CONTENTS

		✓Develop detailed design class model (use GRASP patterns for responsibility assignment) ✓Develop three-layer package diagrams for each case study.		
5	5	**Week 11 & 12** ✓ **For each case study:** 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		
6	6	**Week 13 onwards:** ✓ For each case study: a) Develop sample diagrams for other UML diagrams - state chart diagrams, activity diagrams and deployment diagrams		

Course Objective:

- Construct UML diagrams for static view and dynamic view of the system.
- Generate creational patterns by applicable patterns for given context.
- Create refined model for given Scenario using structural patterns.
- Construct behavioral patterns for given applications.

Course Outcomes:

- Understand the Case studies and design the Model..
- Understand how design patterns solve design problems.
- Develop design solutions using creational patterns.
- Construct design solutions by using structural and behavioural patterns

UML & Design Patterns Lab

Take three case studies:

Customer Support System (in the Object-Oriented Analysis & Design with the Unified Process by Satzinger, Jackson & Burd Cengage Learning)

Point-Of-Sale Terminal (in Larman textbook)

Library Management System (in the reference book no. 2 i.e. UML toolkit)

Week 1:

Familiarization with Rational Rose or Umbrello

For each case study:

Week 2, 3 & 4:

For each case study:

Identify and analyze events

Identify Use cases

Develop event table

Identify & analyze domain classes

Represent use cases and a domain class diagram using Rational Rose

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

Week 5 & 6:

For each case study:

Develop Use case diagrams

Develop elaborate Use case descriptions & scenarios

Develop prototypes (without functionality)

Develop system sequence diagrams

Week 7, 8, 9 & 10:

For each case study:

Develop high-level sequence diagrams for each use case

Identify MVC classes / objects for each use case

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

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

Develop three-layer package diagrams for each case study

Week 11 & 12:

For each case study:

Develop Use case Packages

Develop component diagrams

Identify relationships between use cases and represent them

Refine domain class model by showing all the associations among classes

Week 13 onwards:

For Each Case Study:

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

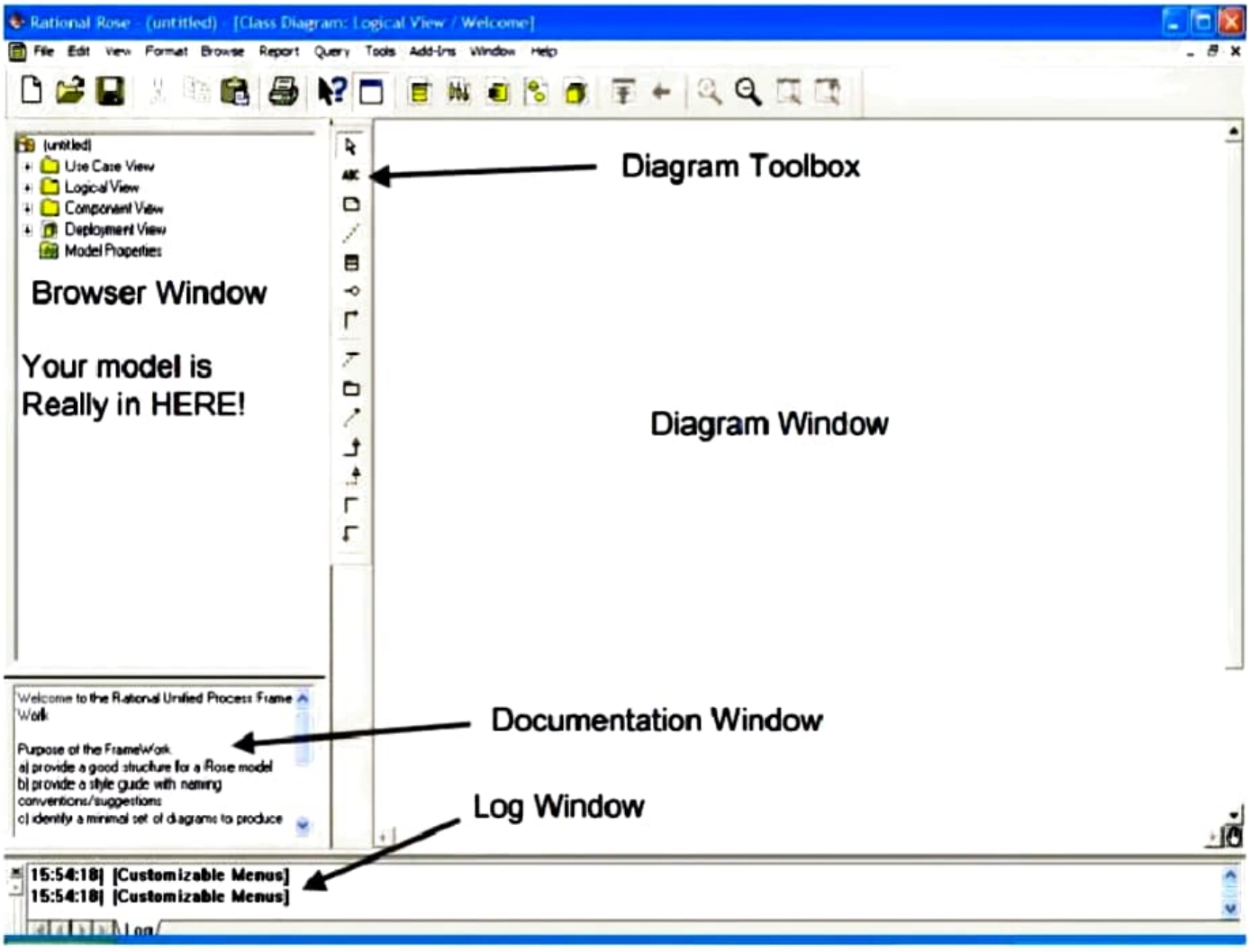
Week-1:

Familiarization with Rational (Rose) Approach/ Umbrello

- Rational Rose is an object-oriented Unified Modeling Language (UML) software design tool intended for visual modeling and component construction of enterprise-level software applications.
- ROSE = Rational Object Oriented Software Engineering Rational Rose is a set of visual modeling tools for development of object oriented software.
- Visual Modeling is the process of graphically depicting then system to be developed • Presenting essential details • Filtering out non-essential details • Viewing the system from different perspectives
- When Should ROSE be Used? Modeling can be useful at any point in the application development process. Initial Design Work (Requirement Analysis and Definition) • Use Cases • Class Diagrams • Sequence Diagram.
- Rational Rose is an object-oriented programming (OOP) and unified modeling language (UML) tool to design enterprise-level software applications and components. It creates visual software application models under object-oriented principles. Example application models include the creation of actors, use cases, relationships, objects, entities, etc. Rational Rose uses classical UML concepts to graphically model software applications. This facilitates documenting the environment, requirements and overall design.
- OOP methodologies use UML to graphically depict software application behavior and architecture. These models are the building blocks and development blueprint of the application's entire construction process. The popular Rational Rose modeling tool allows developers to create entire architecture or component-level system models, while depicting relationship and control flow.
- Rational Rose is a powerful graphical user interface (GUI) modeling tool using efficient and user-friendly drag and drop and

design maneuverability. Certain Rational Rose versions actually produce relevant source code for designed models.

- Rose Rose Enterprise:• Supports multiple languages, including VC++, VB, Java, CORBA
- Rose GUI: • Standard ToolBar • Diagram ToolBox • Browser • Diagram Window • Documentation Window • Specifications • Log Window



Rational rose tutorial:

- ROSE stands for Rational Object-oriented Software Engineering.
- Rational Rose is developed by Rational Corporation which is under IBM.
- Rational Rose is a tool for modeling software systems.
- Rational Rose supports UML.

- Rational Rose is a tool that supports round-trip engineering means a tool that supports conversion of a model to code and from code to a model.

Rational Rose Interface:

Menubar: The menubar consists of several menus like the file menu, edit menu, view menu etc. All these menus contain several options.

Toolbar: The toolbar contains the most frequently used actions like New, Open, Save etc…

Statusbar: The statusbar at the bottom displays status messages.

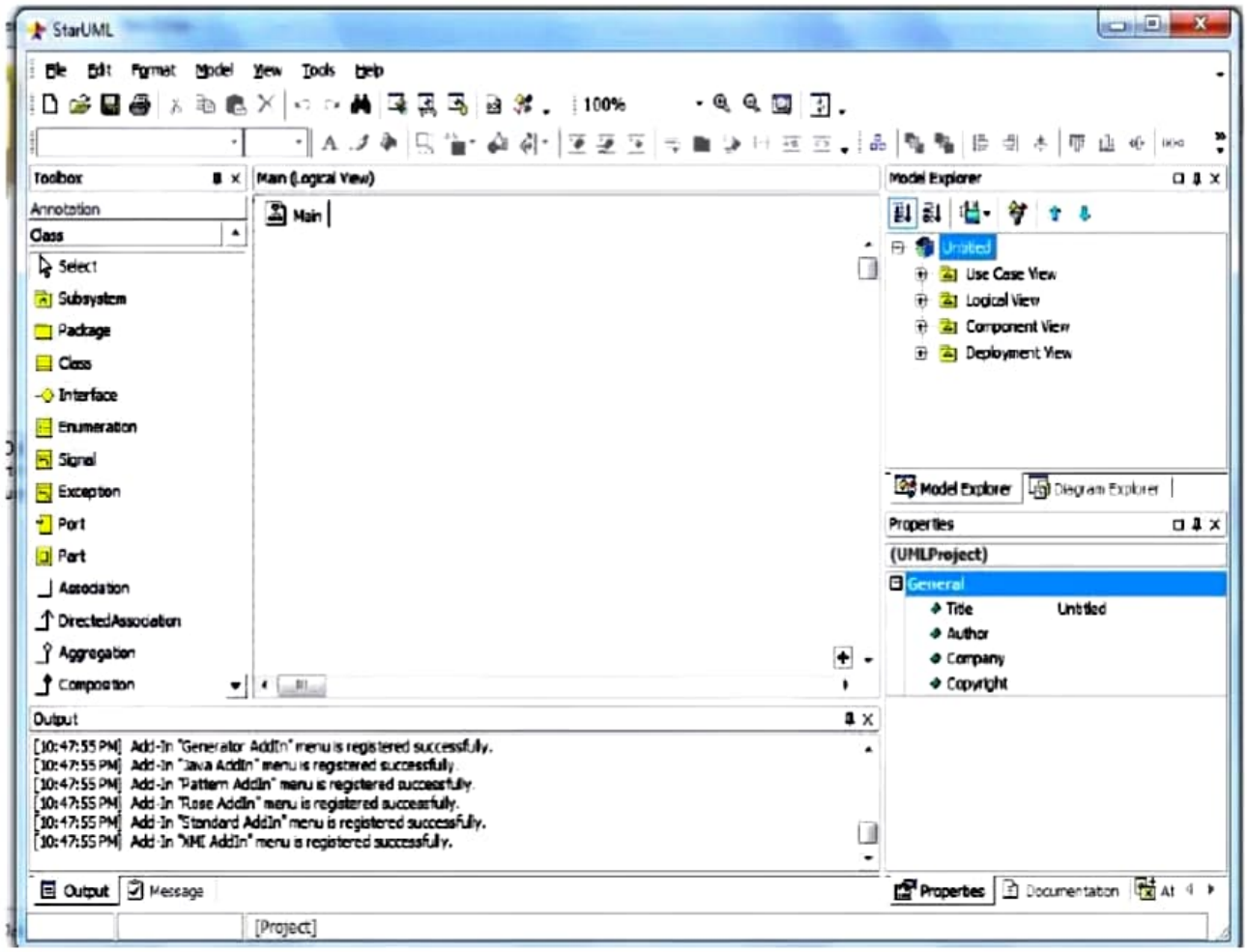
Browser Window: The browser window displays the views: Use Case View, Logical View, Component View and Deployment View. Each of these views contains the diagrams.

