LIST OF EXPERIMENTS

Experiment 1:
Familiarization with Rational Rose or Umbrella environment

Experiment 2:
a) Identify and analyze events
b) Identify Use cases
c) Develop event table

Experiment 3:
a) Identify & analyze domain classes
b) Represent use cases and a domain class diagram using Rational Rose
c) Develop CRUD matrix to represent relationships between use cases and problem domain classes

Experiment 4:
a) Develop Use case diagrams
b) Develop elaborate Use case descriptions & scenarios
c) Develop prototypes (without functionality)

Experiment 5:
a) Develop system sequence diagrams and high-level sequence diagrams for each use case
b) Identify MVC classes / objects for each use case
c) Develop Detailed Sequence Diagrams / Communication diagrams for each use case showing interactions among all the three-layer objects

Experiment 6:
a) Develop detailed design class model (use GRASP patterns for responsibility assignment)
b) Develop three-layer package diagrams for each case study

Experiment 7:
a) Develop Use case Packages
b) Develop component diagrams
c) Identify relationships between use cases and represent them
d) Refine domain class model by showing all the associations among classes

Experiment 8:
Develop sample diagrams for other UML diagrams - state chart diagrams, activity diagrams and deployment diagrams

Experiment 1:

Familiarization with Rational Rose or Umbrella environment

UML BASICS

 Unified Modeling Language

 UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems

Structural Things:

Graphical notations used in structural things are the most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

 Classes

 Object

 Interface

 Collaboration

 Use case

 Active classes

 Components

 Nodes

Class Notation:

UML class is represented by the diagram shown below. The diagram is divided into four parts.

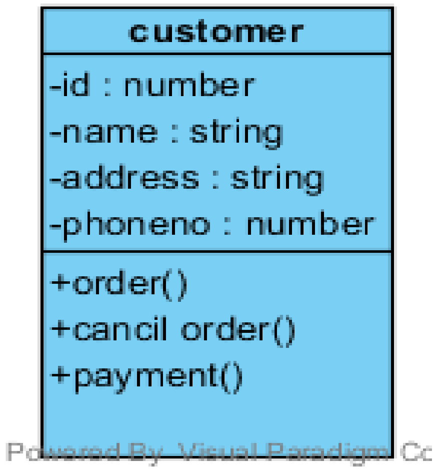
 The top section is used to name the class.

 The second one is used to show the attributes of the class.

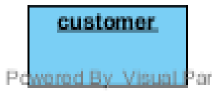
 The third section is used to describe the operations performed by the class.

 The fourth section is optional to show any additional components.

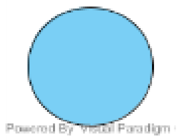
 Classes are used to represent objects. Objects can be anything having properties and responsibility.



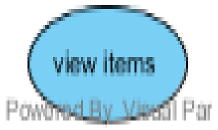
Object Notation:
The object is represented in the same way as the class. The only difference is the name which is underlined as shown below:
 As object is the actual implementation of a class which is known as the instance of a class. So it has the same usage as the class.



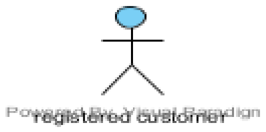
Interface Notation:
Interface is represented by a circle as shown below. It has a name which is generally written below the circle. Interface is used to describe functionality without implementation. Interface is the just like a template where you define different functions not the implementation. When a class implements the interface it also implements the functionality as per the requirement.



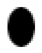
Use case Notation:
Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.
Use case is used to capture high level functionalities of a system.



Actor Notation:
An actor can be defined as some internal or external entity that interacts with the system.
Actor is used in a use case diagram to describe the internal or external entities.



Initial State Notation:
Initial state is defined to show the start of a process. This notation is used in almost all diagrams.



Initial state

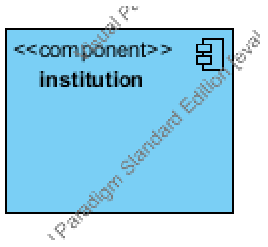
The usage of Initial State Notation is to show the starting point of a process.

Final State Notation:
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.

Final state

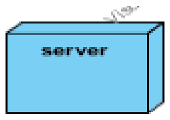
The usage of Final State Notation is to show the termination point of a process.

Component Notation:
A component in UML is shown as below with a name inside. Additional elements can be added wherever required.



Component is used to represent any part of a system for which UML diagrams are made.

Node Notation:
A node in UML is represented by a square box as shown below with a name. A node represents a physical component of the system.

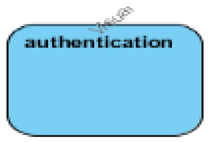


Node Node is used to represent physical part of a system like server, network etc.

Behavioural Things:
Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non software systems. These features include interactions and state machines. Interactions can be of two types:

- Sequential (Represented by sequence diagram)
- Collaborative (Represented by collaboration diagram)

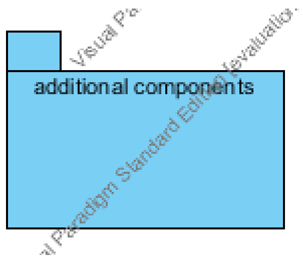
Activity Notation:



State machine Notation:
State machine describes the different states of a component in its life cycle. The notation is described in the following diagram.
State machine is used to describe different states of a system component. The state can be active, idle or any other depending upon the situation.

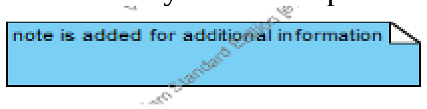
Grouping Things:
Organizing the UML models are one of the most important aspects of the design. In UML there is only one element available for grouping and that is package.

Package Notation:
Package notation is shown below and this is used to wrap the components of a system.



Package

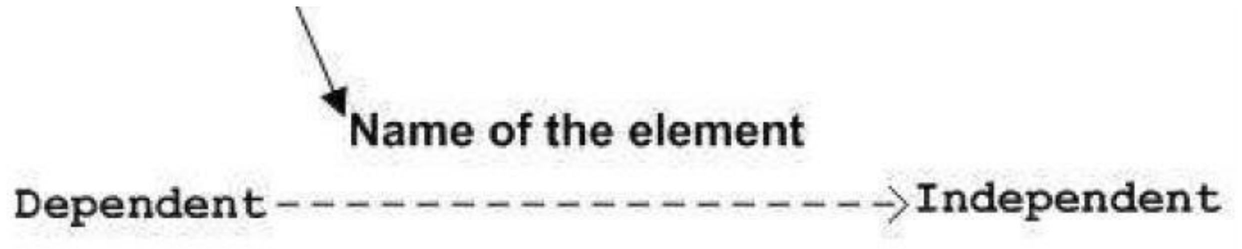
Annotational Things:
In any diagram explanation of different elements and their functionalities are very important. So UML has notes notation to support this requirement.
Note Notation: This notation is shown below and they are used to provide necessary information of a system.



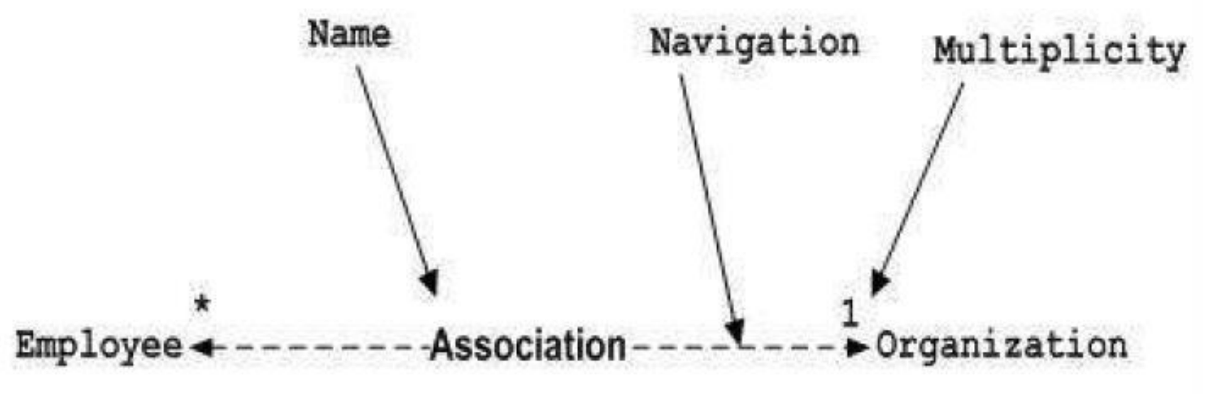
Relationships:
A model is not complete unless the relationships between elements are described properly. The Relationship gives a proper meaning to an UML model. Following are the different types of relationships available in UML.
- Dependency
- Association
- Generalization
- Extensibility

Dependency Notation:

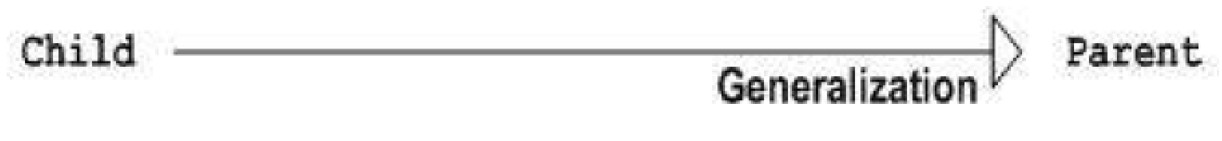
Dependency is an important aspect in UML elements. It describes the dependent elements and the direction of dependency.
Dependency is represented by a dotted arrow as shown below. The arrow head represents the independent element and the other end the dependent element. Dependency is used to represent dependency between two elements of a system.



Association Notation:
Association describes how the elements in an UML diagram are associated.
In simple word it describes how many elements are taking part in an interaction. Association is represented by a dotted line with without arrows on both sides.
The two ends represent two associated elements as shown below. The multiplicity is also mentioned at the ends 1, ∗ etc to show how many objects are associated.
Association is used to represent the relationship between two elements of a system.



Generalization Notation:
Generalization describes the inheritance relationship of the object oriented world.
It is parent and child relationship. Generalization is represented by an arrow with hollow arrow head as shown below.



One end represents the parent element and the other end child element. Generalization is used to describe parent-child relationship of two elements of a system.
Extensibility Notation: All the languages programming or modeling have some mechanism to extend its capabilities like syntax, semantics etc. UML is also having the following mechanisms to provide extensibility features.

