:int
Auther:string
name:string
NumberOFcopies:int

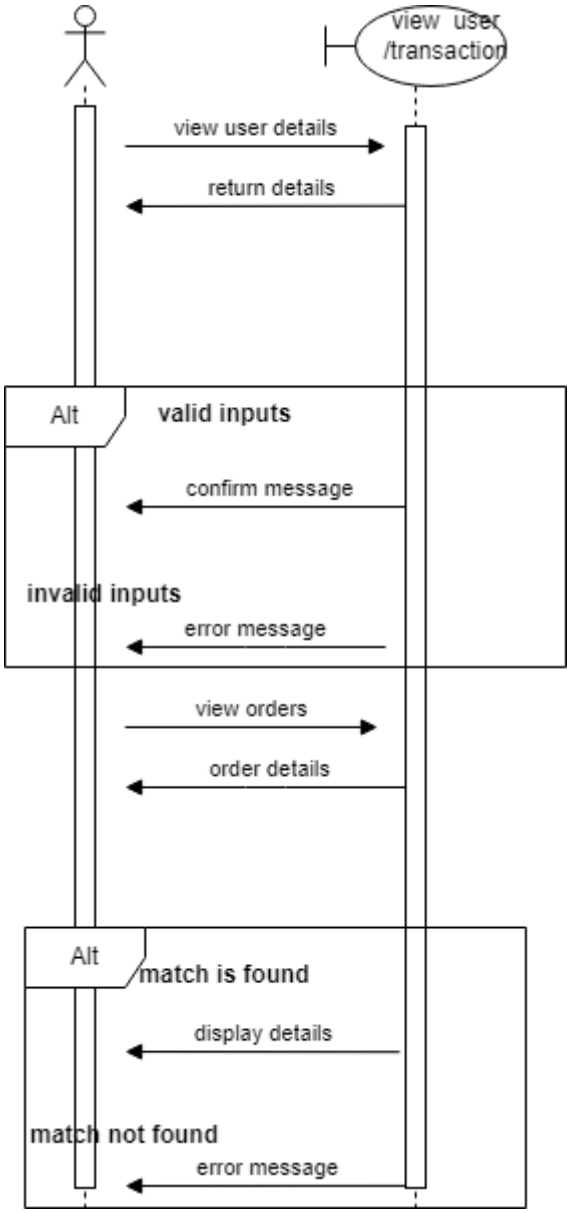
Version3:-

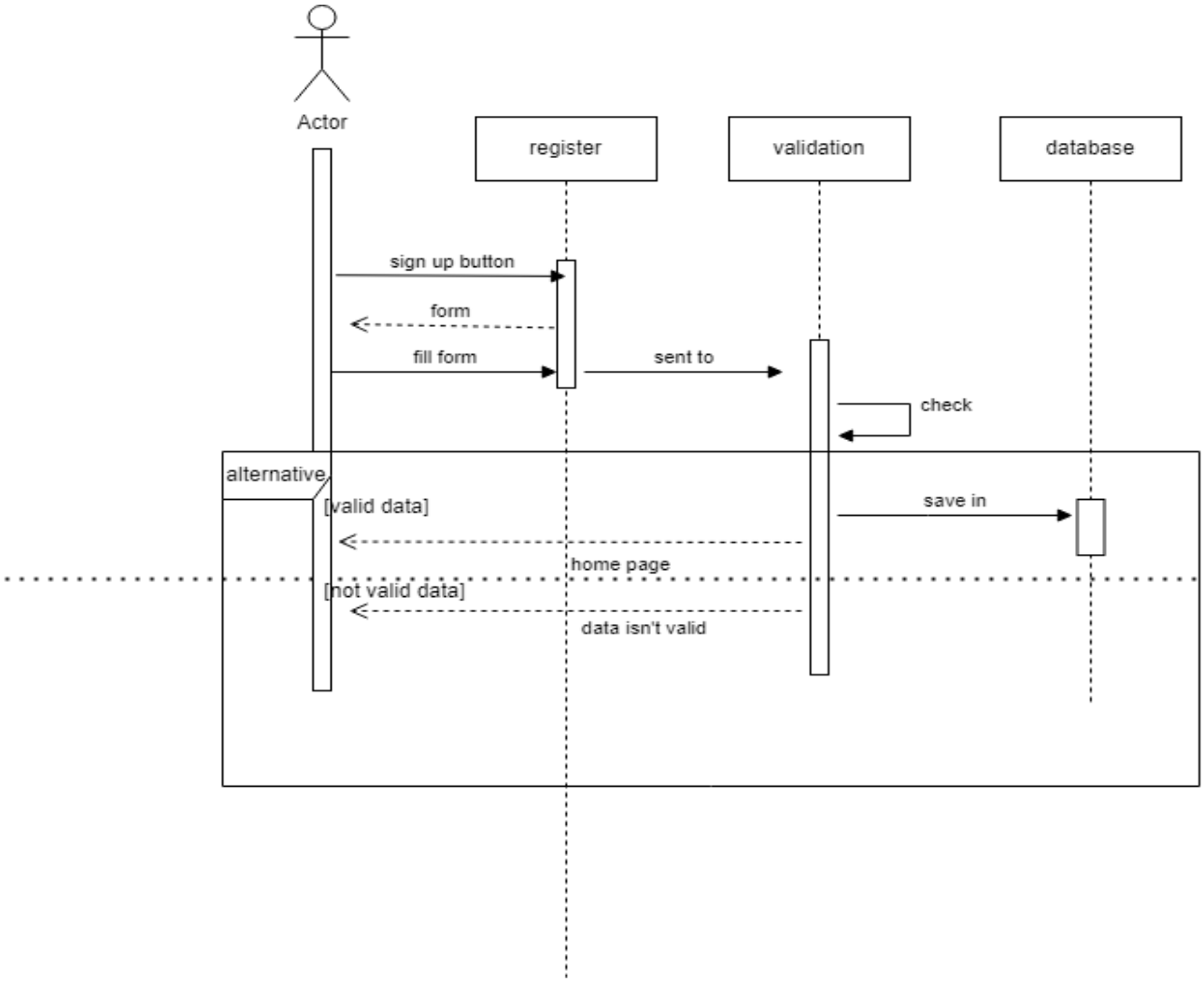

0..1
USER
Name:string
ID:int
Email :nvarchar(50)
password:string
Login();
Registration();
UpdateAccInfo();
checkAccInfo();
IFactory
+ CreatePayment : payment
<<intarface>>
payment
processPayment();
<<intarface>>
paymentInjector
processPayment(payment);
ConcreteFactory
+ GetObject() : payment
credit
CreditNum:int
Password:int
processPayment();
cash