Diagram Toolbar: The diagram toolbar displays the symbols of the respective type of diagram.

Diagram Window: The diagram window is the place where the user draws the diagrams using the symbols from the diagram toolbar.

Log Window: This window is used to display error messages, warnings and information messages.

Documentation Window: This window is used to display the documentation related to the symbols and other aspects.



- Two popular features of Rational Rose are its ability to provide iterative development and round-trip engineering. Rational Rose allows designers to take advantage of iterative development (sometimes called evolutionary development) because the new application can be created in stages with the output of one iteration becoming the input to the next. (This is in contrast to waterfall development where the whole project is completed from start to finish before a user gets to try it out.)

- Then, as the developer begins to understand how the components interact and makes modifications in the design, Rational Rose can perform what is called "round-trip engineering" by going back and updating the rest of the model to ensure the code remains consistent.

- Rational Rose is extensible, with downloadable add-ins and third-party partner applications. It supports COM/DCOM (ActiveX), JavaBeans, and Corba component standards.

1. Browser Window This presents a hierarchical view of the analysis and design model, including all the diagrams and all the individual elements that make up a diagram.

2. Drawing Tools This tool presents a set of icons that indicate the different elements that can be added to a diagram. The elements that can be used will change, depending on the type of diagram being created. Different diagram types have different sets of icons. If you were creating a different diagram type, you would see a different set of icons. The above example is a class diagram in logical view.

3. Diagram Window This is where the diagram is actually created. You will see that the diagram shown in the drawing window on Figure 1 represents a high-level model of this course. Course content can be seen as a system composed of four interacting subsystems, two of which involve software. We have used the Package element to represent the subsystems, and the Note element to indicate which packages contain software.

4. Documentation Window It is strongly recommended that each element added to a diagram have documentation to accompany it. To add documentation, right click on the element, select specification, and fill in the documentation field. The documentation will then be shown in the documentation window each time the mouse is clicked on the element. Documentation can also be added directly to the documentation window.

VIVA QUESTIONS

1) Description of UMBRELLO software ?
2) What is UML?
3) What is an object?
4) What is an abstract class?
5) What type of core-relationship is represented by the symbol in the figure below?Aggregation
6) Which core element of UML is being shown in the figure?.
7) Which UML diagram is shown below?
8) Which UML diagram is shown below?
9) Which UML diagram is shown below?
10) Which UML diagram is shown below?
11) What is Design pattern?
12) How do we represent private, public and protected in class diagrams?

Week-2,3&4

a) Identify the users and Analyze Events

b) Identify Use cases

c) Develop event table

d) Identify & analyze domain classes

e) Represent Use cases & Domain class diagram using Umbrello

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

Case study 1: Customer Support System

a) Identify the users and Analyze Events

USERS	**Events**
CUSTOMER	□ Customer requests Catalog □ Customer wants to check item availability □ Customer places an order □ Customer changes or cancels order □ Customer wants to check order status □ Customer returns item □ Customer updates account information
MANAGER	□ Produces order summary reports □ Produces transaction summary reports □ Produces fulfillment summary reports □ Produces prospective customer activity reports □ Produces customer adjustment concession reports □ Adjusts customer charges □ Sends promotional materials to customers
SHIPMENT	□Shipping fulfills order □ Shipping identifies back order

b) Identifying Use cases

A *use case diagram* shows a set of use cases and actors (a special kind of class) and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

USER / ACTOR	USER GOAL
Order Clerk	Look up Item availability. Create new order. Update Order
Shipping Clerk	Record order fulfillment Record back order
Merchandising Manager	Create special Promotion Produce catalog activity report

c) Developing Event Table

Event: An event in the Unified Modeling Language (UML) is a notable occurrence at a particular point in time.

S.No	Event	Trigger	Source	Use Case	Response	Destination
1	Customer wants to check item availability	Item Inquiry	Customer	Look up item availability	Item availability details	Customer
2	Customer places an order	New order	Customer	Create new order	Real-time link order confirmation order details transaction	Credit bureau customer shipping bank
3	Customer changes or cancels order	Order Change Request	Customer	Update order	Change confirmation Order:change detail Transaction	Customer Shipping Bank
4	Time to produce order summary reports	"End of work, quarter, and year"		Produce order summary reports	Order summary reports	Management
5	Time to produce	"End of Day"		Produce transaction	Transaction summary	Accounting

	transaction summary reports			summary reports	reports	
6	Customer or Management wants to check order status	Order status inquiry	Customer or Management	Look up order status	Order status details	Customer or Management
7	Shipping fulfills order	Order fulfillment notice	Shipping	Record order fulfillment		
8	Shipping identifies back order	Back order notice	Shipping	Record back order	Back order notification	Customer
9	Customer returns item	Order return notice	Customer	Create order return	Return confirmation Transaction	Customer
10	Time to produce fulfillment summary reports	"End of week month, quarter, and year"		Production fulfillment summary reports	fulfillment summary reports	Management
11	Prospective customer requests catalogue	Catalogue request	Prospective customer	Provide catalogue info	Catalogue	Prospective customer
12	Time to produce prospective customer activity reports	"End of month"		Produce prospective customer activity reports	prospective customer activity reports	Marketing

13	Customer updates account information	Customer account update notice	Customer	Update customer account		
14	Marketing wants to send promotional materials to customers	Promotion package details	Marketing	Distribute promotional package	Promotional package	Customer and prospective customer
15	Management adjusts customer charges	Customer charge adjustment	Management	Create customer charge adjustment	Charge adjustment notification Transaction	Bank
16	Time to produce customer adjustment concession reports	"End of month"		Produce customer adjustment reports	Customer adjustment reports	Management
17	Merchandising updates, catalog	Catalog updat	Merchandising	Update catalog		
18	Merchandising creates new catalog	Special promotion details	Merchandising	Create special promotion		
19	Merchandising creates new catalog	New catalog	Merchandising	Create new catalog	Catalog	Customer and prospective customer
20	Time to produce catalog activity reports	"End of month"		Produce catalog activity reports	Catalog activity reports	Merchandising

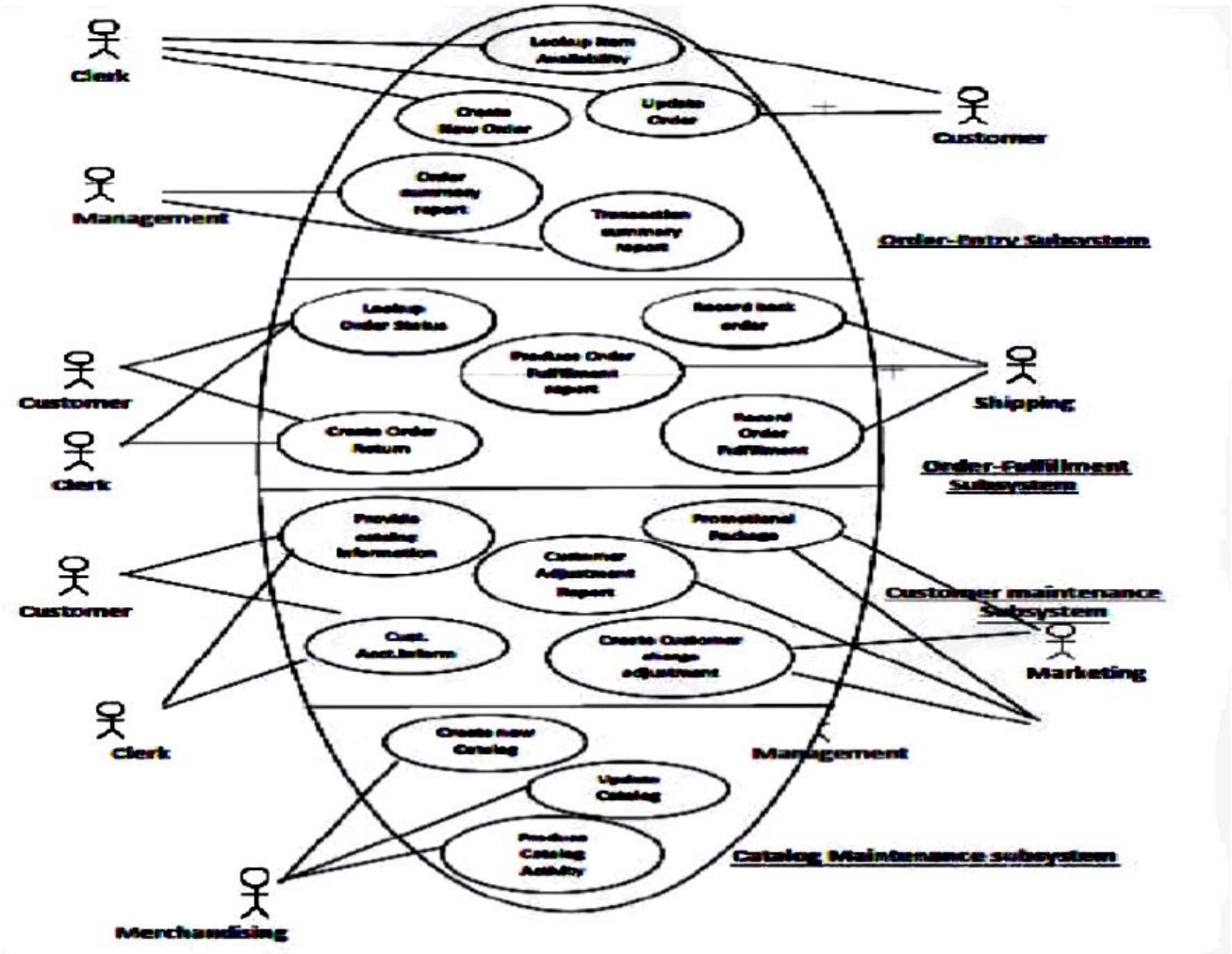
d) Identify & analyze domain classes

Domain classes: A domain model is a representation of real-world conceptual classes, not of software components. It is *not* a set of diagrams describing software classes, or software objects with responsibilities. Using UML notation, a domain model is illustrated with a set of **class diagrams** in which no operations are defined