- Stereotypes(Represents new elements)
- Tagged values (Represents new attributes)
- Constraints (Represents the boundaries)

- **Extensibility notations** are used to enhance the power of the language. It is basically additional elements used to represent some extra behaviour of the system. These extra behaviours are not covered by the standard available notations

Experiment 2:

a) Identify and analyze events
b) Identify Use cases
c) Develop event table

Take an Example of Online Shopping

a) Identify and analyze events

- **View Items**
  - **-Search Items**
  - **-Browse Items**
  - **-View Recommended Items**
  - **-Add Shopping cart**
  - **-Add Wish List**

- **Check Out**

  - **-Customer Authentication**
  - **-View/Update Shopping cart**
  - **-Calculate Taxes and shipping**
  - **-Payment**

- **Customer Authentication**

  - **-Usersign-In**
  - **-Remember Me**

b) Identify Use cases

- **View Items**
- **Check Out**
- **Make Purchase**
- **Client Register**
- **Login**
- **Logout**
- **View order status**
- **Request order**
- **Finish Order**
- **Payment**
- **Order cancellation**
- **Change account information**
- **Search Products**

c) Develop event table

	Customer	Order	Product	Database
Customer registered	+			
Unsubscribed Customer	+			
Product Add to cart	*	*	*	
Product Remove from cart	*	*	*	
Place Order	*	+	*	*
Finish Order		*		
Product registration			+	*
Order Processing		*	*	*
Product Shipment			+	*
Product Delivery				*

* Event can occur zero to many times
+ Event can occur zero to one time

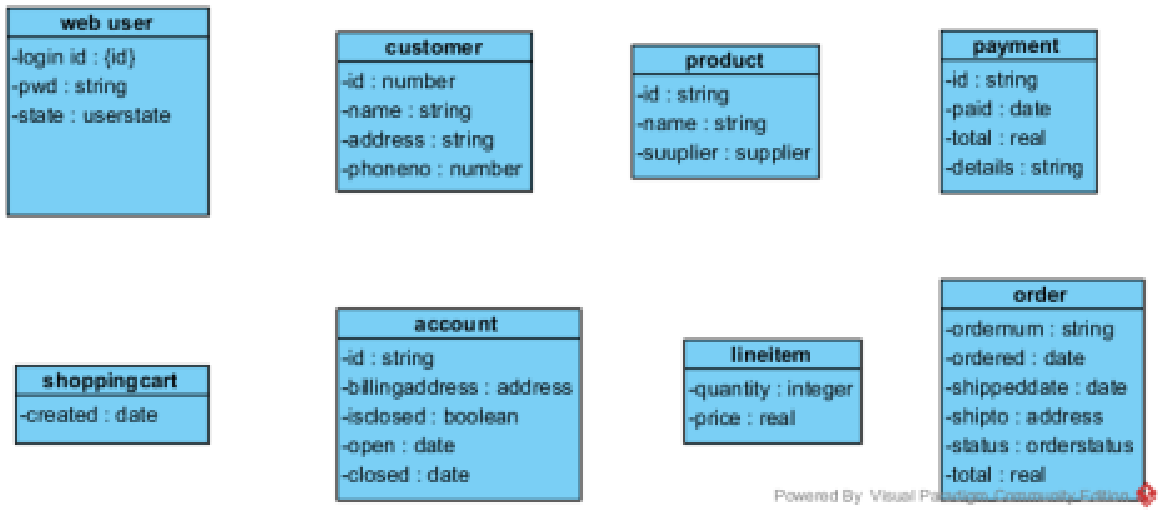
Experiment 3:

a) Identify & analyze domain classes

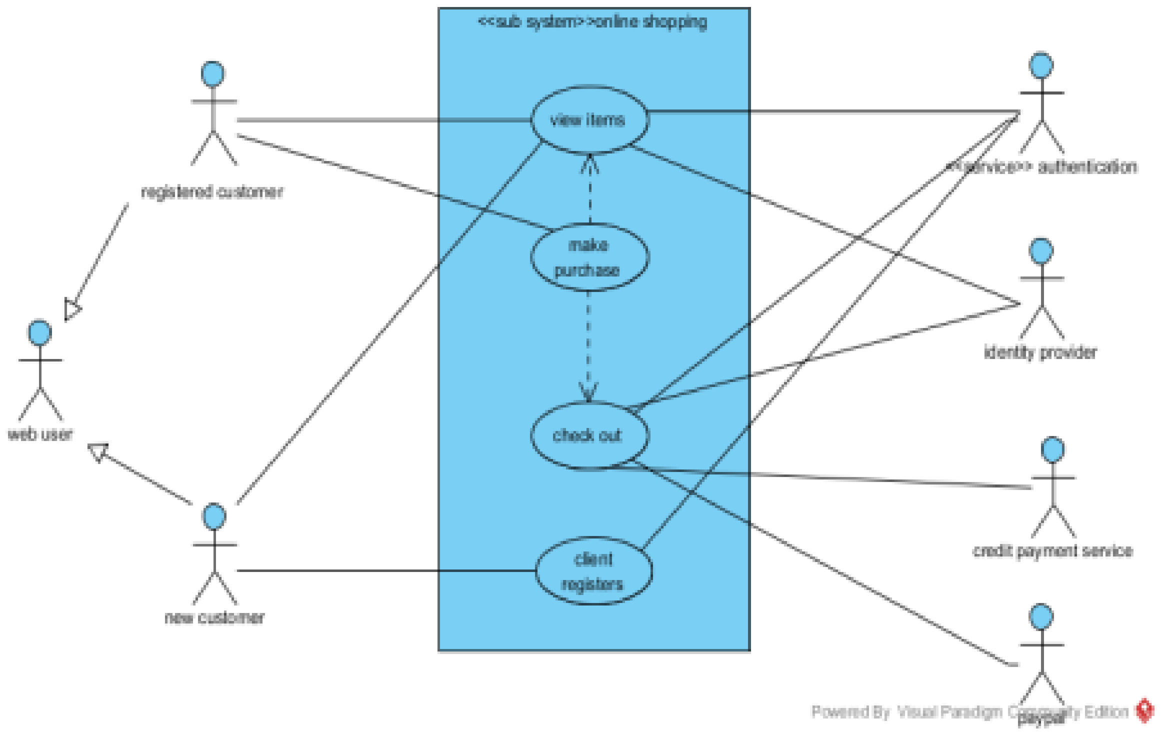
b) Represent use cases and a domain class diagram using Rational Rose

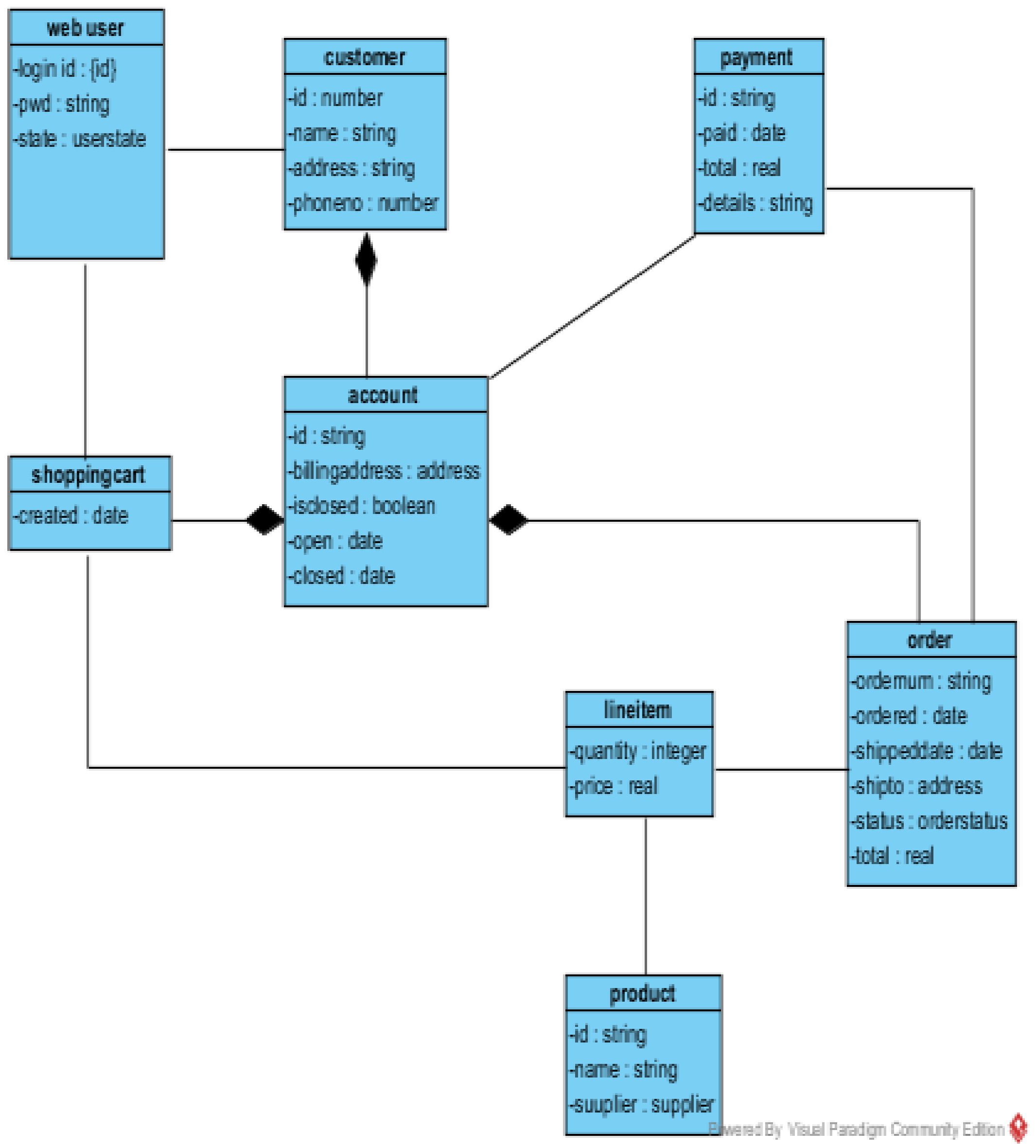
c) Develop CRUD matrix to represent relationships between use cases and problem domain classes

A) Identify & analyze domain classes



B) Represent use cases and a domain class diagram using Rational Rose




web user
-login id : {id}
-pwd : string
-state : userstate
customer
-id : number
-name : string
-address : string
-phoneno : number
payment
-id : string
-paid : date
-total : real
-details : string
account
-id : string
-billingaddress : address
-isclosed : boolean
-open : date
-closed : date
shoppingcart
-created : date
order
-ordernum : string
-ordered : date
-shippeddate : date
-shipto : address
-status : orderstatus
-total : real
lineitem
-quantity : integer
-price : real
product
-id : string
-name : string
-suuplier : supplier
Powered By Visual Paradigm Community Edition

C) Develop CRUD matrix to represent relationships between use cases and problem domain classes

Data Entity	CRUD	Resulting Usecase
Customer	Create	Add new Customer
	Read/Report	Find Customer
		Generate Customer List
	Update	Update Customer information
	Delete	Delete inactive customer
Order	Create	Create New Order
	Read/Report	View Order
		View Order history
	Update	Update order
	Delete	Cancel order
Inventory Item	Create	Add new inventory item
	Read/Report	View inventory item
		Find inventory item
		Generate inventory item List
	Update	Update inventory item
	Delete	Delete inventory item
Shipment	Create	Create New Shipmnent Request
	Read/Report	View Shipment Details
		Generate Shipment Report
	Update	update shipment request
	Delete	cancel shipment request