processPayment();
Member
phone:string
ADDtoCart();
View history();
ADMIN
Name:string
ID:int
Email :nvarchar(50)
password:string
ViewUser();
1
ORDER
ID:Int
bookType:ArrayList<string>
shipmentInfo();
viewOrder();
can view
1
0....*
1..*
Browse
bookID:int
UserID:int
Title:String
MemberBrowseBook();
BOOK
name:string
ID:Int
numberOFcopies:int
auther:string
checkBookDuplication();
checkCopyRightsI();
EditBookDetails();
ViewBooks();
ADDbook();

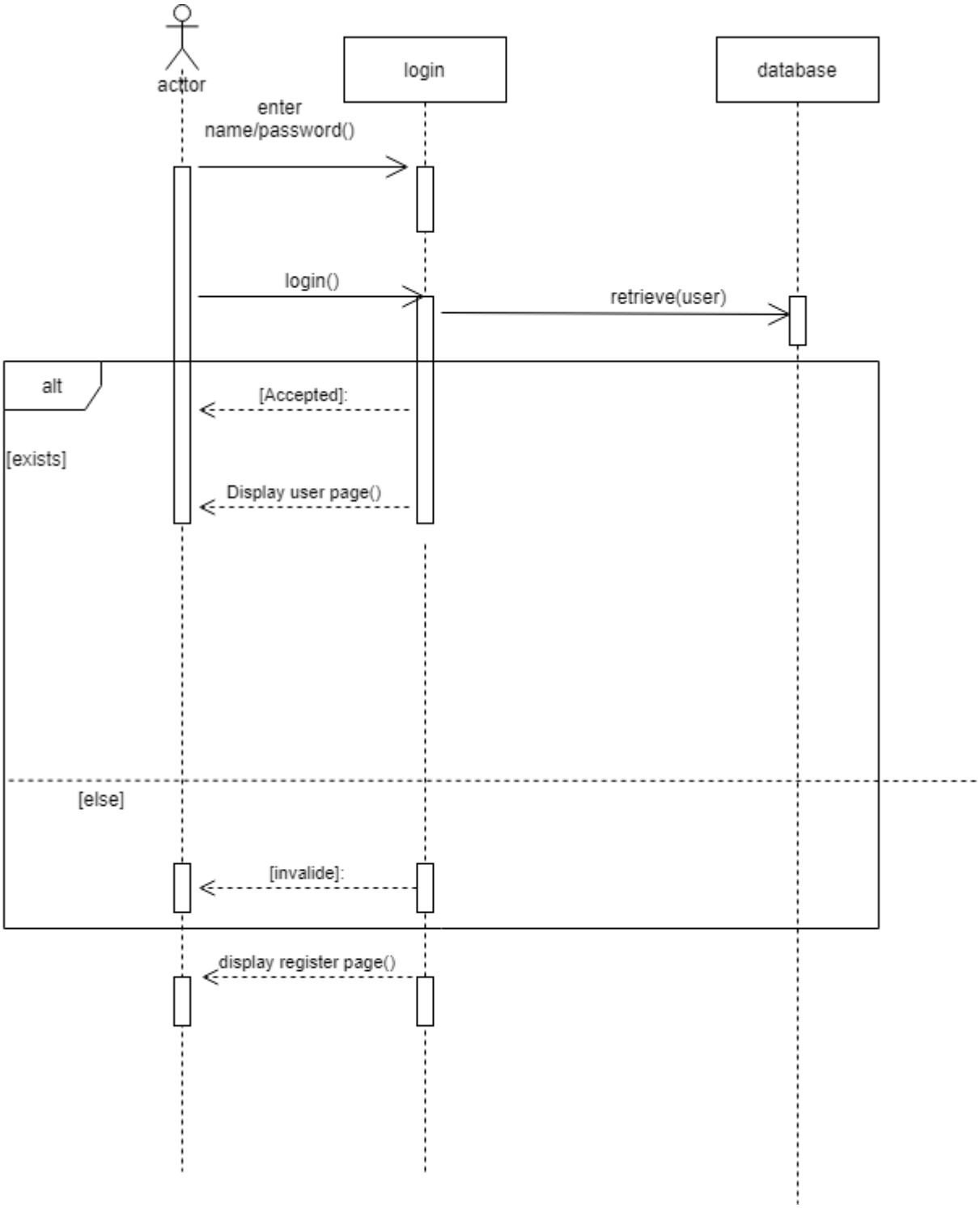
Sequence Diagram:

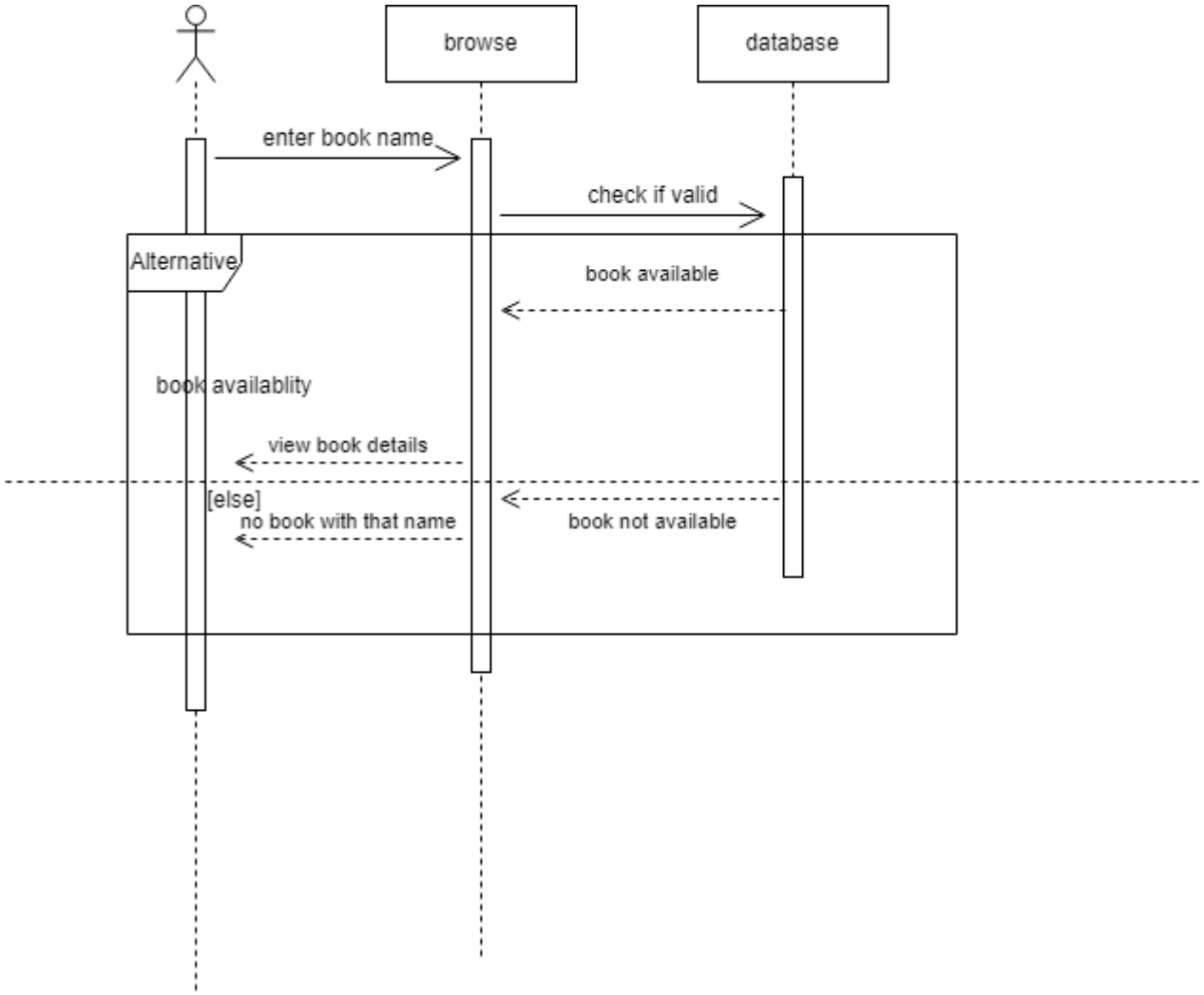


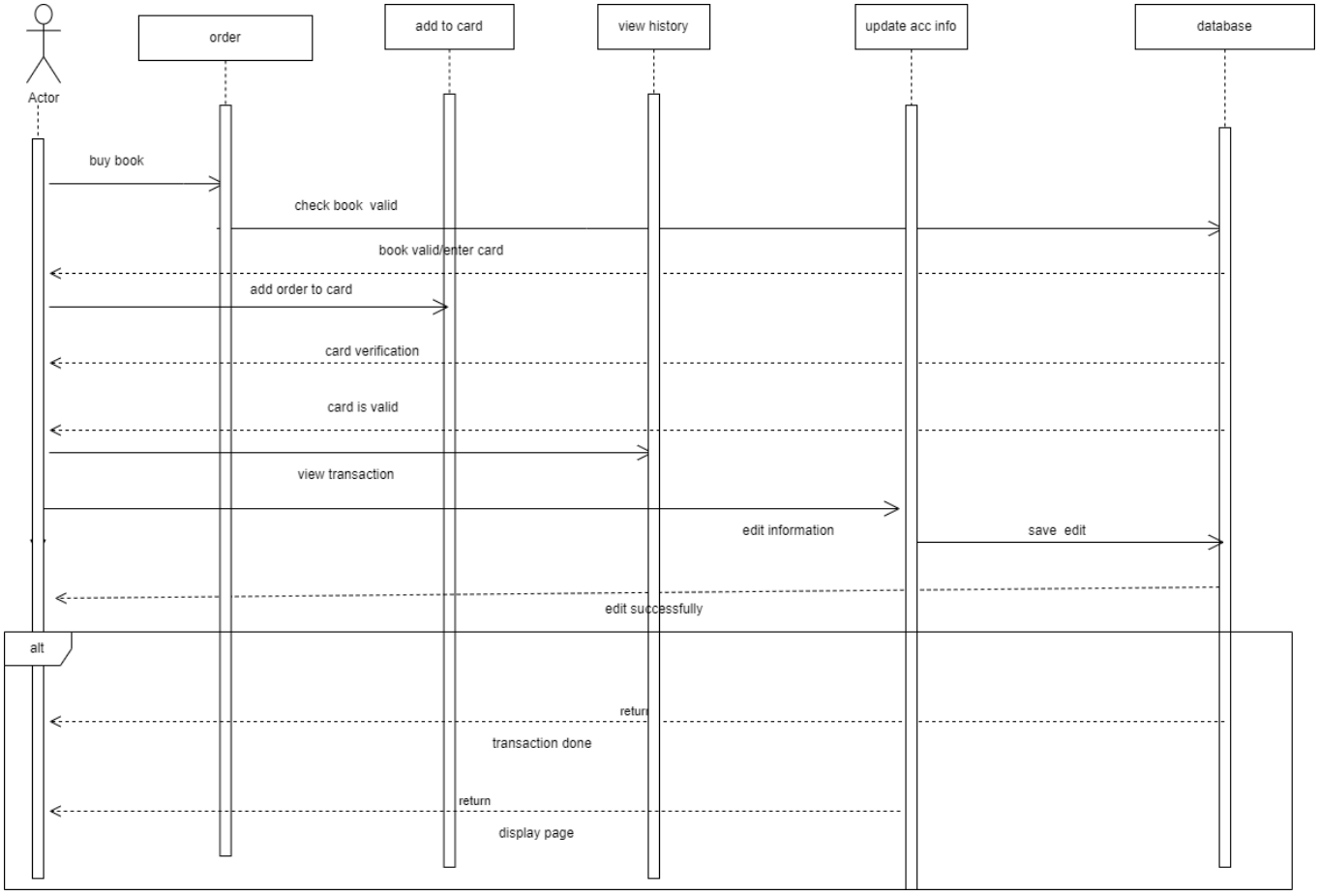

view book
database
add /delete/edit book
enter details
entered
Verify inputs
verification result
Alt
valid inputs
store
confirm message
invalid inputs
error message
view book
enter id.
entered
check matching
Alt
match is found
return details
display details
match not found
error message