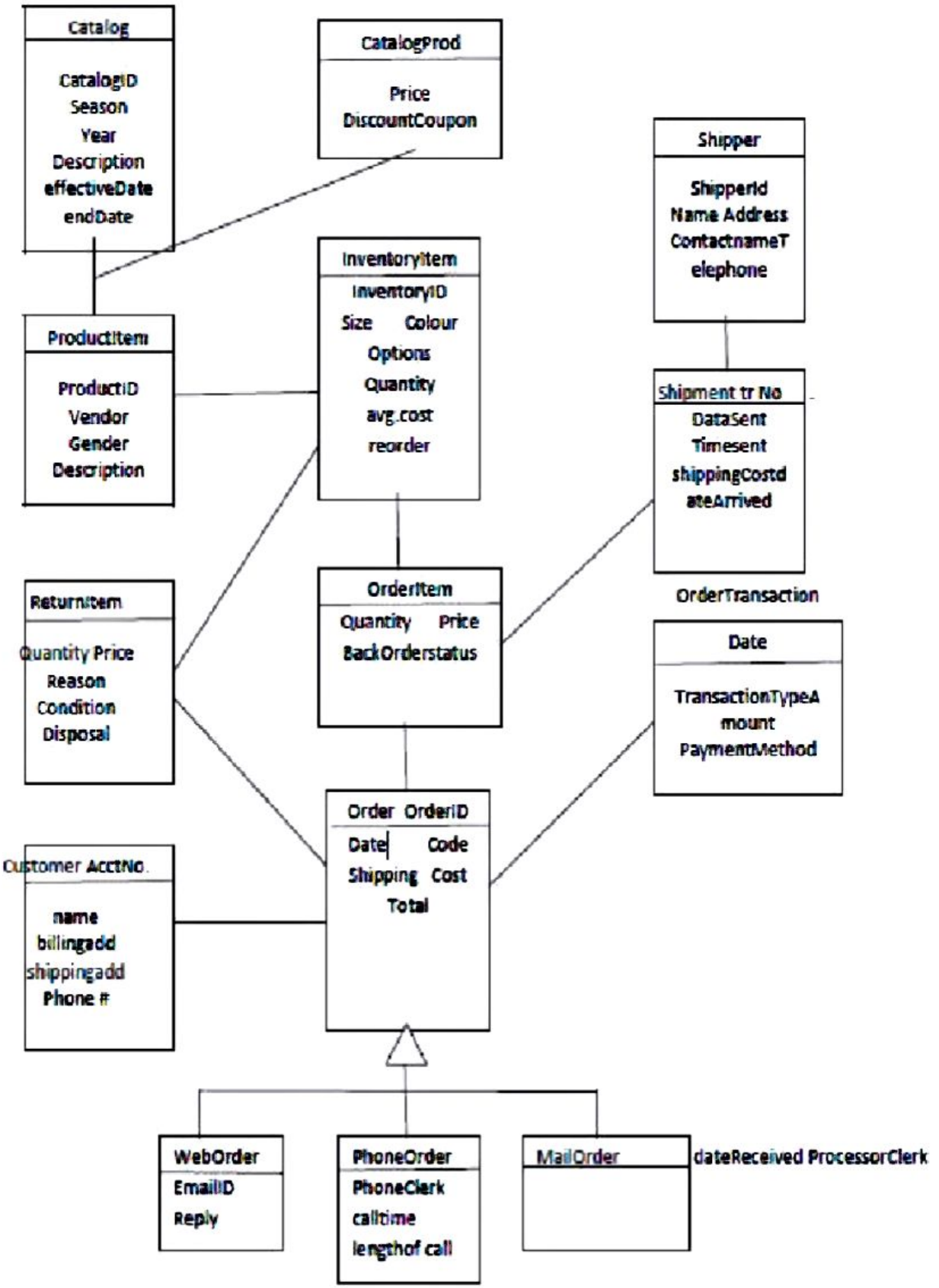
e) Represent use cases and a domain class diagram using Umbrello

Use Case Diagram for Customer Support System

DOMAIN CLASSES	Actor / Use Case
Catalog	• Customer views • Customer creates • Customer Updates
Customer	• Customer creates new account • Updates account information
Inventory	• List of Products • Updating the catalog • Creating customer catalog
Order	• Adding to the cart • Placing order
Transactions	• Charge adjustment details • Creating transaction invoice
Packaging	• Acquiring order information • Packing the product • Placing the invoice on the package • Assigning to the shipment
Shipment	• Acquires the packaging order • Sets the delivery status
Shipper	• Acquires packaging orders • Delivers to the customer



Class diagram for Customer Support System



f) **Developing CRUD matrix to represent relationships between use cases and problem domain classes.**

- The CRUD Matrix is an excellent technique to identify the Tables in a Database which are used in any User interaction with a Web Site.
- CRUD means 'Create, Read, Update or Delete', and the CRUD Matrix identifies the Tables involved in any CRUD operation.
- It is very valuable to combine a CRUD Matrix with the analysis of possible User Scenarios for the Web Site
- The analysis helps to identify any Tables which are not used, and any Tables which are used heavily, and may therefore be a performance bottleneck.

USE CASES	DOMAIN CLASSES										
	Catalog	Customer	Inventory Item	Order	Order Item	Order Transaction	Package	Product Item	Return Item	Shipment	Shipper
Look up item availability			R								
Create new order		CRU	RU	C	C	C	R	R		C	R
Update Order		RU	RU	RUD	RUD	RUD	R	R		CRUD	R
Look Up ord		R		R	R	R				R	R
Record order Fulfillment					RU					RU	
Record bac					RU					CRU	
Create order		CRU		RU		C			C		
Provide Catalog Info	R		R				R	R			
Update account custo		CRUD									

Distribute promotional	R	R	R				R	R			
Create customer charge adjust		R U				CRUD					
Update catalog	RU		R				RU	R			
Create special Prom	R		R				R	R			
Create new catalog	C		R				CRU	R			

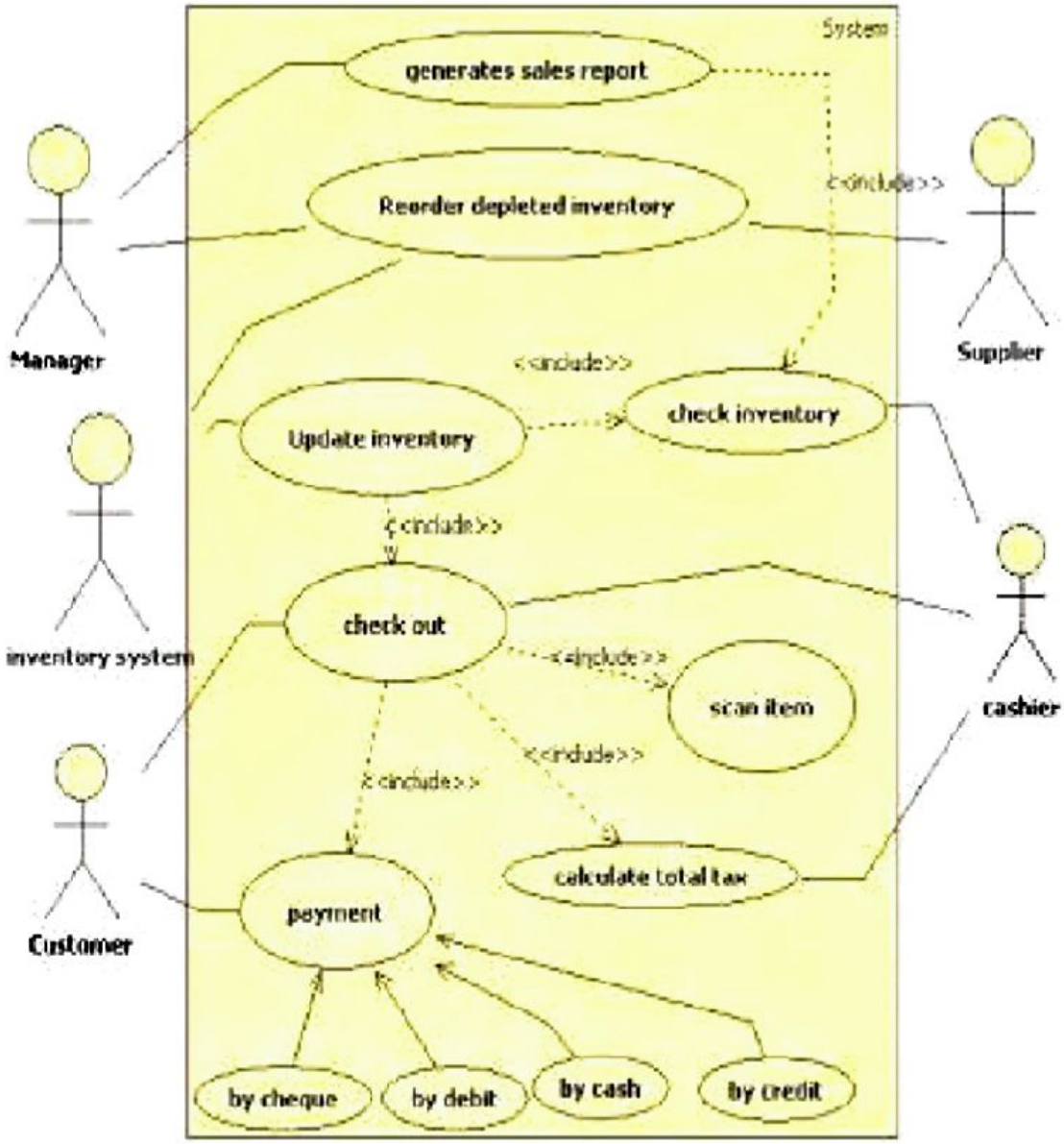
Case study 2:

Point Of Sale Terminal

When the customer arrives at the post check point with the items to purchase. The cashier records each item, price and adds the item information to the running sales transaction. The description and price of the current item are displayed. On completion of the item entry the cashier informs the sales total and tax to the customer. The customer chooses payment type (cash, cheque, credit or debit).After the payment is made the system generates a receipt and automatically updates the inventory. The cashier handovers the receipt to the customer.