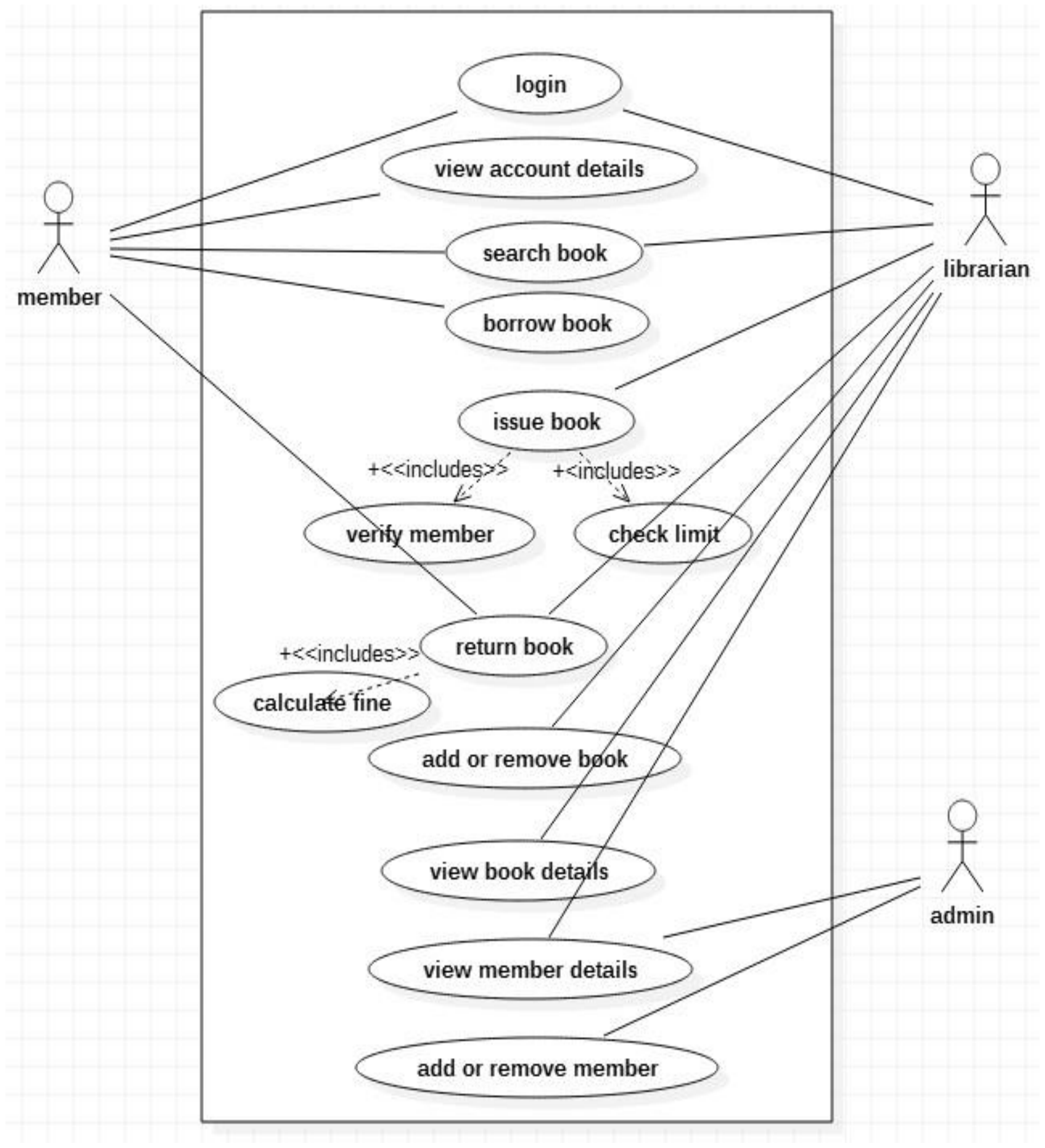
Experiment 4:

a) Develop Use case diagrams
b) Develop elaborate Use case descriptions & scenarios
c) Develop prototypes (without functionality)

a) Develop Use case diagrams

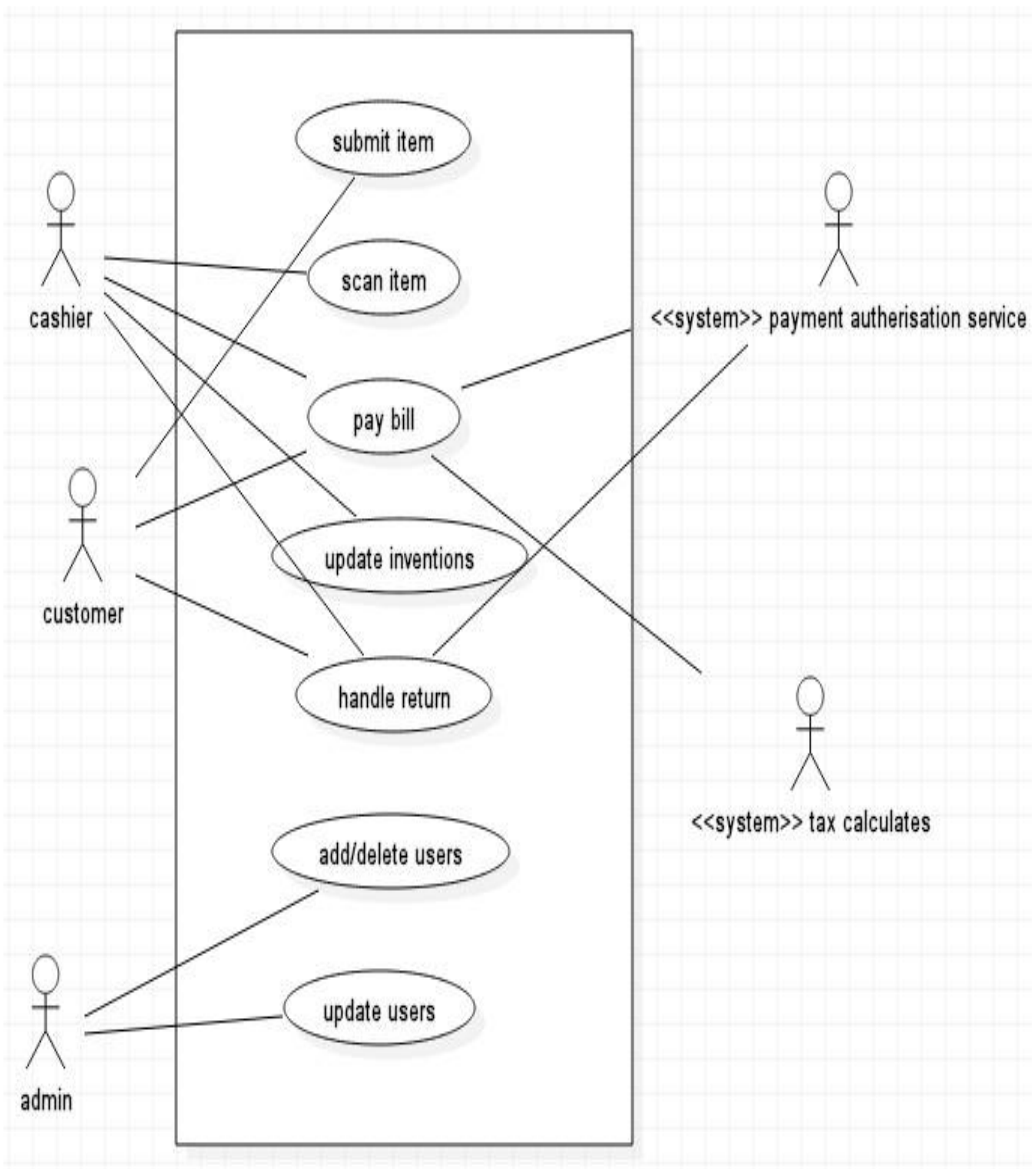
1. Library Management System:

Use case diagram of library management system

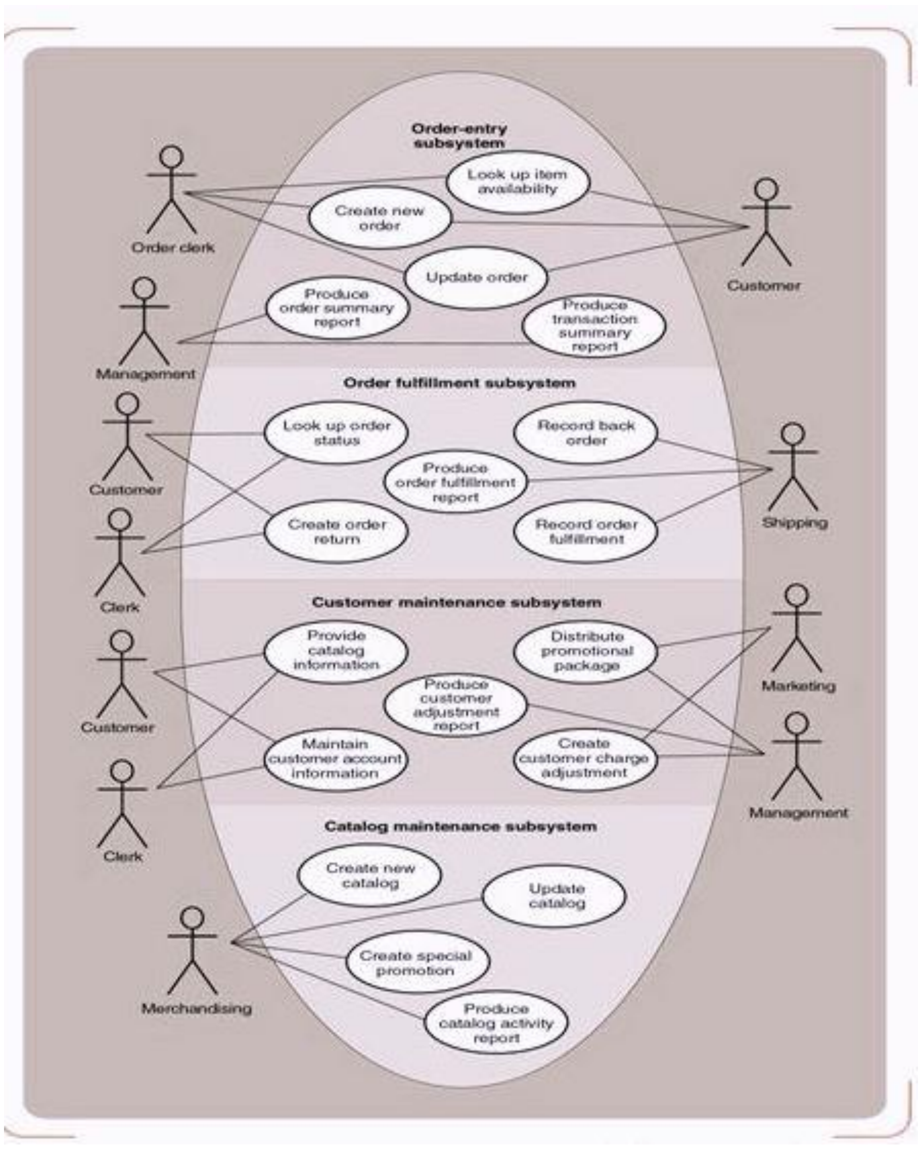


2.Point-Of-Sale Terminal:

Use case diagram of point of sale terminal



3.CUSTOMER SUPPORT SYSTEM USE CASE DIAGRAM


Order-entry subsystem
Order clerk
Look up item availability
Create new order
Update order
Customer
Management
Produce order summary report
Produce transaction summary report
Order fulfillment subsystem
Customer
Look up order status
Record back order
Produce order fulfillment report
Shipping
Create order return
Record order fulfillment
Clerk
Customer maintenance subsystem
Customer
Provide catalog information
Distribute promotional package
Marketing
Produce customer adjustment report
Maintain customer account information
Create customer charge adjustment
Management
Clerk
Catalog maintenance subsystem
Create new catalog
Update catalog
Merchandising
Create special promotion
Produce catalog activity report

b) Develop elaborate Use case descriptions & scenarios

Scenario is a specific sequence of actions and interactions between actors and the system under discussion; it is also called a use case instance. **Use case** is a collection of related success and failure scenarios that describe actors using a system to support a goal.

<u>1. Library Management System:</u>

Make Reservation Use case along with its scenarios

Primary Actor:	Borrower
Stakeholders & Interests: 1. Borrower: fast & easy environment for making reservation 2. Librarian: easy management of borrower, reservation & load details 3. Master Librarian: easy management of librarian, borrowers	
Pre-conditions:	Borrower login, Enable for making reservation
Post-conditions:	Reservation details for book are saved, Taking necessary actions when book is not available
Main Success Scenarios: 1. Borrower login to the system 2. Borrower search for the required title 3. Selects one of the books form the list of searched results 4. Selects for make reservation option for the book 5. If book is available book will be issued to the respective borrower 6. Borrower loan details will be updated to indicate issue of book	
Extensions: 1. At any time, System fails: • To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario. 2. Invalid borrower credentials: • System signals error and rejects entry. 3. If selected book is not available: • Borrower need to check until book is returned by other borrower. • Borrower will be notified if new book is add to library.	

2. Point-Of-Sale Terminal:

Process Sale Use case along with its scenarios

Primary Actor:	Cashier
Stakeholders & Interests: 1. Cashier: wants accurate, fast entry, and no payment errors 2. Sales Person: wants sales commissions updated 3. Customer: wants purchase and fast service with minimal effort 4. Company: wants accurately record transactions and satisfy customer interests 5. Manager: wants to be able to quickly perform override operations 6. Government Tax agencies: want to collect tax from every sale 7. Payment Authorization Service: want to receive digital authorization request in correct format and protocol	
Pre-conditions:	Cashier Identified and Authenticated
Post-conditions:	Sale is saved, Tax is correctly calculated, Accounting and Inventory are update, Commissions recorded, Receipt is generated, Payment authentication approvals are recorded
Main Success Scenarios: 1. Customer arrives at POS checkout with items 2. Cashier starts new sale 3. Cashier enters item identifiers 4. System records sales line item & presents item details Cashier repeats steps 3-4 until indicates done. 5. System presents total with taxes and asks for payment 6. Customer pays and system handles payment 7. System logs completed sales and sends payment information to accounting and inventory systems 8. System presents receipt 9. Customer leaves with receipt and items	

Extensions:
1. At any time, System fails: • To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario. 2. Invalid identifier: • System signals error and rejects entry. 3. System detects failure to communicate with external tax calculation system service: • System restarts the service on the POS node, and continues. • Cashier may manually calculate and enter the tax, or cancel the sale. 4. Customer says they intended to pay by cash but don't have enough cash: • Customer uses an alternate payment method. • Customer tells Cashier to cancel sale. Cashier cancels sale on System.

3. <u>Customer Supporting System:</u>

Add items to cart Use case along with its scenarios

Primary Actor:	Customer
Stakeholders & Interests: 1. Customer: fast & easy environment for placing orders 2. Company: accurate entry of product, payment, shipment details 3. System Administrator: easy maintenance of product details	
Pre-conditions:	Customer login
Post conditions:	Selected items/products are successfully added to cart
Main Success Scenarios: 1. Customer login to the system 2. Borrower search for the required item/product 3. Selects one of the item/product form the list of searched results 4. Add selected item/product to the cart Customer repeats 2,3,4 steps until required items/products added to cart 5. System displays list of items/products added to the cart	

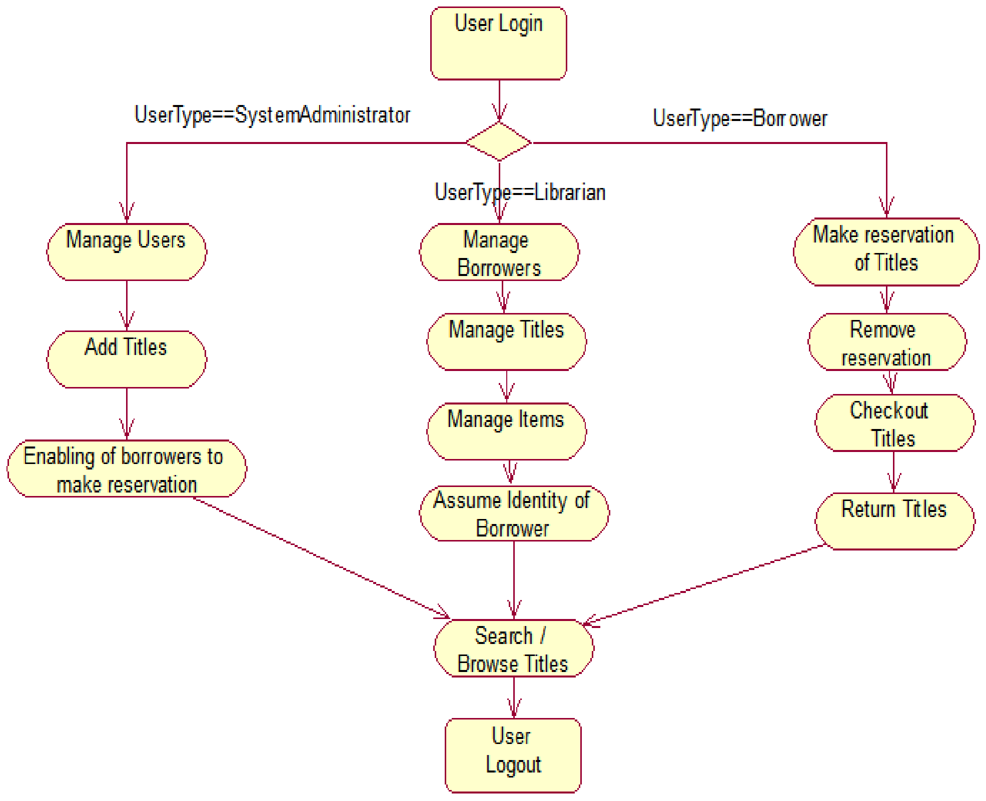
Extensions:

1. At any time, System fails:
   - To support recovery and correct accounting, ensure all transaction sensitive state and events can be recovered from any step of the scenario.
2. Invalid customer credentials:
   - System signals error and rejects entry.
3. If customer is unable to get cart details:
   - Customer need to re-login.
   - Customer need to refresh for getting updated cart details.

c) Develop prototypes (without functionality)

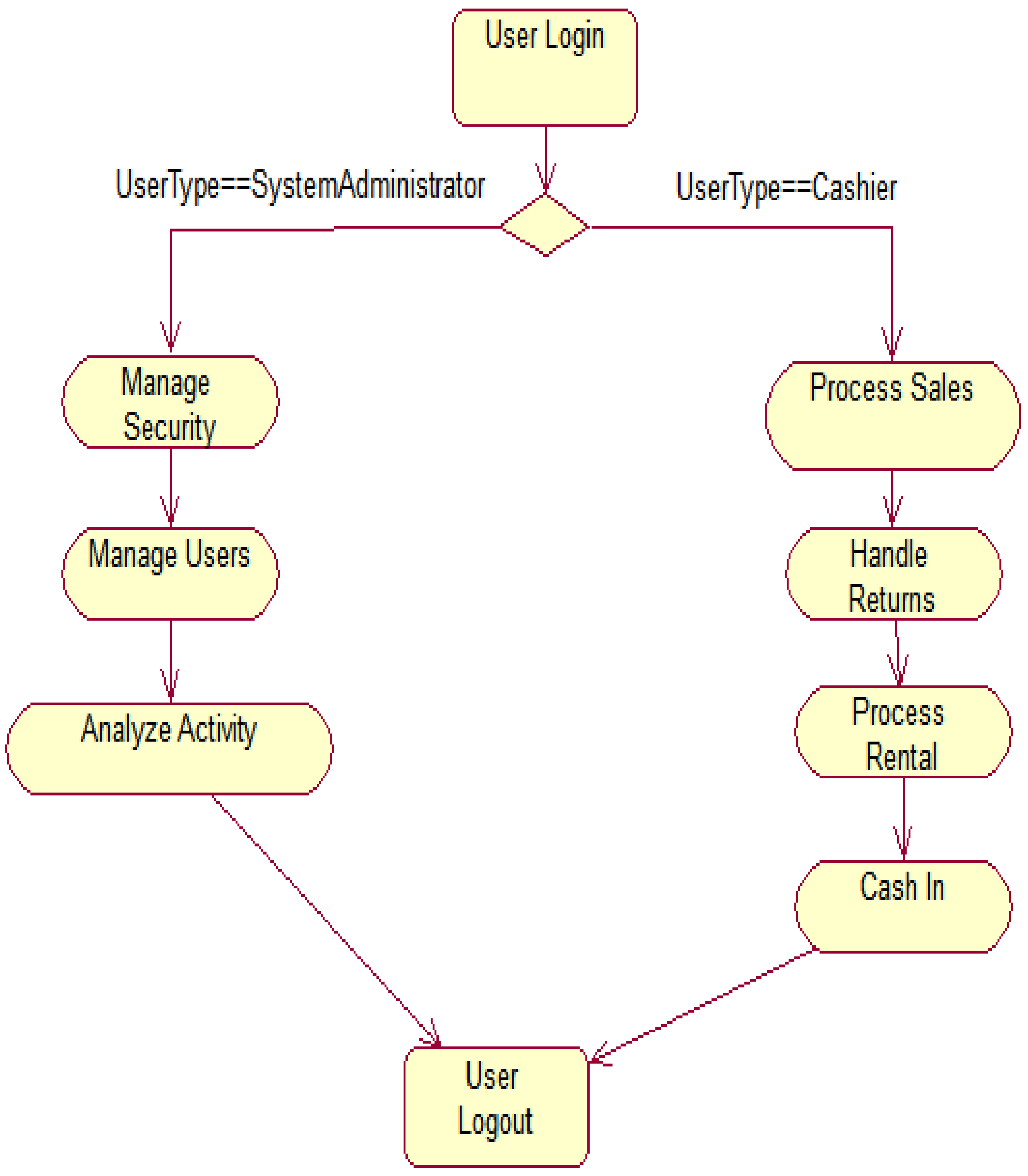
1. Library Management System:

Prototype of LMS



2. Point-Of-Sale Terminal:

Prototype of POS



Experiment 5:

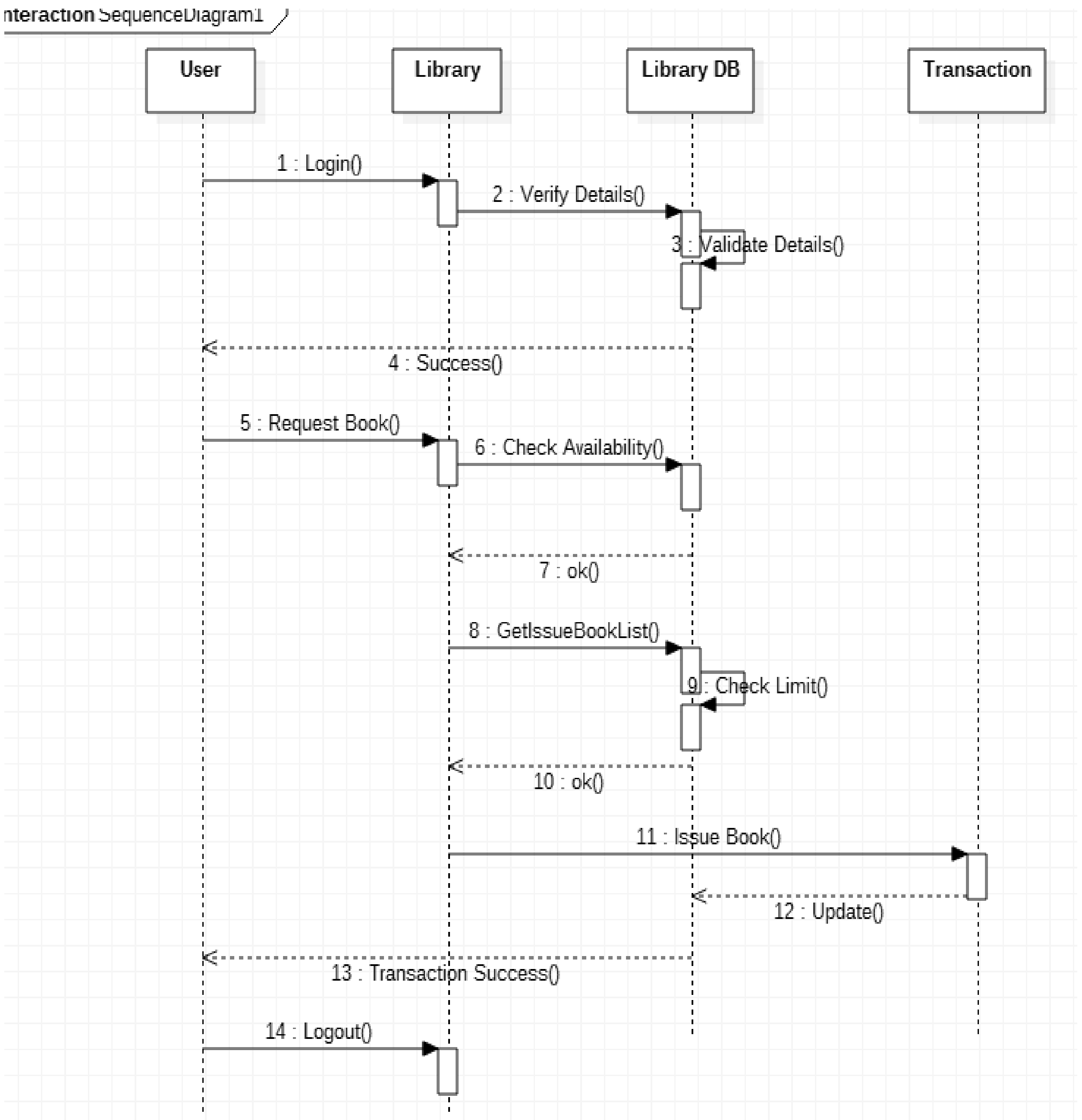
a) Develop system sequence diagrams and high-level sequence diagrams for each use case
b) Identify MVC classes / objects for each use case
c) Develop Detailed Sequence Diagrams / Communication diagrams for each use case showing interactions among all the three-layer objects

a) Develop system sequence diagrams and high-level sequence diagrams for each use case

A system sequence diagram (SSD) is a picture that shows, for a particular scenario of a use case, the events that external actors generate their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems.

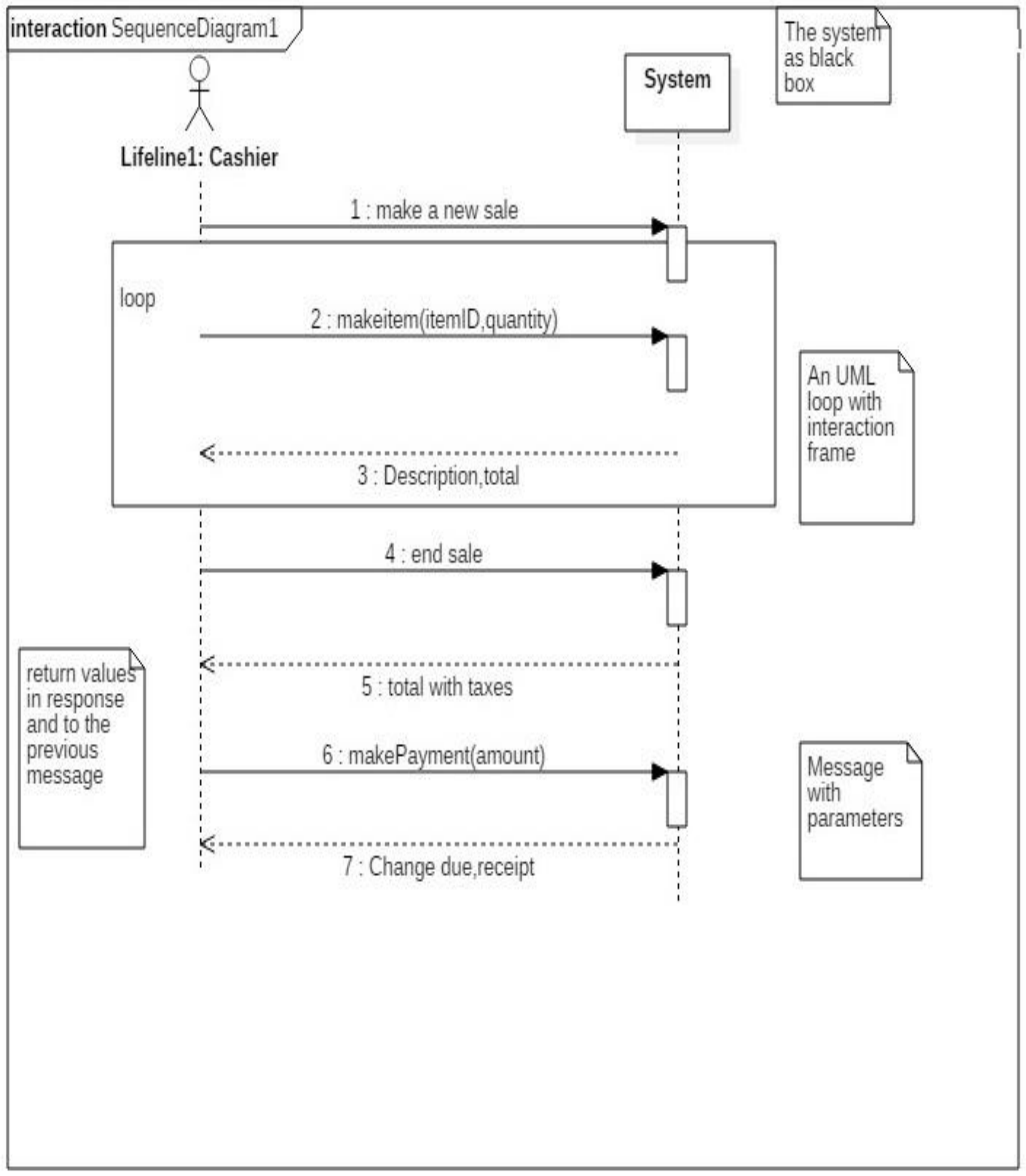
1. Library Management System:

System sequence diagram for library management system

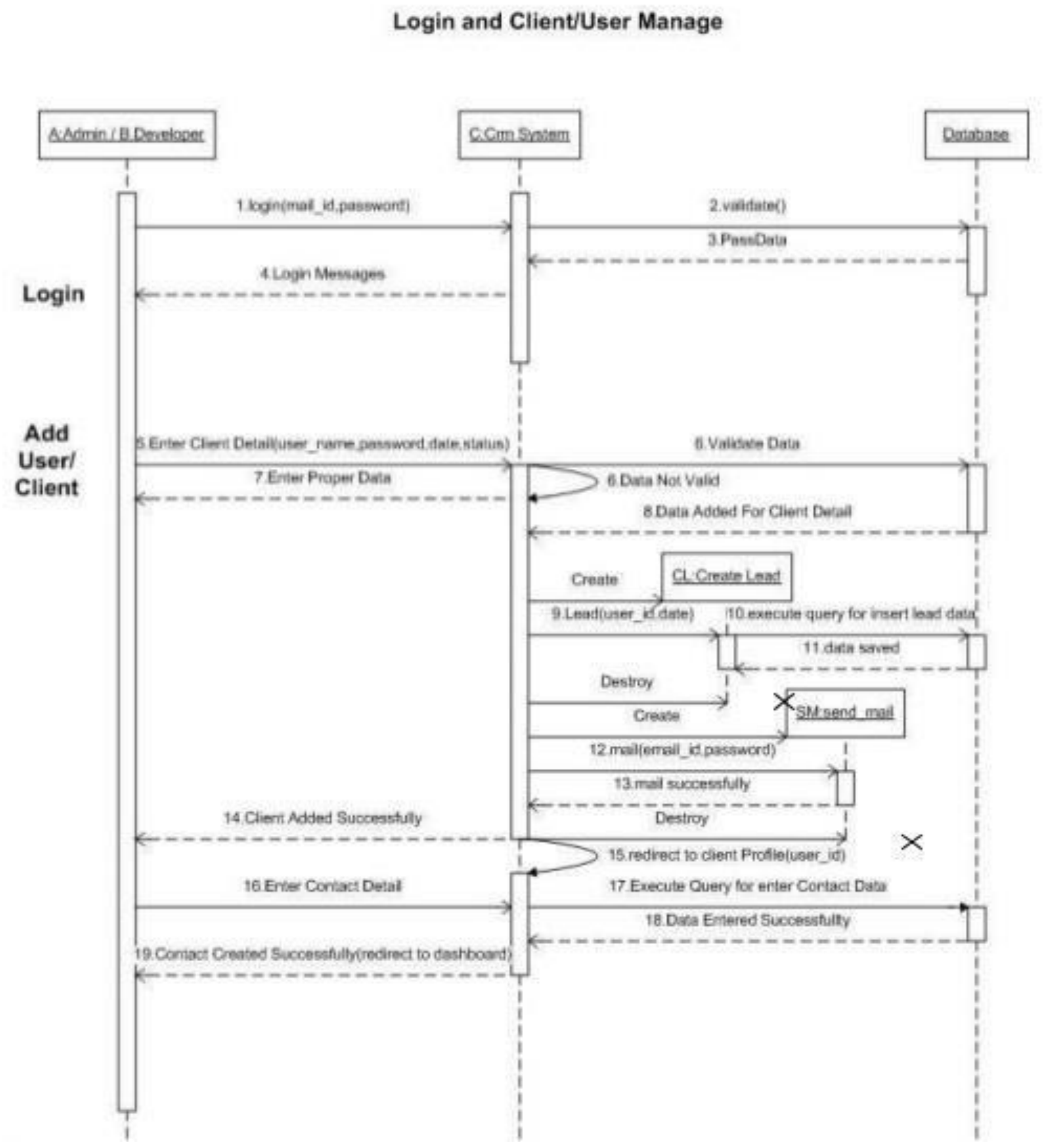


2.Point-Of-Sale Terminal:

System sequence diagram for POS terminal system



3.SEQUENCE DIAGRAM FOR CUSTOMER SUPPORT SYSTEM:


Login and Client/User Manage
A:Admin / B.Developer
C:Crm System
Database
Login
Add User/ Client
1.login(mail_id,password)
2.validate()
3.PassData
4.Login Messages
5.Enter Client Detail(user_name,password,date,status)
6.Validate Data
7.Enter Proper Data
6.Data Not Valid
8.Data Added For Client Detail
Create
CL:Create Lead
9.Lead(user_id,date)
10.execute query for insert lead data
11.data saved
Destroy
Create
SM:send_mail
12.mail(email_id,password)
13.mail successfully
14.Client Added Successfully
Destroy
15.redirect to client Profile(user_id)
16.Enter Contact Detail
17.Execute Query for enter Contact Data
18.Data Entered Successfully
19.Contact Created Successfully(redirect to dashboard)

b) Identify MVC classes / objects for each use case

a) 1. Library Management System:

MVC classes/objects of LMS

Use case	Model Classes	View Classes	Controller Classes
Make Reservation	Loan	BorrowerFrame	Reservation
Remove Reservation			
Checkout Item			Titles Items
Return Titles/Items			
Manage Titles/Items	TitlesModel	LibrarianFrame	
Manage Librarians	UsersModel		Librarian
Manage Borrowers			Borrower
Assume Identity of Borrower			

2. Point-Of-Sale Terminal:

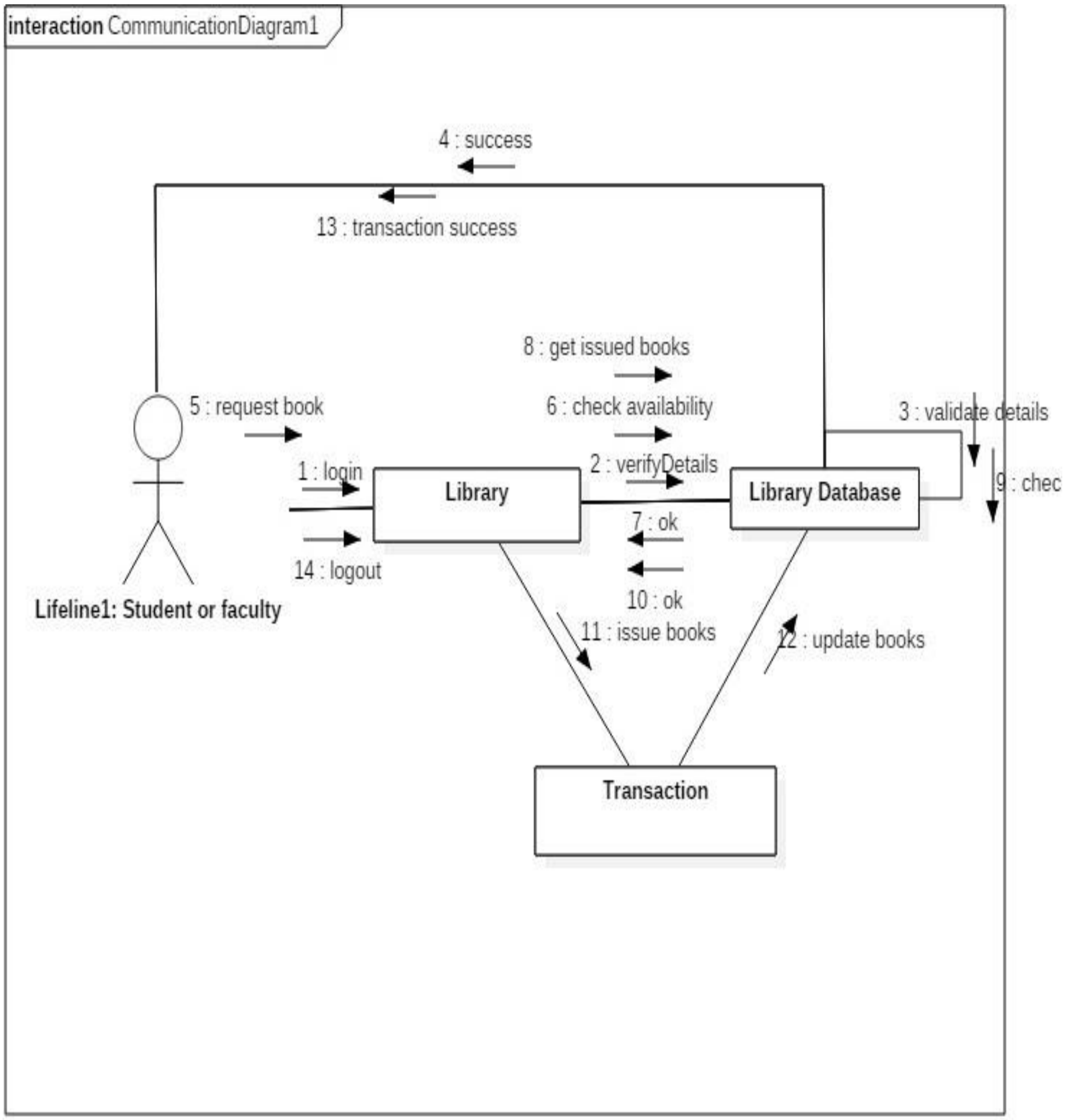
MVC classes/objects of POS

Use case	Model Classes	View Classes	Controller Classes
Process Sale	SalesModel TaxCalculator HRSystem Store	ProcessSaleFrame ProcessSaleConsole	Register Sale Payment SalsLineItem
Handle Returns			
Process Rental			
Cash In			
Analyze Activity	ActivityModel	ActivityFrame	Analyze
Manage Security	UserModel	LoginFrame	Users
Manage Users			
Login			

c)Develop Detailed Sequence Diagrams / Communication diagrams for each use case showing interactions among all the three-layer objects

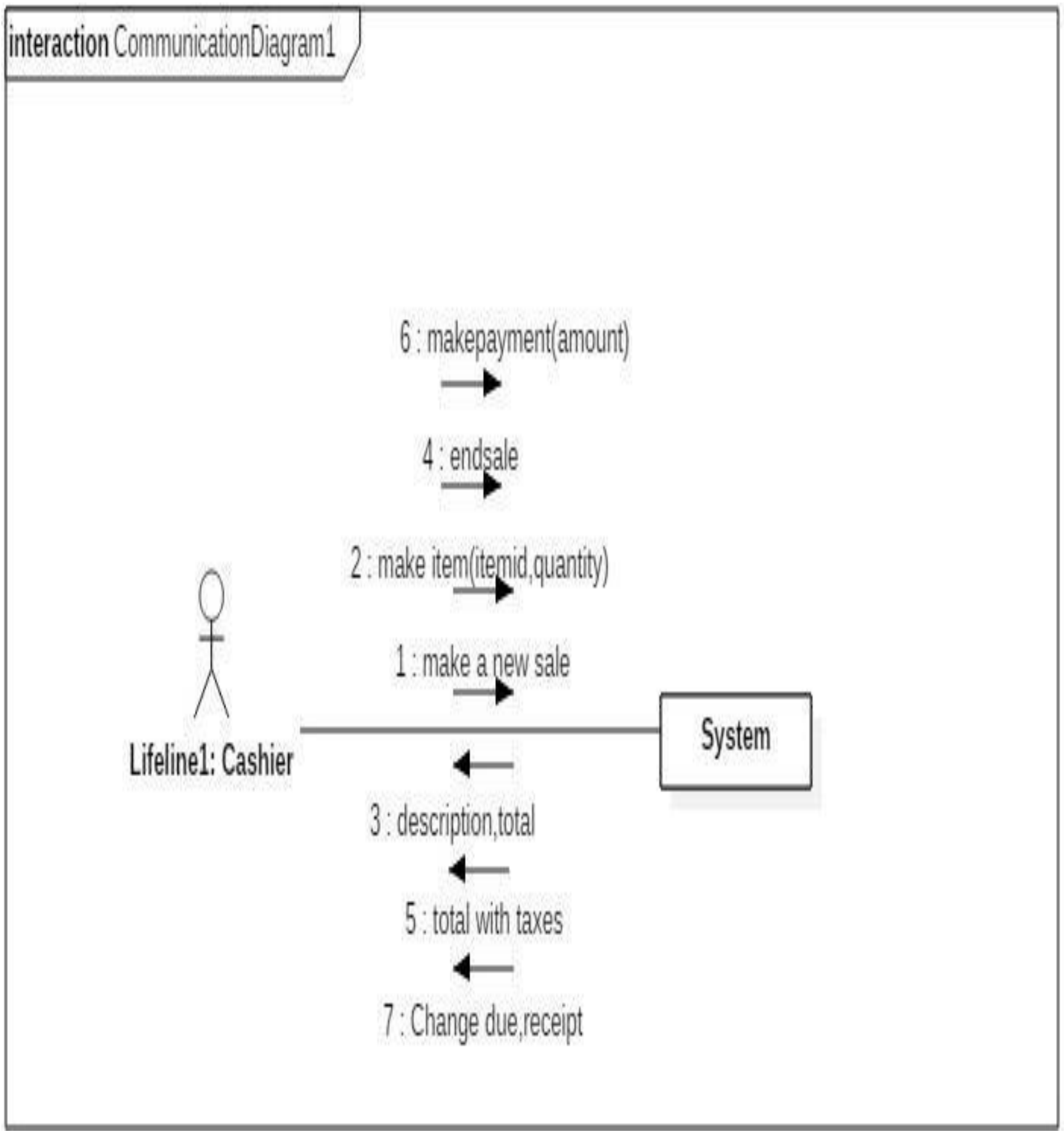
1. Library Management System:

Communication diagrams for LMS

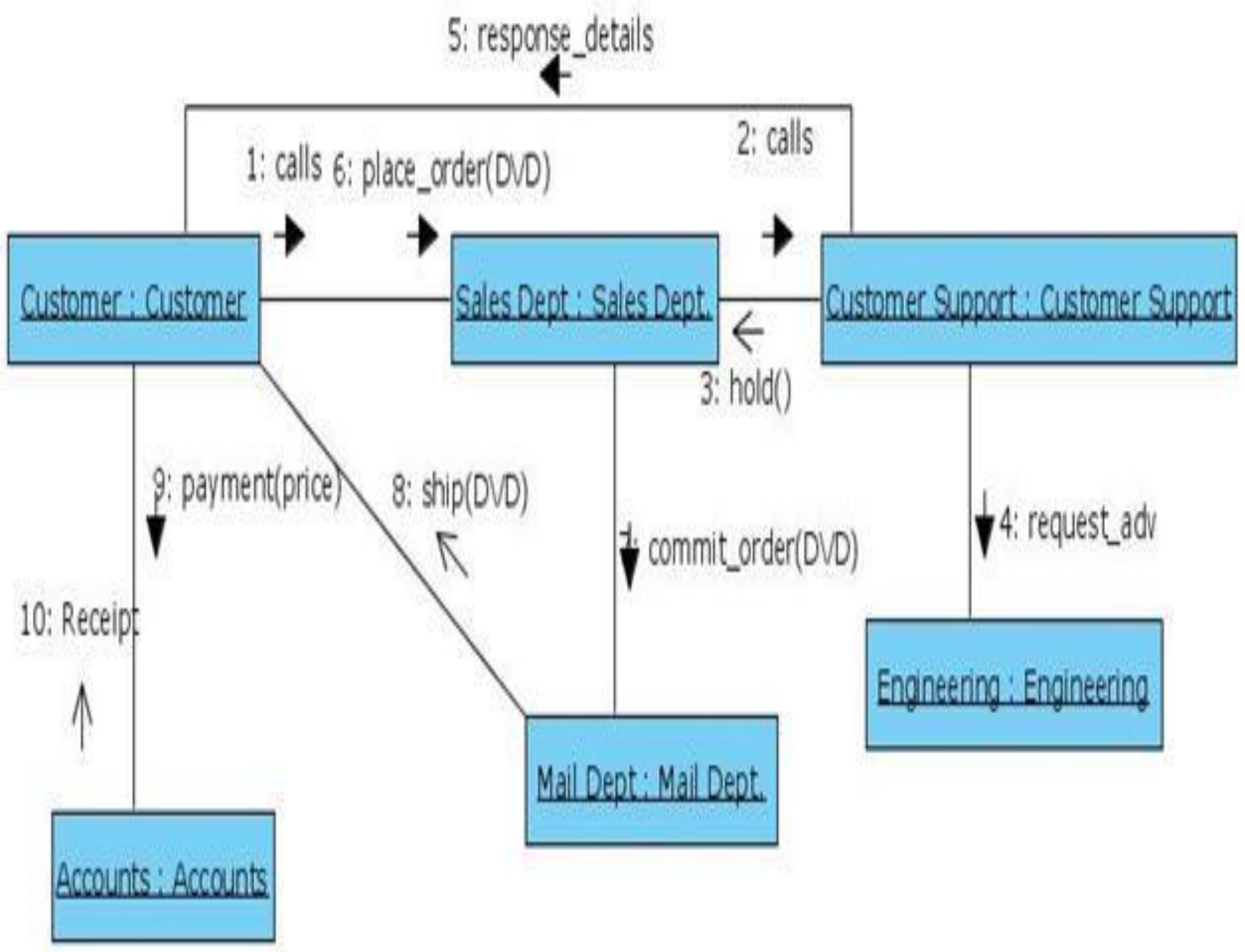


2.Point-Of-Sale Terminal:

Communication diagrams for POS



3. Communication diagram for customer support system:



Experiment 6:

a) Develop detailed design class model (use GRASP patterns for responsibility assignment)

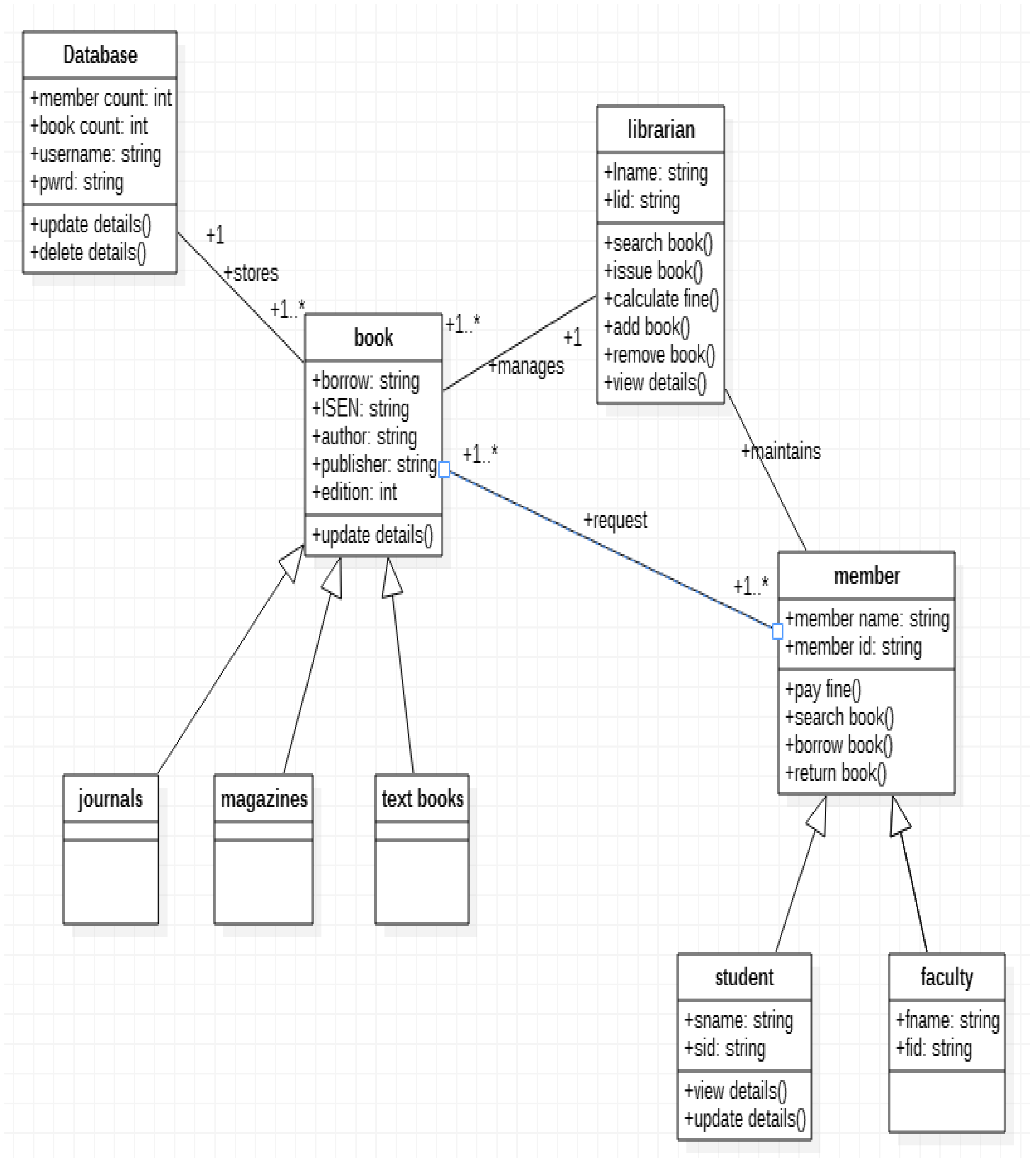
b) Develop three-layer package diagrams for each case study

a) Develop detailed design class model (use GRASP patterns for responsibility assignment)

Class diagrams: Rose uses class diagrams to graphically describe generic descriptions of the system you're going to build. Class diagrams contain icons that represent classes and interfaces and their relationships to one another.