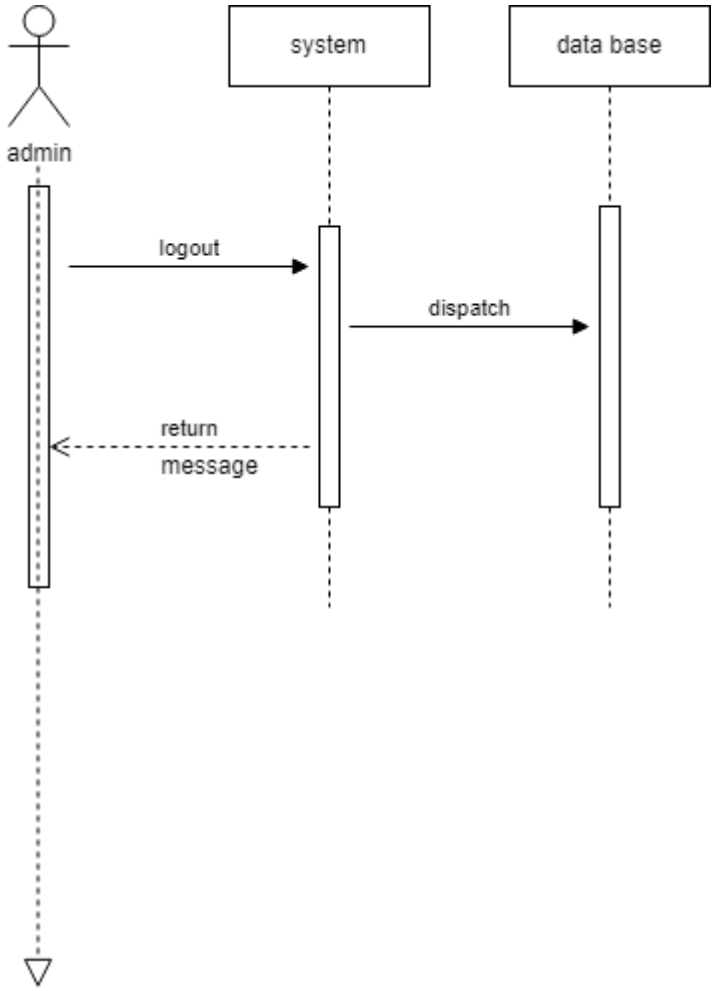

view user /transaction
view user details
return details
Alt
valid inputs
confirm message
invalid inputs
error message
view orders
order details
Alt
match is found
display details
match not found
error message


Actor
register
validation
database
sign up button
form
fill form
sent to
check
alternative
[valid data]
save in
home page
[not valid data]
data isn't valid