USE CASE DIAGRAM: A use case diagram shows a set of use cases and actors (a special kind of class) and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

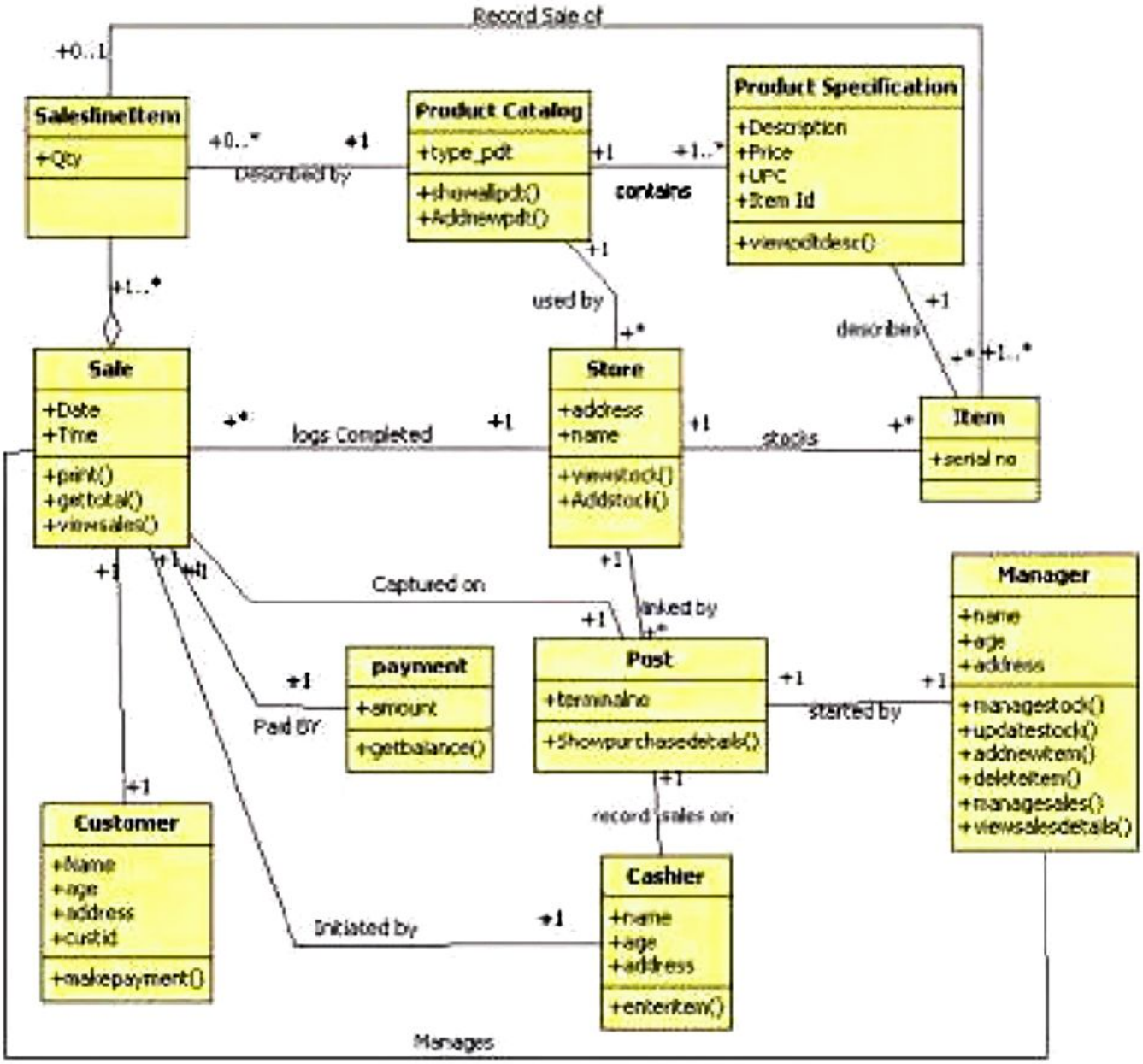
Use case diagram for online transaction



Class diagram for online transaction

A class diagram shows a set of classes, interfaces, and collaborations and their relationships.

These diagrams are the most common diagram found in modeling object-oriented systems. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system.



Case study 3

Library Management System

Identify and analyze the events related to Library Management System :

It is a support system for a library.

- The library lends books and magazines to borrowers, who are registered in the system, as are the books and magazines.
- The library handles the purchase of new titles for the library. Popular titles are bought in multiple copies. Old books and magazines are removed when they are out of date or in poor condition.
- The librarian is an employee of the library who interacts with the customers(borrowers) and whose work is supported by the system.
- A borrower can reserve a book or magazine that is not currently available in the library, so that when it's returned or purchased by the library, that borrower is notified. The reservation is canceled when the borrower checks out the book or magazine or through an explicit canceling procedure.

- The librarian can easily create, update, and delete information about the titles, borrowers, loans, and reservations in the system.
- The system can run on all popular Web browser platforms (Internet Explorer 5.1+, Netscape 4.0+, and so on).
- The system is easy to extend with new functionality.

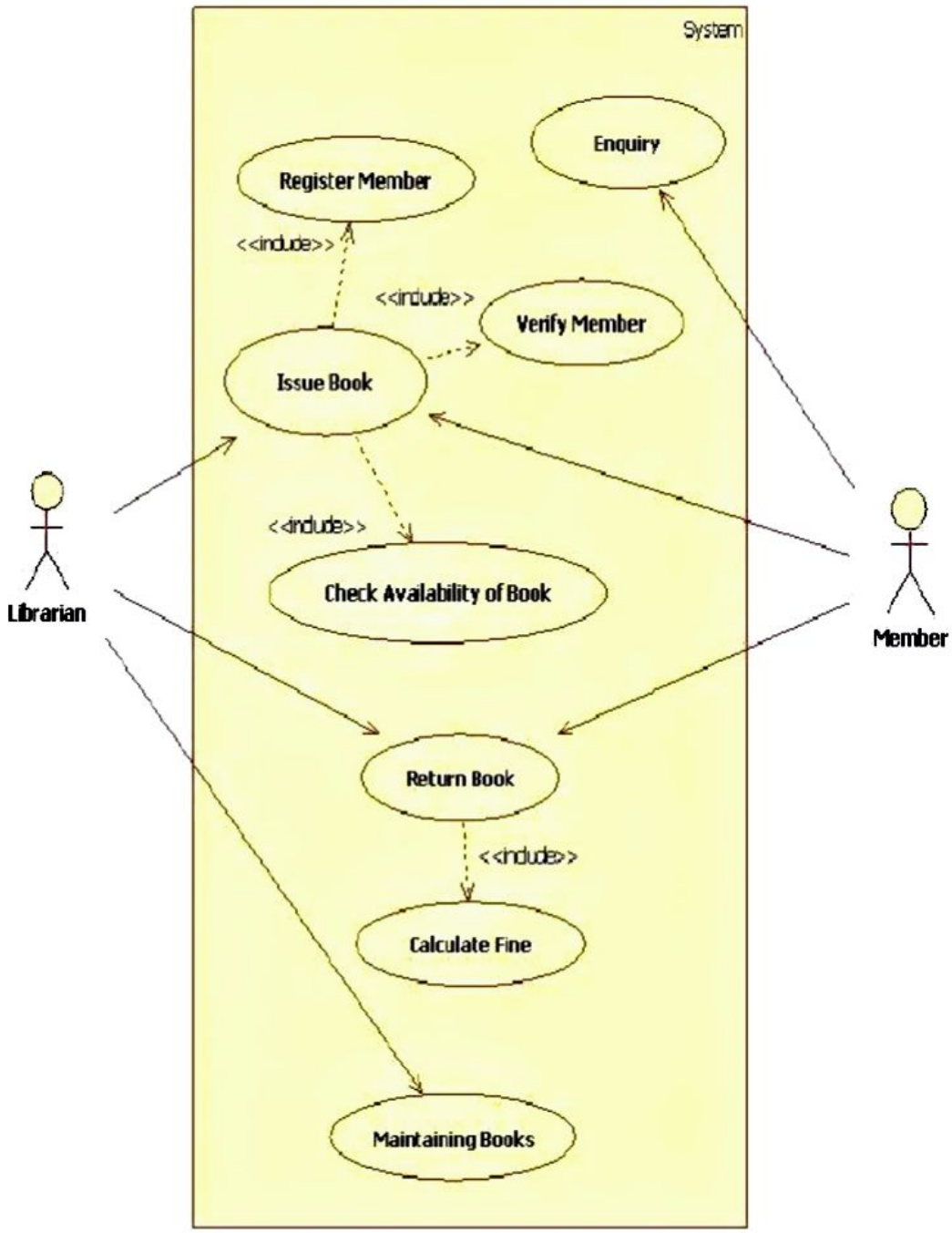
Identify the Use Cases with respective LMS

The use cases in the library system are as follows:

- Login
- Search
- Browse
- Make Reservation
- Remove Reservation
- Checkout Item
- Return Item
- Manage Titles
- Manage Items
- Manage Borrowers
- Manage Librarians
- Assume Identity of Borrower