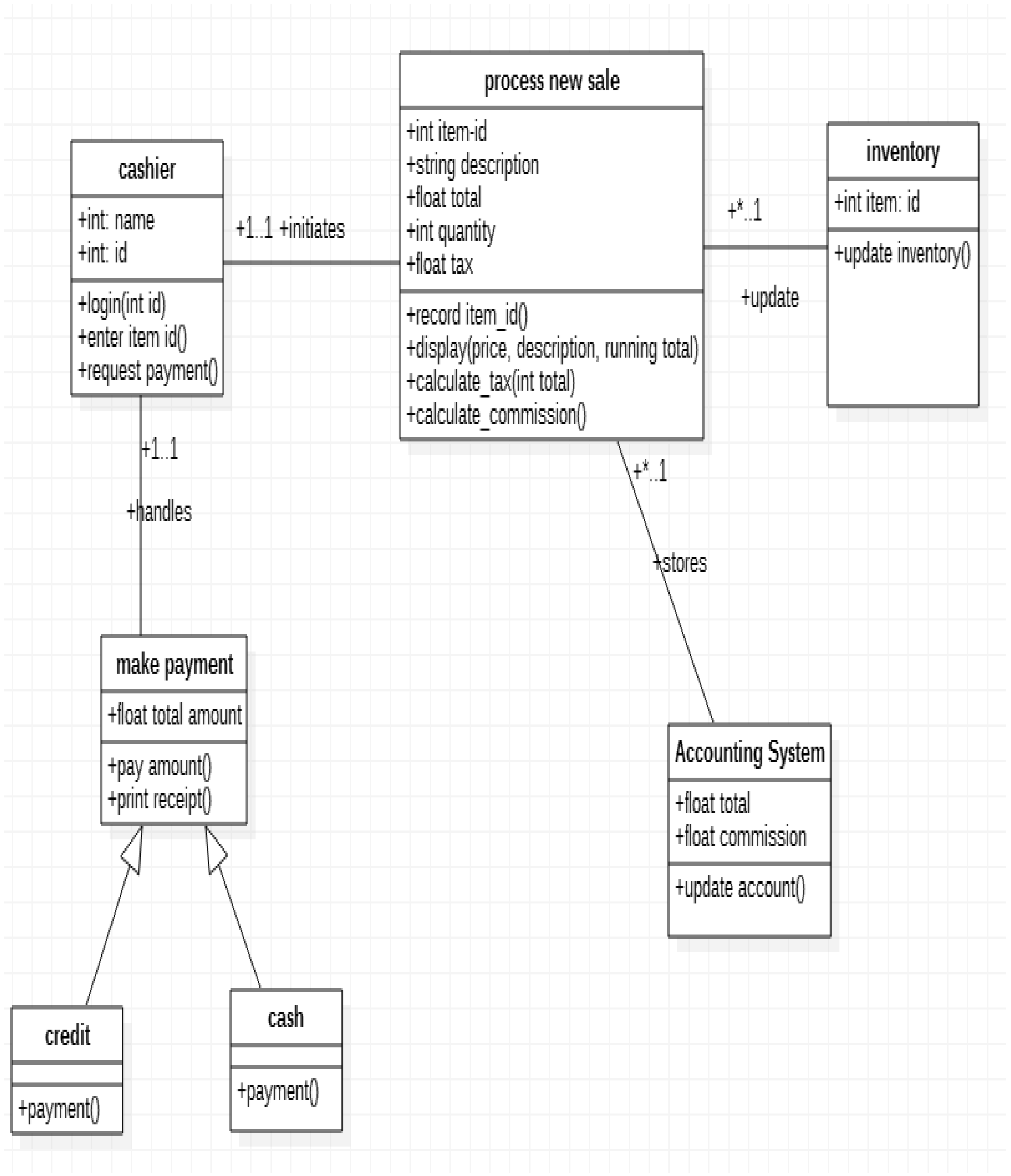
1. **Library Management System**:

Design class diagram for LMS

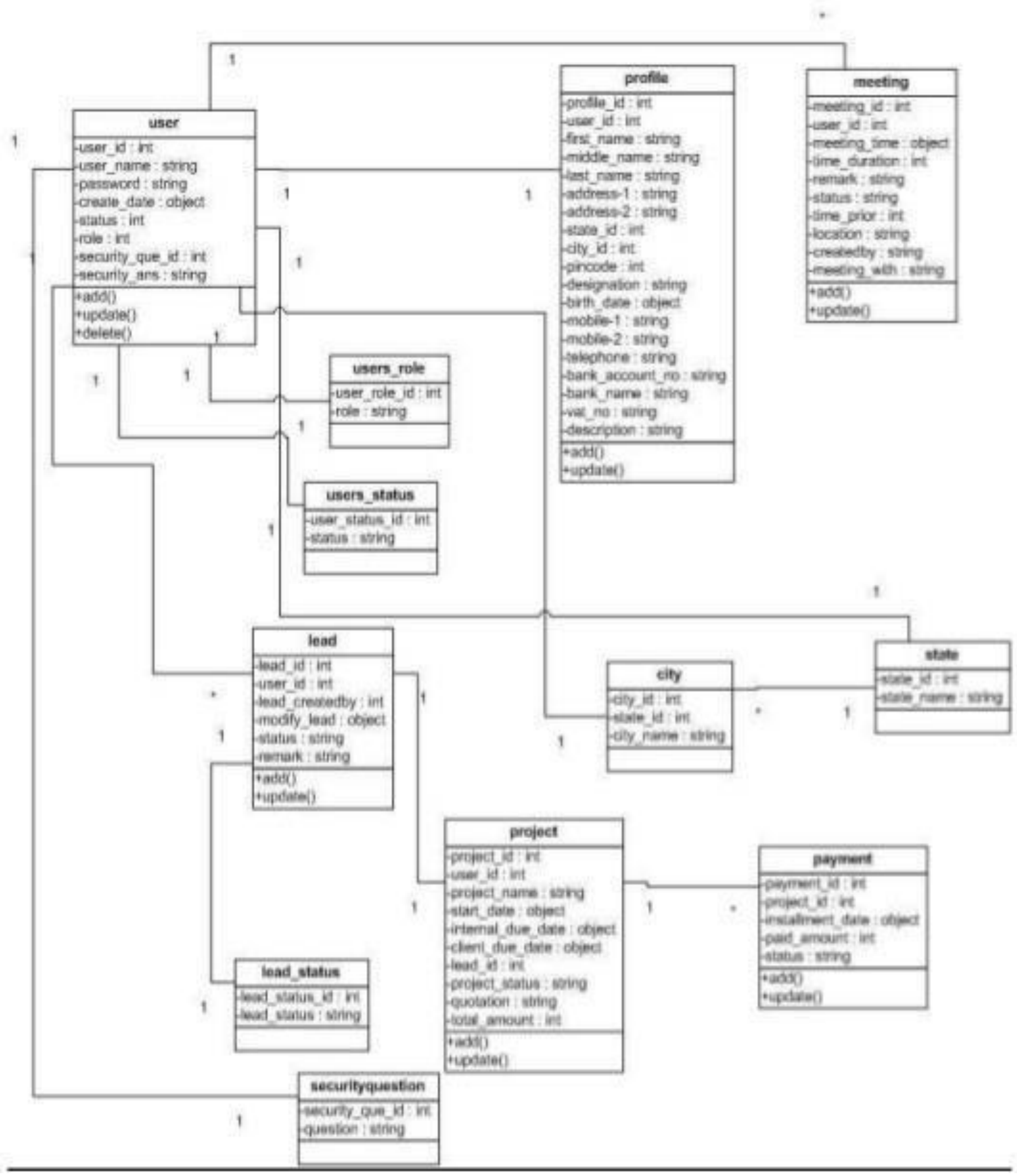

Database
+member count: int
+book count: int
+username: string
+pwrd: string
+update details()
+delete details()
+1
+stores
+1..*
book
+borrow: string
+ISEN: string
+author: string
+publisher: string
+edition: int
+update details()
+1..*
+manages
+1
librarian
+lname: string
+lid: string
+search book()
+issue book()
+calculate fine()
+add book()
+remove book()
+view details()
+maintains
+1..*
+request
+1..*
member
+member name: string
+member id: string
+pay fine()
+search book()
+borrow book()
+return book()
journals
magazines
text books
student
+sname: string
+sid: string
+view details()
+update details()
faculty
+fname: string
+fid: string

2. Point-Of-Sale Terminal:

Design class diagram for POS



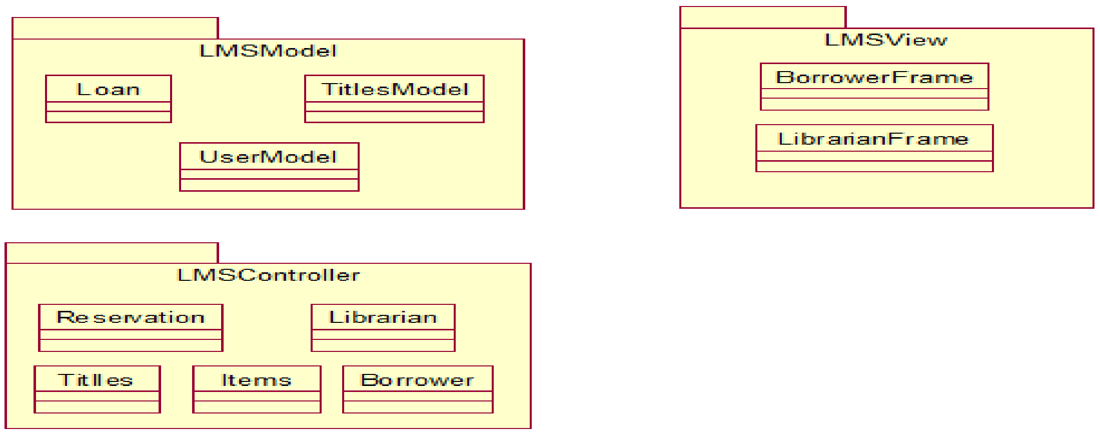
3. Class Diagram for Customer Support System:



b) Develop three-layer package diagrams for each case study

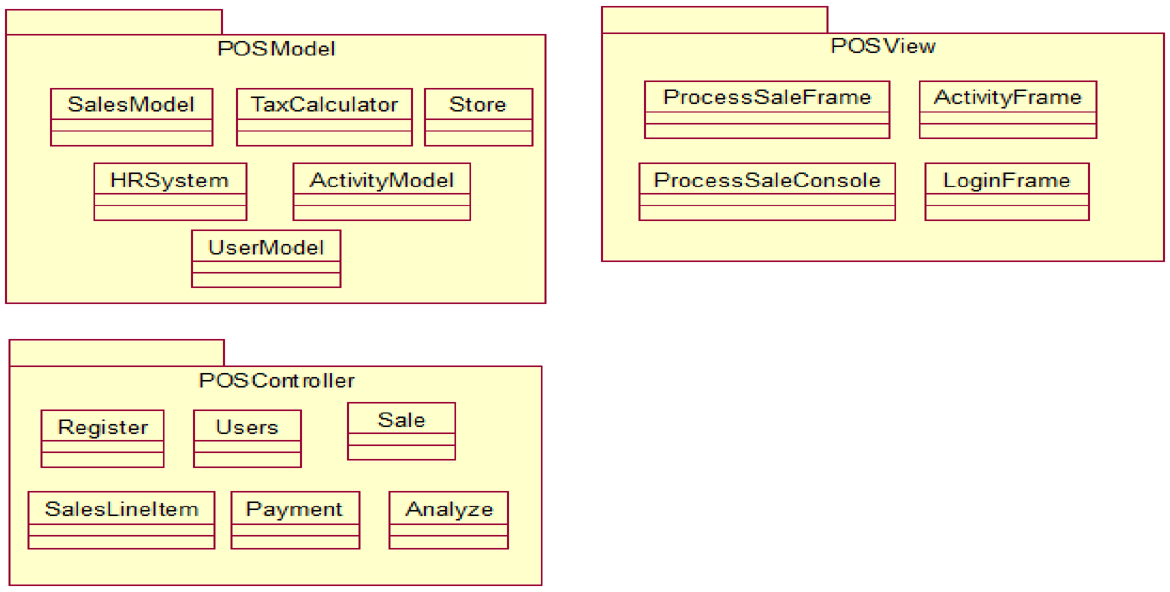
1. Library Management System:

Package diagram for LMS



2. Point-Of-Sale Terminal:

Package diagram for POS


POSModel
SalesModel
TaxCalculator
Store
HRSystem
ActivityModel
UserModel
POSView
ProcessSaleFrame
ActivityFrame
ProcessSaleConsole
LoginFrame
POSController
Register
Users
Sale
SalesLineItem
Payment
Analyze