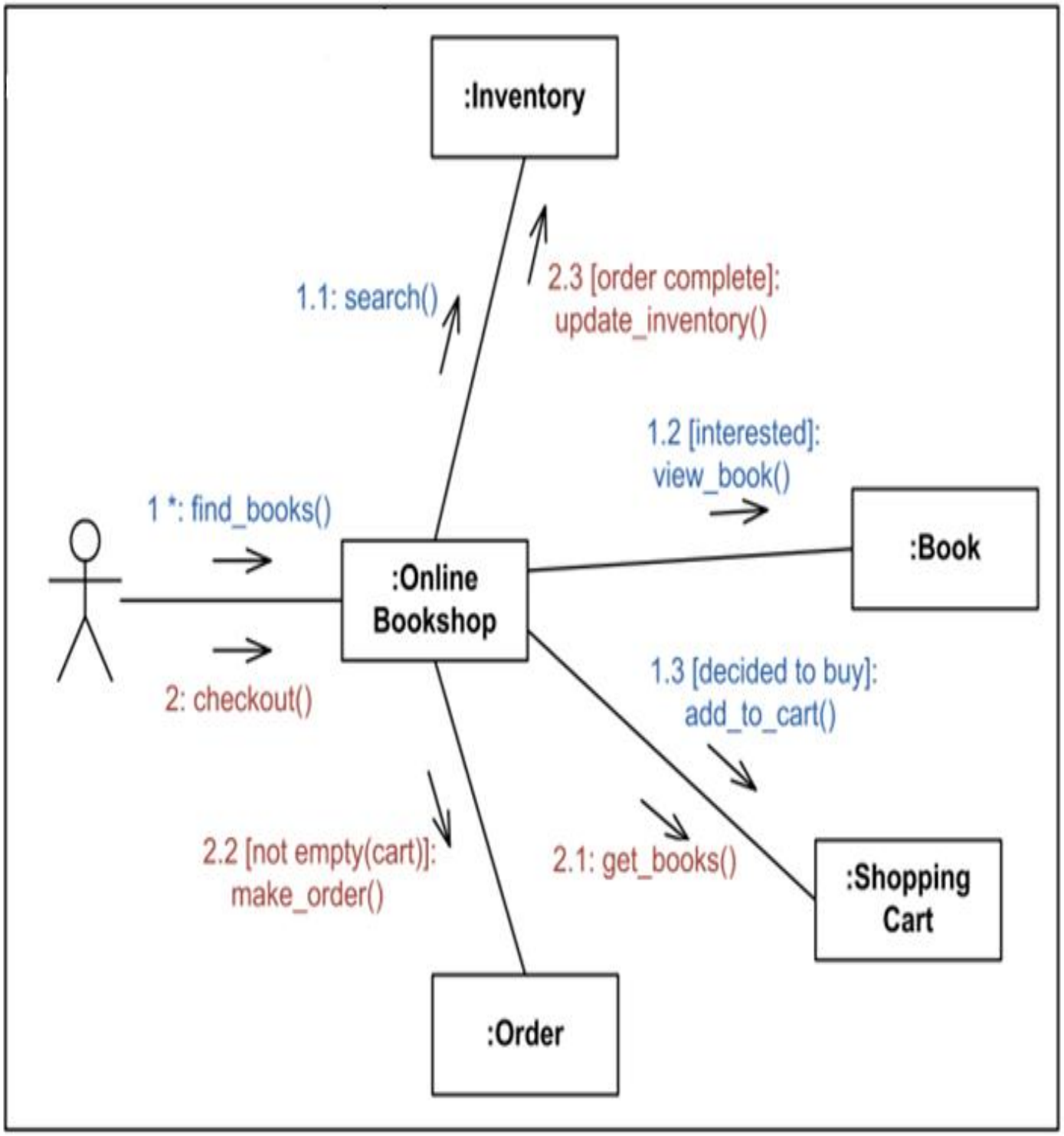

actor
login
database
enter
name/password()
login()
retrieve(user)
alt
[Accepted]:
[exists]
Display user page()
[else]
[invalide]:
display register page()


browse
database
enter book name
check if valid
Alternative
book available
book availablity
view book details
[else]
no book with that name
book not available