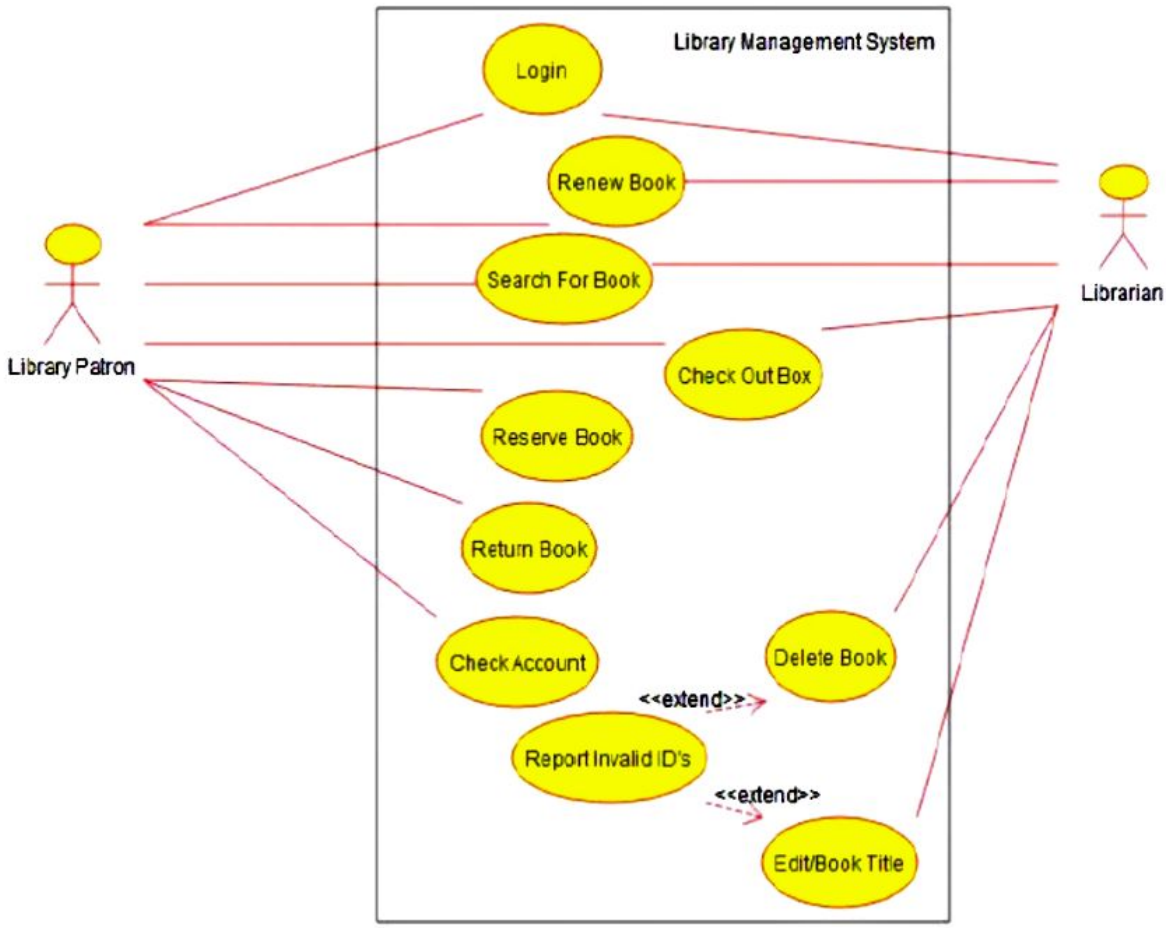
Draw the Use case diagram for Library Management system

Use cases are defined to satisfy the user goals of the primary actors. Hence, the basic procedure is:



Usecase Diagrams using relationships for LMS

A use case diagram shows a set of use cases and actors and their relationships.



Class diagram for Library Management system

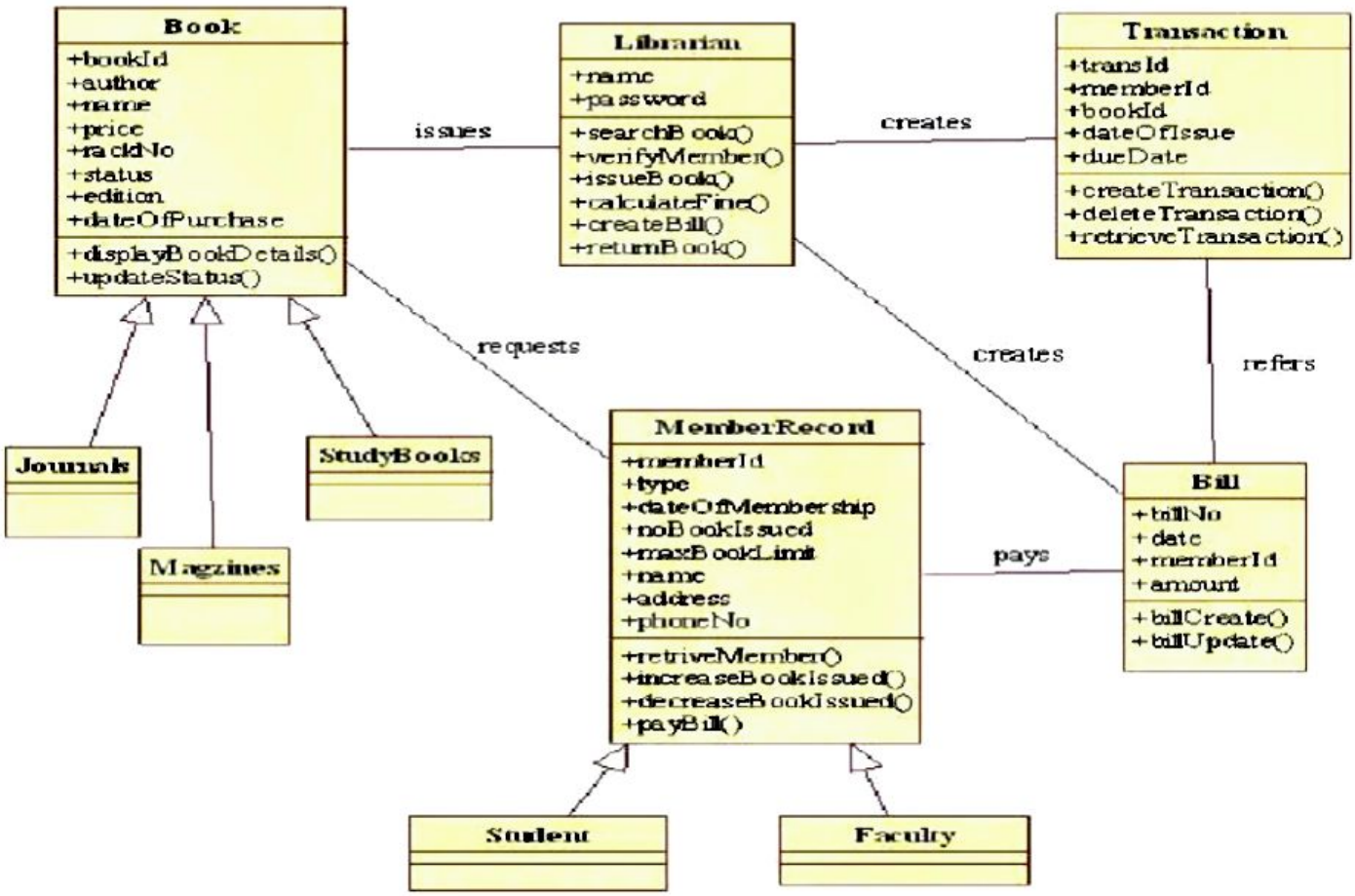
Using UML notation, a domain model is illustrated with a set of **class diagrams** in which no operations are defined. It may show:

Domain objects or conceptual classes

- Associations between conceptual classes
- attributes of conceptual classes

Strategies to Identify Conceptual Classes

- Use a conceptual class category list.
- Identifying Noun Phrases



VIVA QUESTIONS

1. What is an **event**?
2. What is **Use case**?
3. . What is the use of **CRUD** Matrix?
4. What is **Domain class**?
5. What is **Use Case diagram**?
6. What **is Class diagram**?
7. What are the steps for Use cases?

Week-5 & 6

a) Develop Use Case diagrams
b) Develop elaborate Use case descriptions & scenarios
c) Develop Prototypes
d) Develop system sequence diagrams.

Case study 1: Customer Support System

a) Develop Use case diagrams:

USECASE:

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

Steps for designing a Usecase:

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.

EXAMPLE:

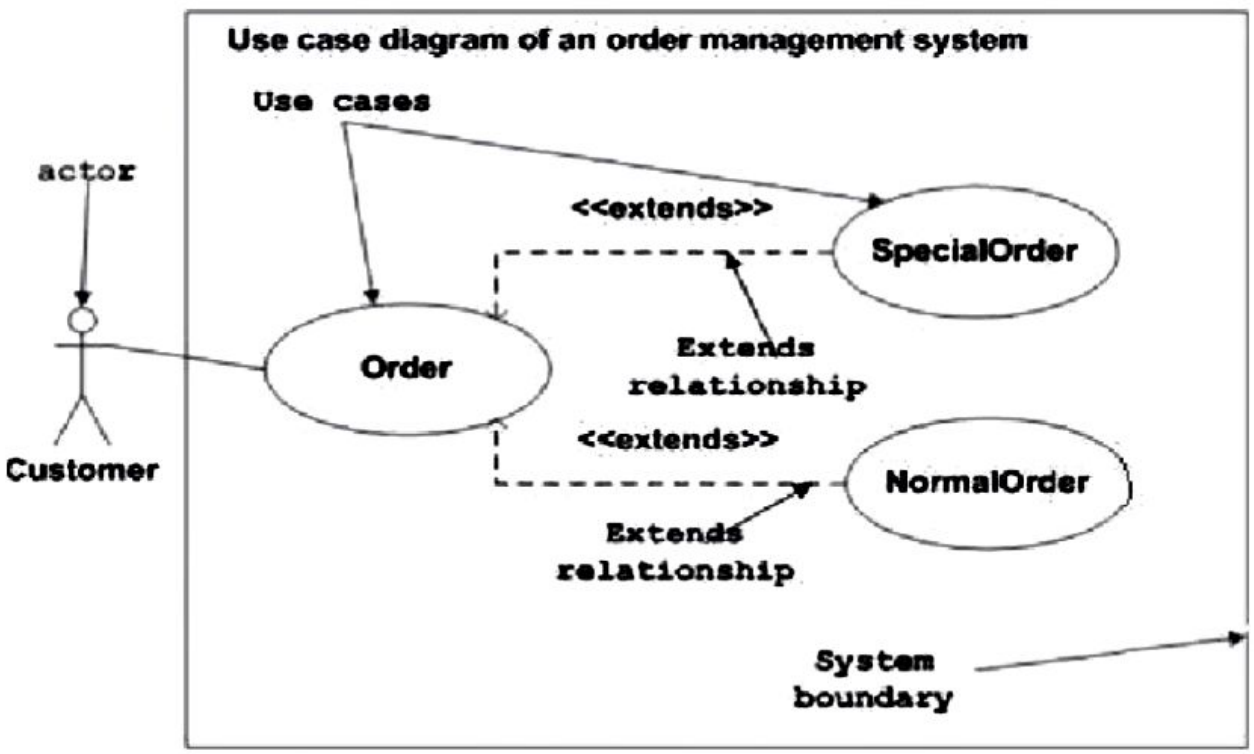


Figure: Sample Use Case diagram

a) Develop elaborate Use case descriptions & scenarios

UseCase Name	Create New Order	
Scenario	Create new Web Order	
Triggering Event	Customer Logs on to website and requests to Purchase an Item	
Brief Description	Customer logs in and requests the new order form. The customer searches the catalog online and purchases items from the catalog. The system adds the purchased items to the order. At the end, the customer enters credit card information	
Actors	Customer	
Related Use Cases	Includes: Register new customer, check item availability	
Stake Holders	Sales Department Shipping department Marketing department	
Pre-Conditions	Catalog, products, Inventory items	
Post-Conditions	Order items must be created Order transaction must be created for the order payment Inventory items must have the quantity on hand up The order must b related to a customer	
Flow of Events	Actor	System
	• Customer connects to homepage and then links to order page • A new customer links to customer acct. info page and adds appropriate inform, to establish acct. • Existing customer logs in directly • Customer	• Create customer new record • Valid customer account • Shopping cart order is created • Display products from catalog based on search • Add item to shipping cart order • Display shopping cart items with total amount

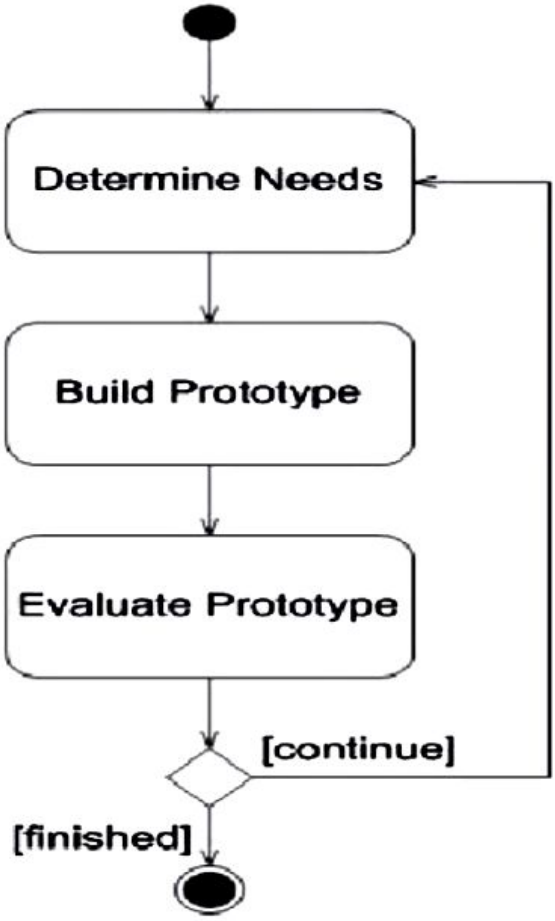
	searches catalog • Customer keeps an order on finding the correct item • Customer may change the order • Customer requests end of order • Customer requests payment screen • Customer enters payment	
Exception Conditions	□ If an item is not in stock then customer can Choose not to purchase item or, Request item be added as a back ordered item □If a customer payment is rejected due to bad credit verification then Order is cancelled, order is put on hold until check is received.	

C)Develop Prototypes:

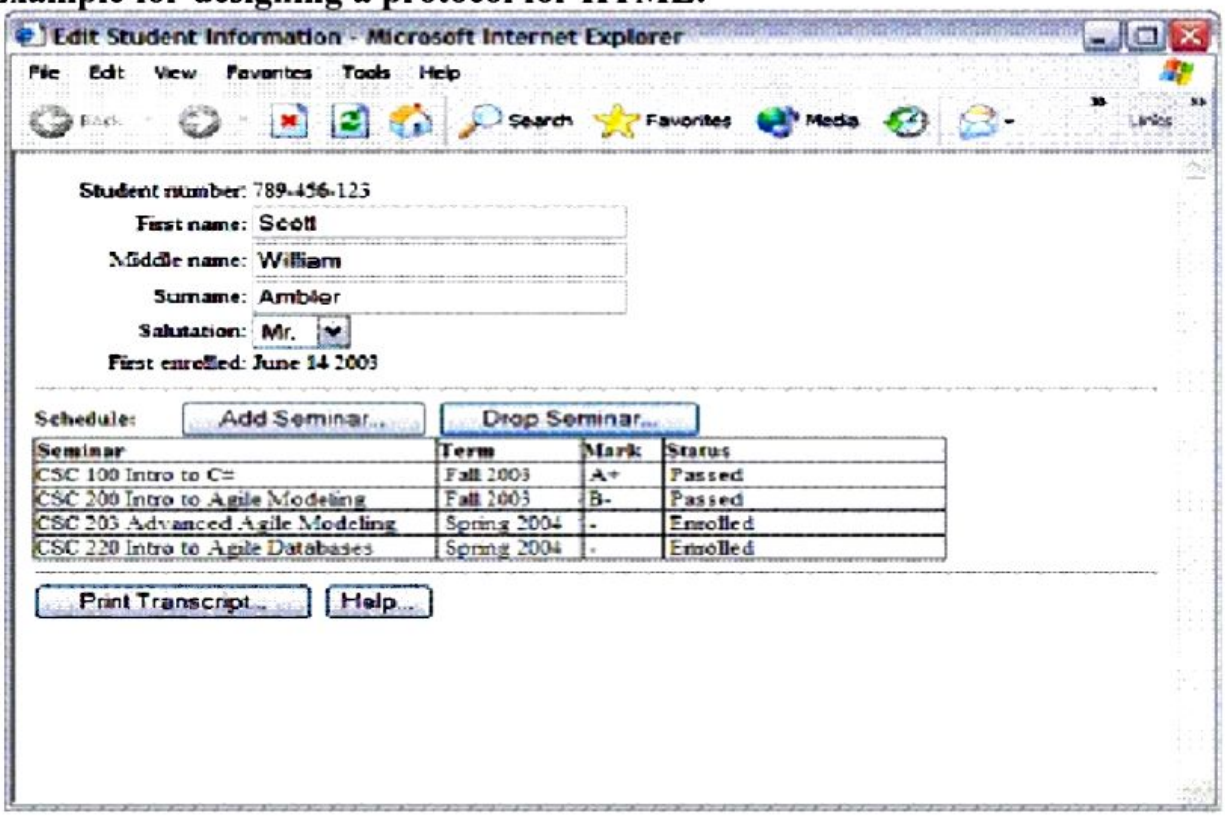
- Prototyping of requirements analysis should demonstrate the important functional effects of the use cases in terms of atomic state changes rather than the detailed algorithms for internal object interactions.
- State changes are mainly about creating new objects or links, removing old objects or links, or modifying object attributes. Therefore, the prototype concentrates on the atomic actions on the SEOD which are about creating an object.

Types of prototypes:

- **Exploratory prototype** helps developing common vision and elicitating user requirements
- **Evolutionary prototoype** supports building the system architecture
- **Experimental prototype** enables feasibility studies
- **mock-up** is a facade of the human computer interface without any functionality
- **functional prototype** is an implementation of a partial system without customizing
- **pilot system** is a system with limited functionality (use cases).



Example for designing a protocol for HTML:



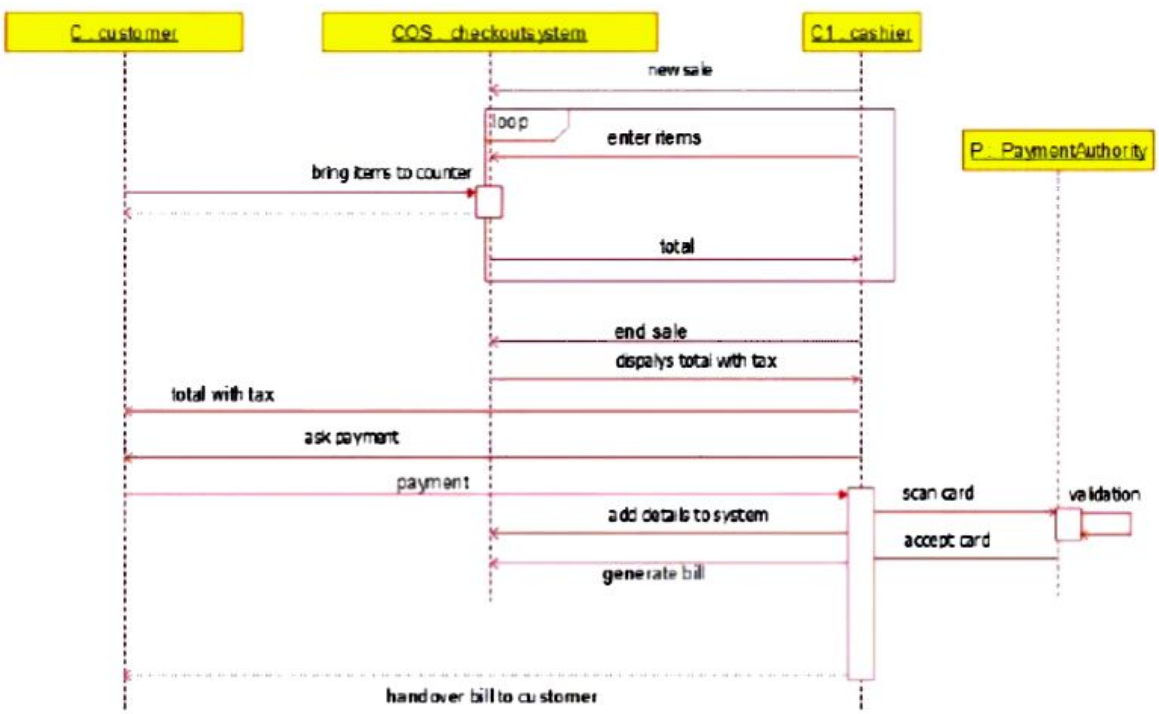
b) Develop system sequence diagrams:

Sequence diagrams:

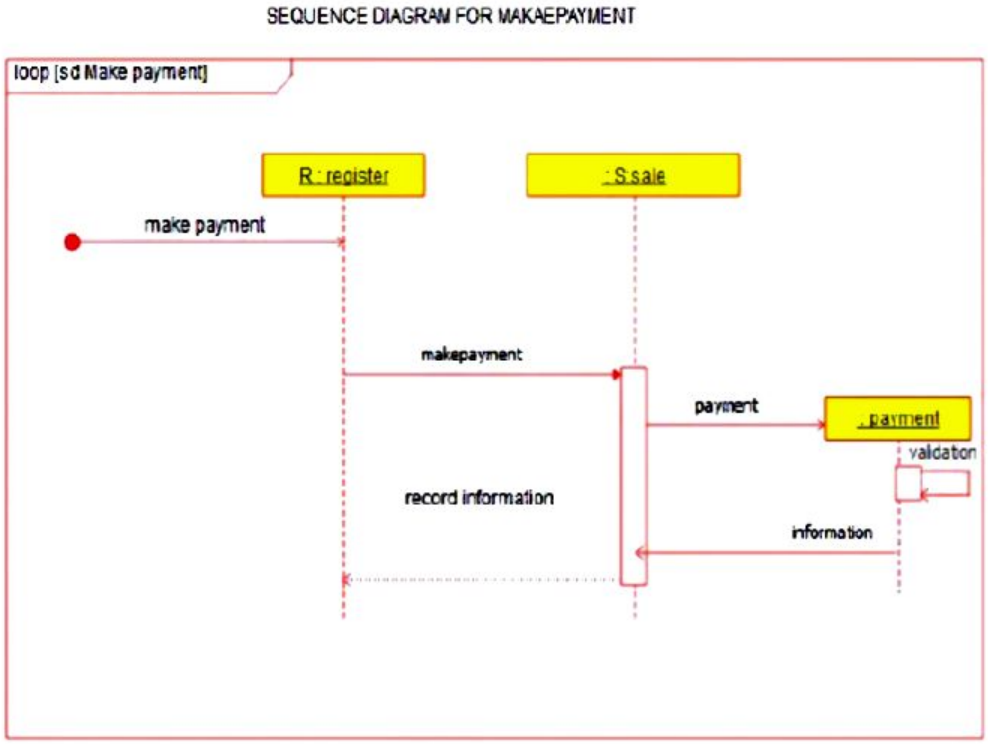
- A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
- The diagram captures the behaviour of a single use case.
- It shows objects and the messages that are passed between these objects in the use case.

When to use a sequence diagram

- A good design can have lots of small methods in different classes. Because of this it can be difficult to figure out the overall sequence of behaviour. This diagram is simple and visually logical, so it is easy to see the sequence of the flow of control.
- A sequence diagram also clearly shows concurrent processes and activations.



Example: sequence diagram for make payment



VIVA QUESTIONS

1) What is meant by **transition**?
2) What does a **message** mean?
3) What is **a sequence diagram**?
4) What is **Interaction diagram**?
5) What are the three different types of message arrows?
6) What is meant by **transition**?
7) Which among these are **the rules to** be considered to form Class diagrams?.
8) An **operation** can be described as?
9) Which UML diagram is shown below?
10. Can you explain primary and secondary actors?
11. **What are Relationships?**
12. **What is a Use Case?**
13. Can you explain 'Extend' and 'Include' in use cases?

Week 7, 8 ,9&10

a) Develop 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
d) Develop detailed design class model
e) Develop three layer package diagram for each case study

Case study 1: Customer Support System

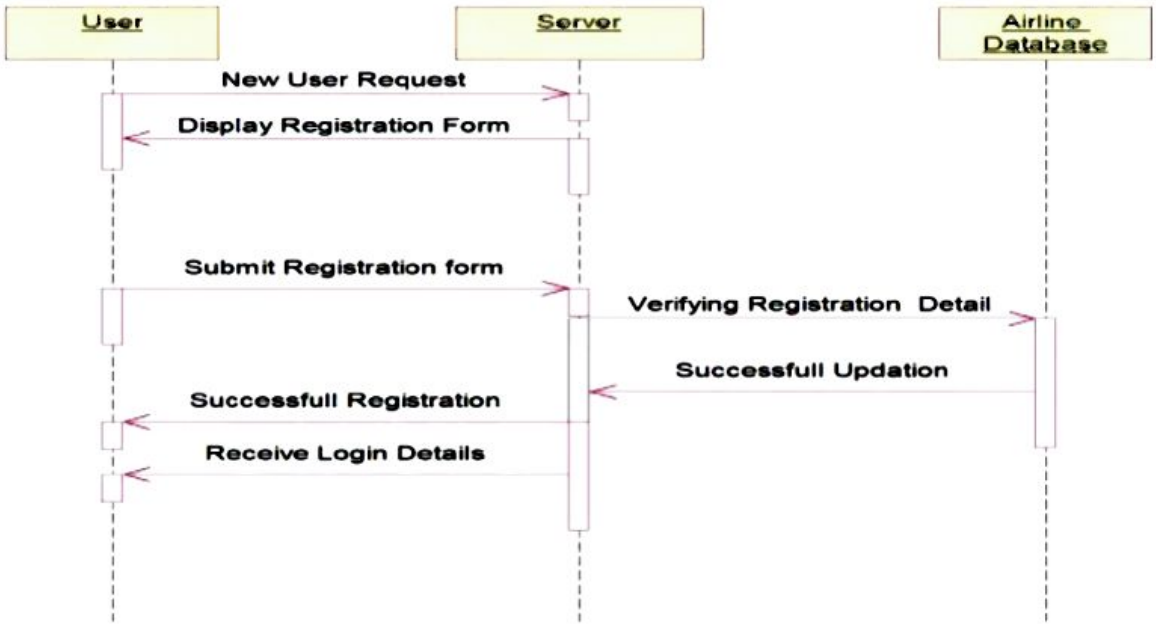
a) Develop high level sequence diagrams for each use case

Sequence diagrams:

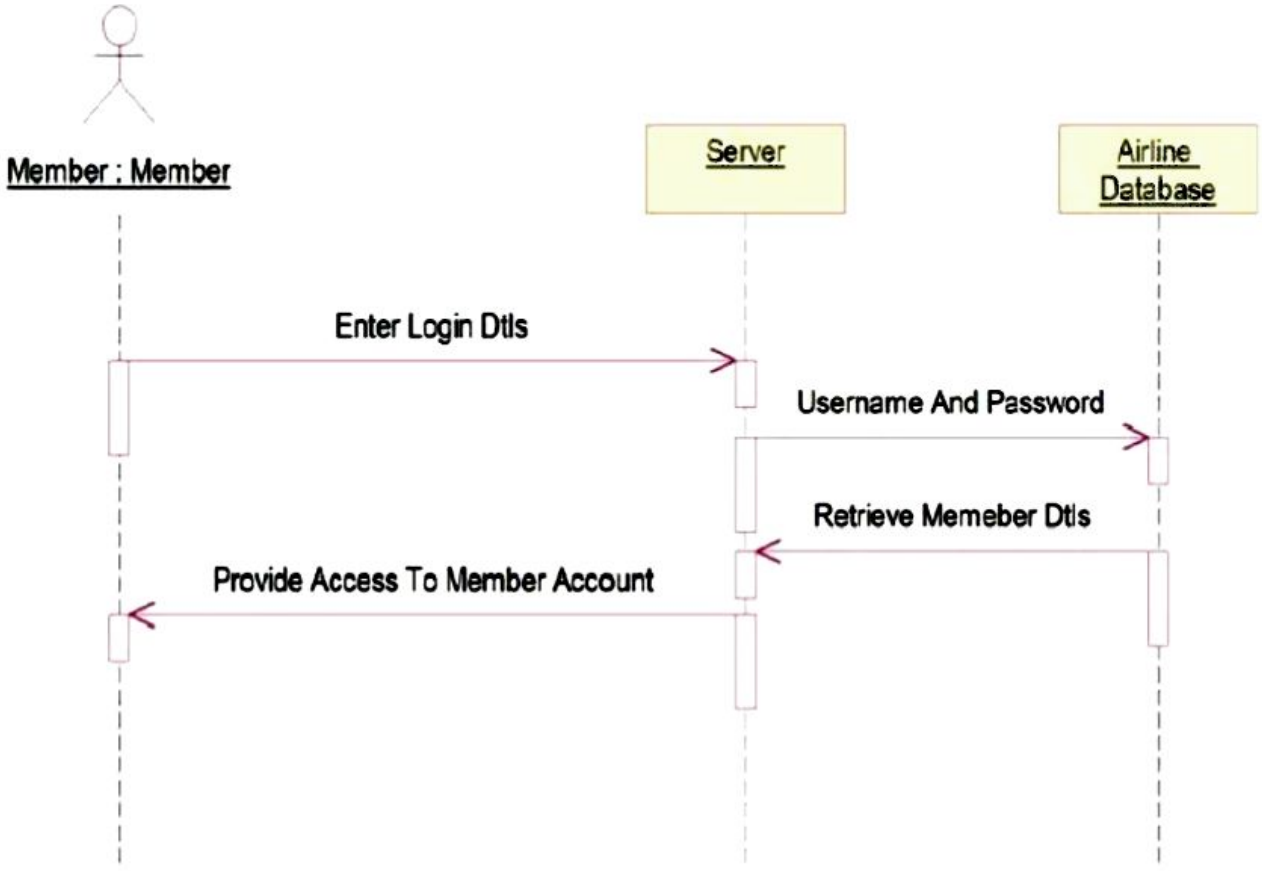
- A sequence diagram is an interaction diagram that emphasizes the time-ordering of messages.
- The diagram captures the behaviour of a single use case.
- It shows objects and the messages that are passed between these objects in the use case.