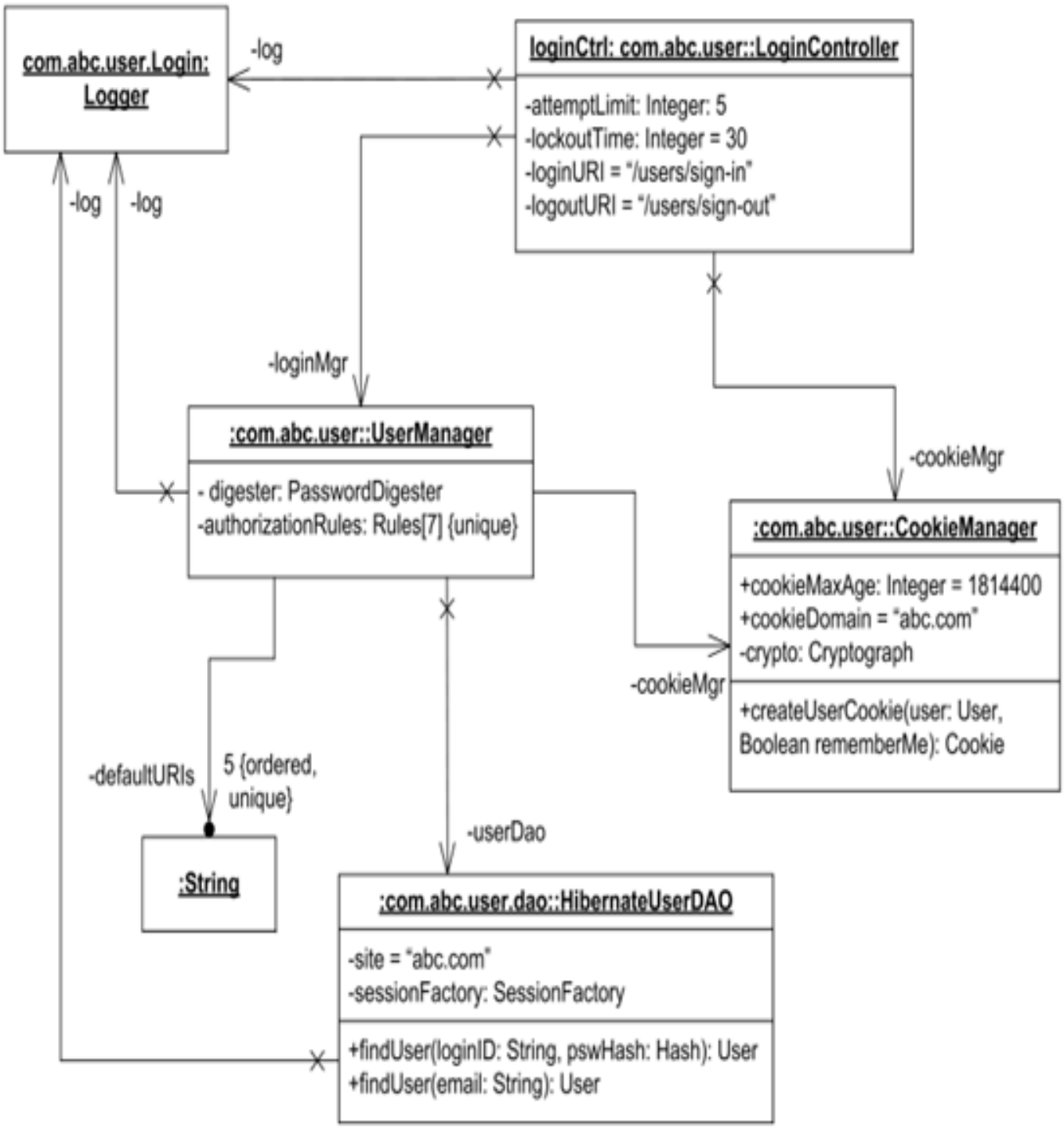

Actor
order
add to card
view history
update acc info
database
buy book
check book valid
book valid/enter card
add order to card
card verification
card is valid
view transaction
edit information
save edit
edit successfully
alt
return
transaction done
return
display page


admin
system
data base
logout
dispatch
return
message

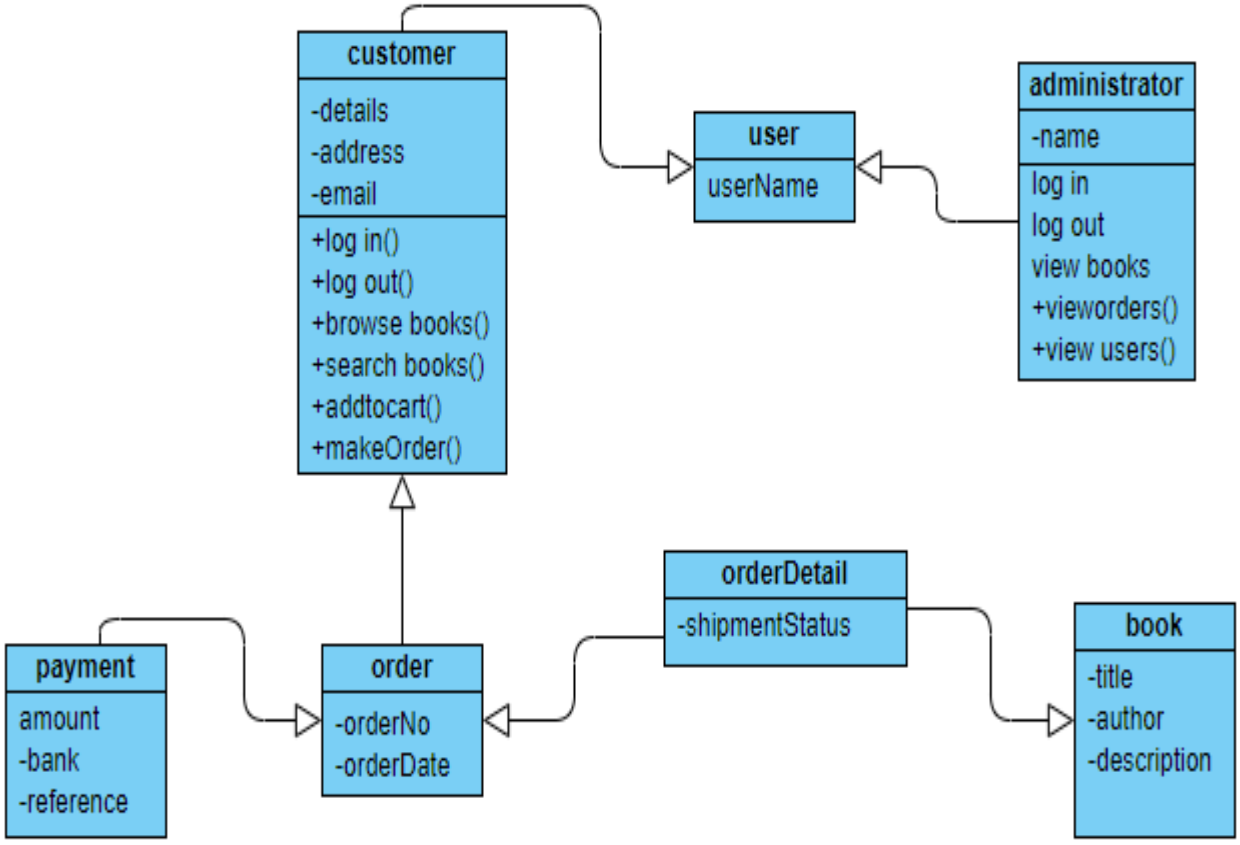
Communication Diagram:



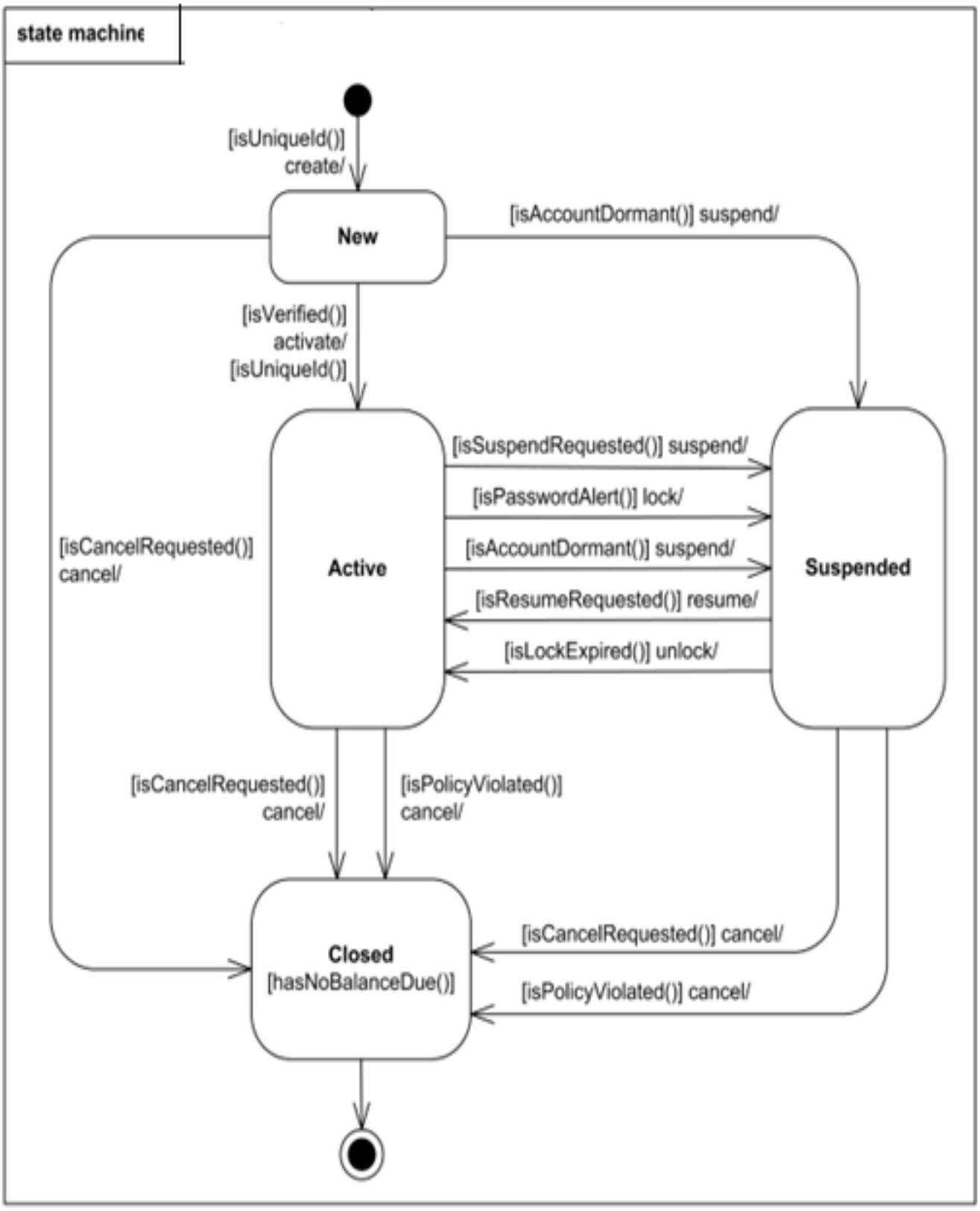
Object Diagrams:



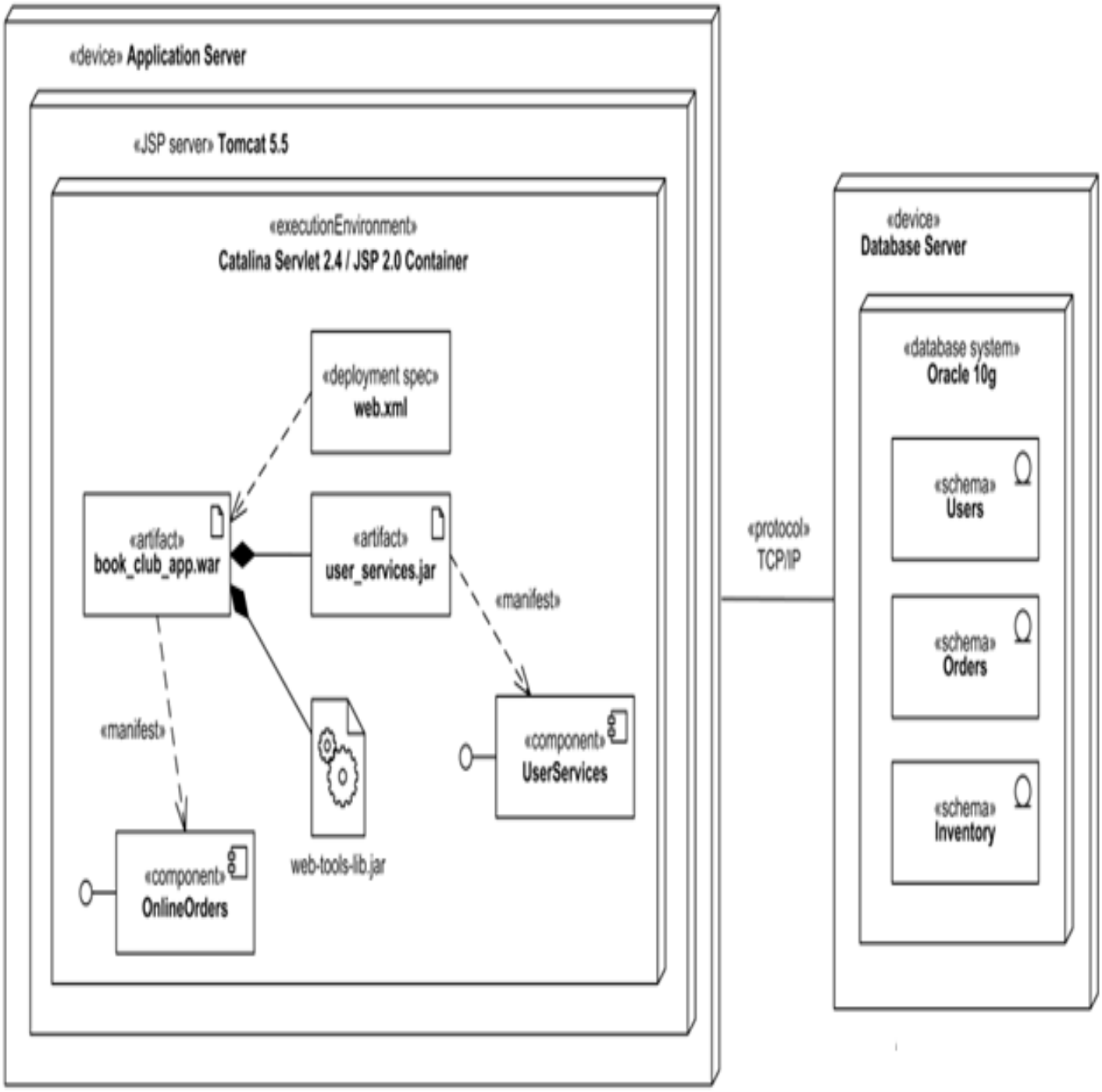
Database Specification:



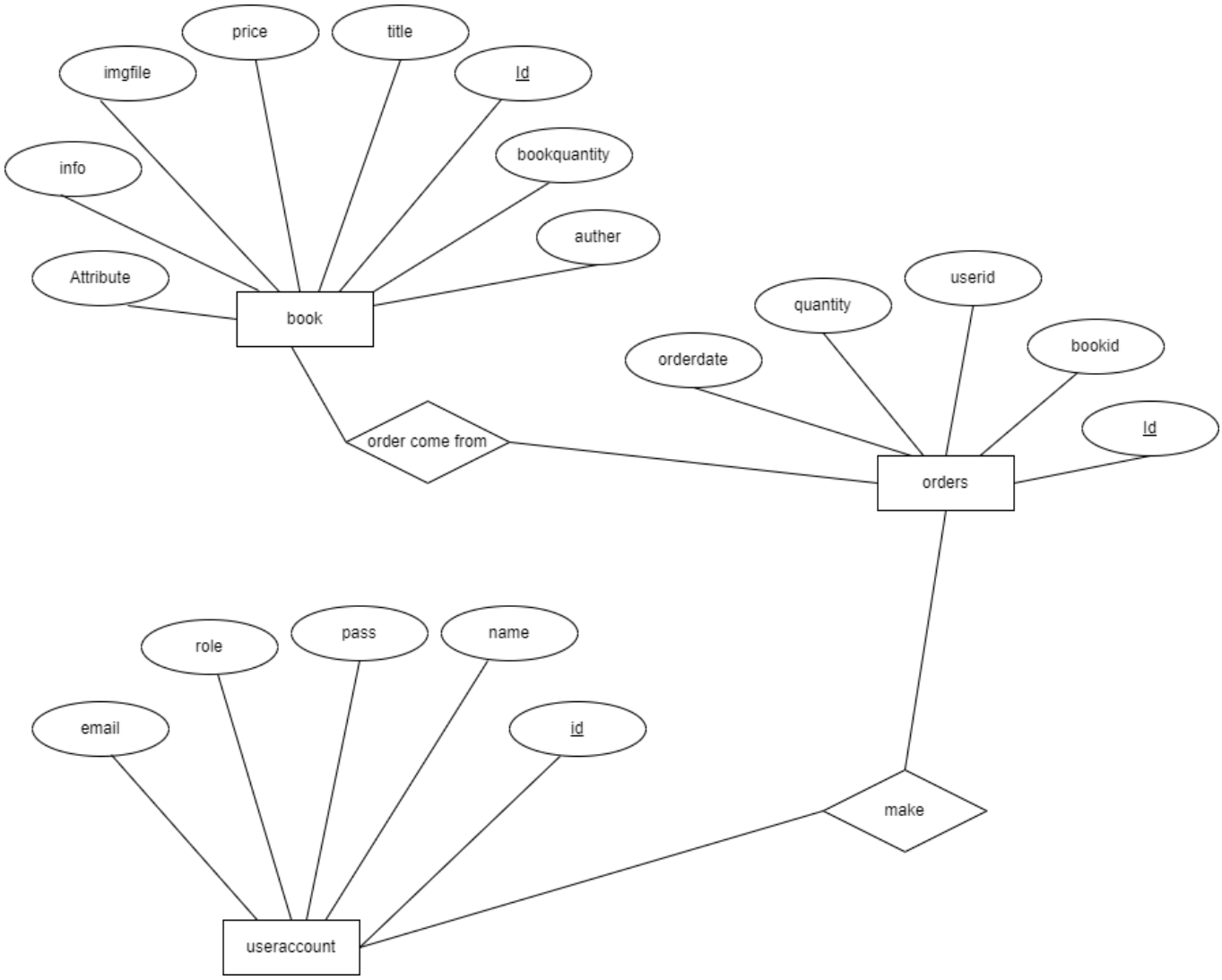
State-Machine Diagram:



Deployment diagram:



Entity Relationship Database (ERD):



DataBase schema:-