Sequence diagrams for Airlines

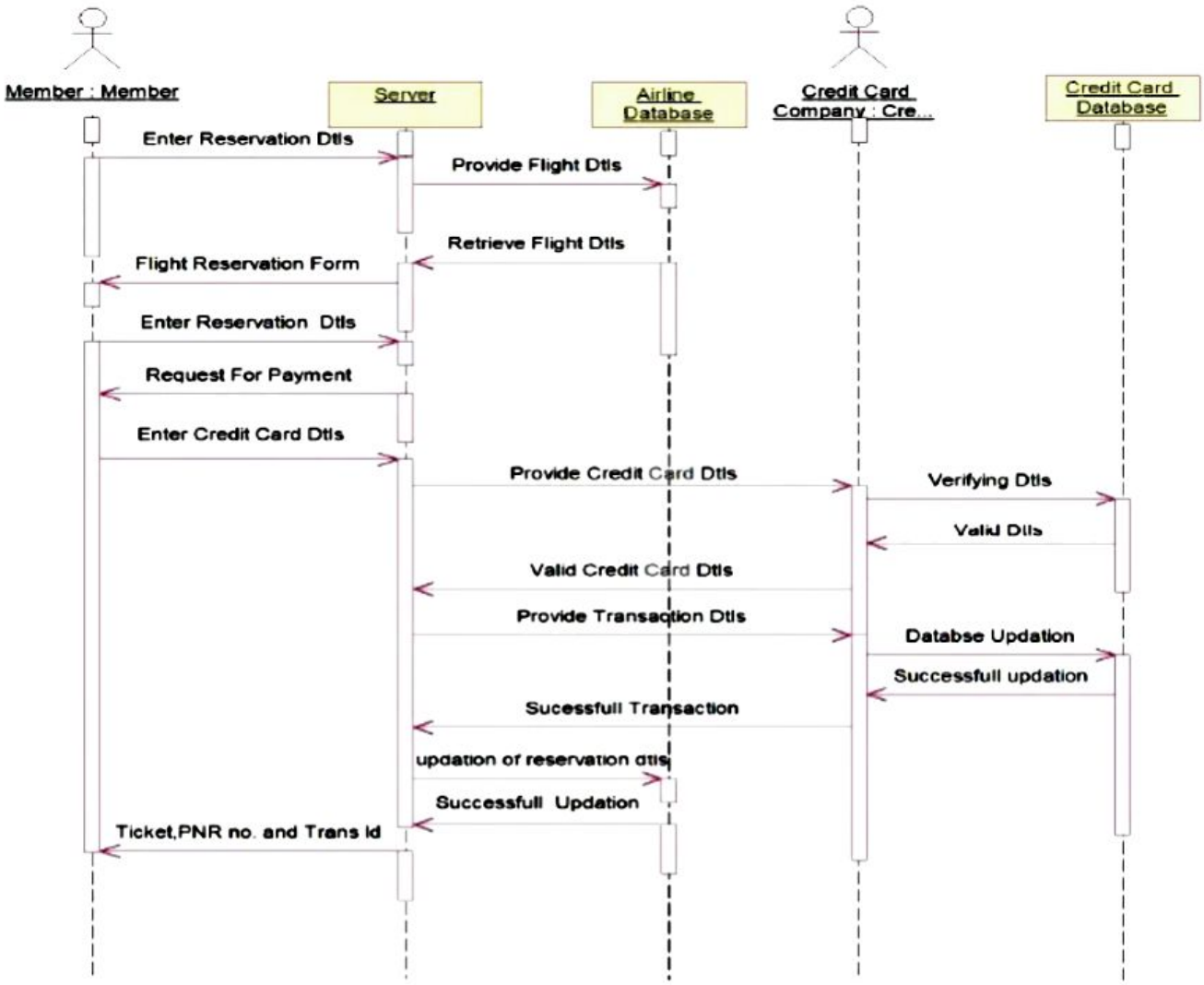
Sequence Diagram for User Registration



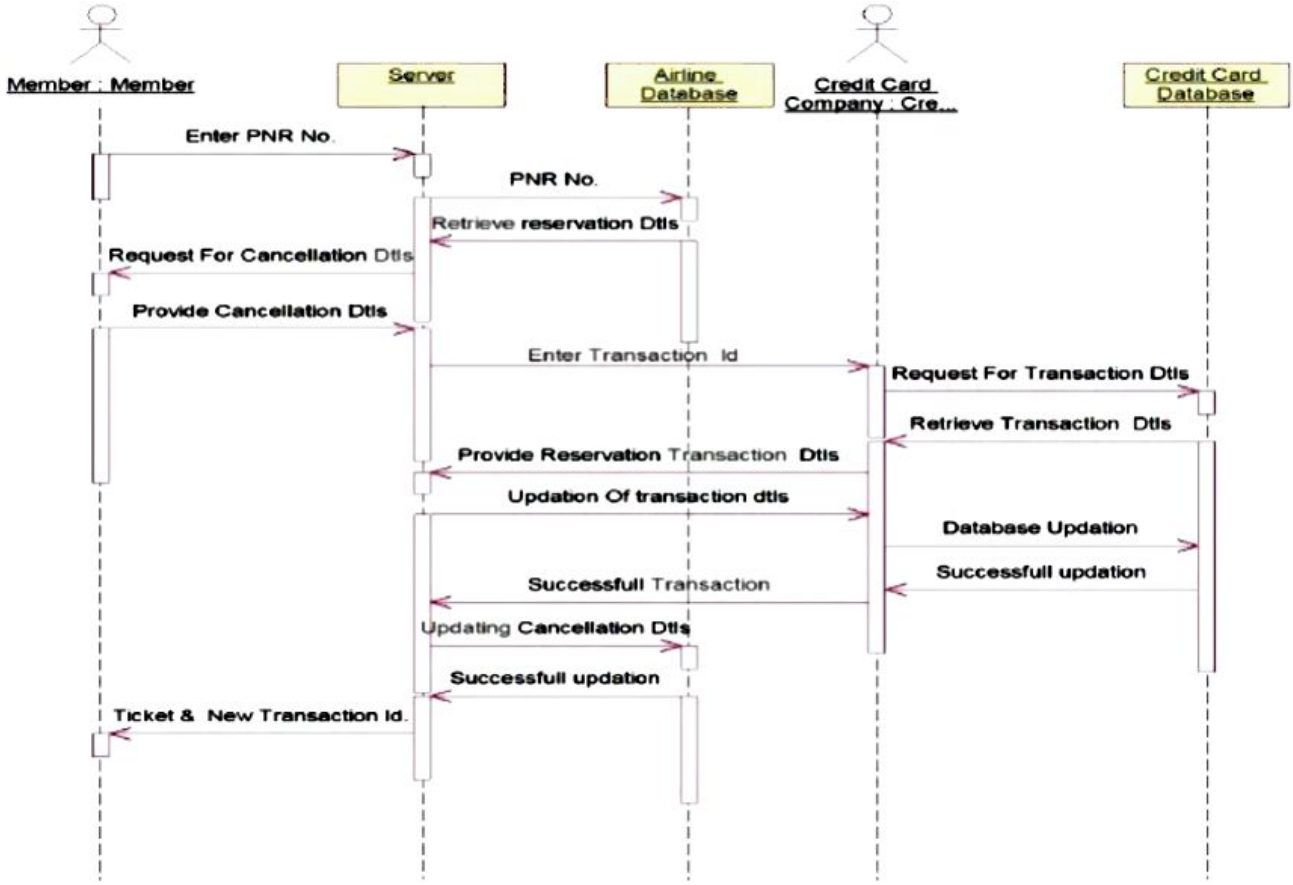
Sequence Diagram for Member Login



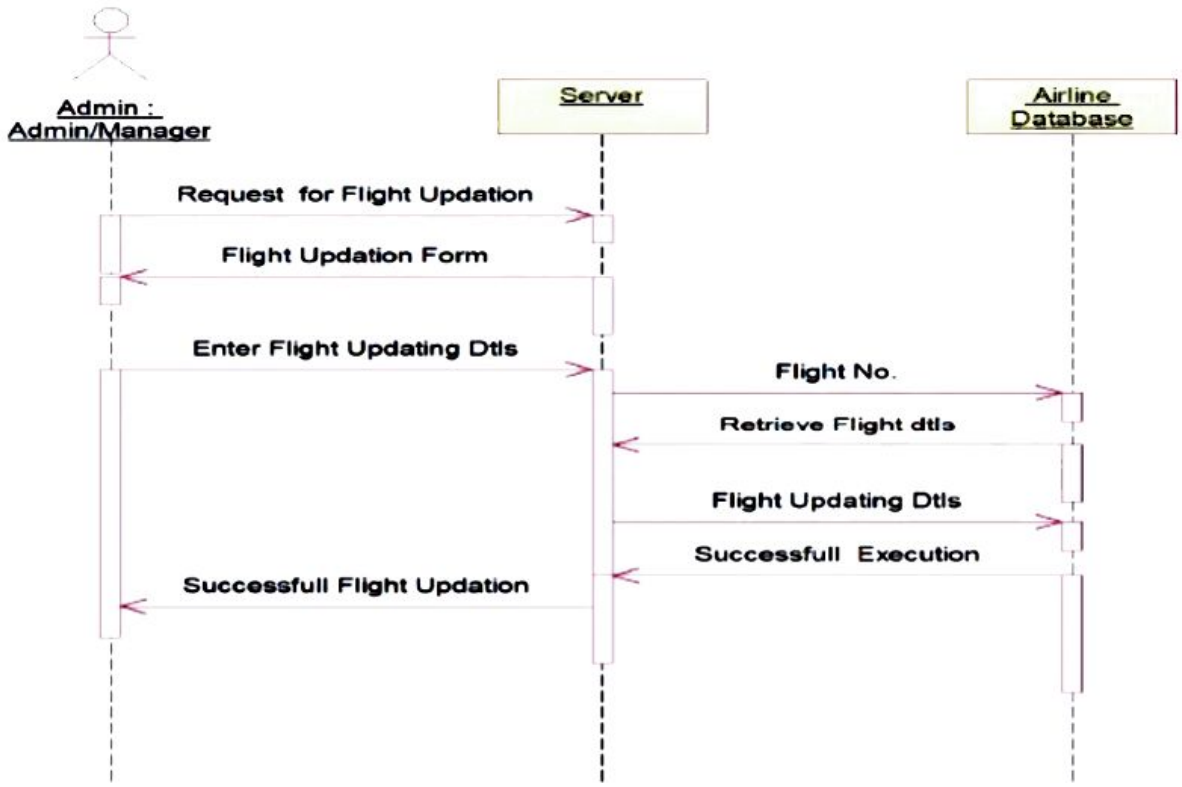
Sequence Diagram for Flight Reservation



Sequence Diagram for Flight Cancellation



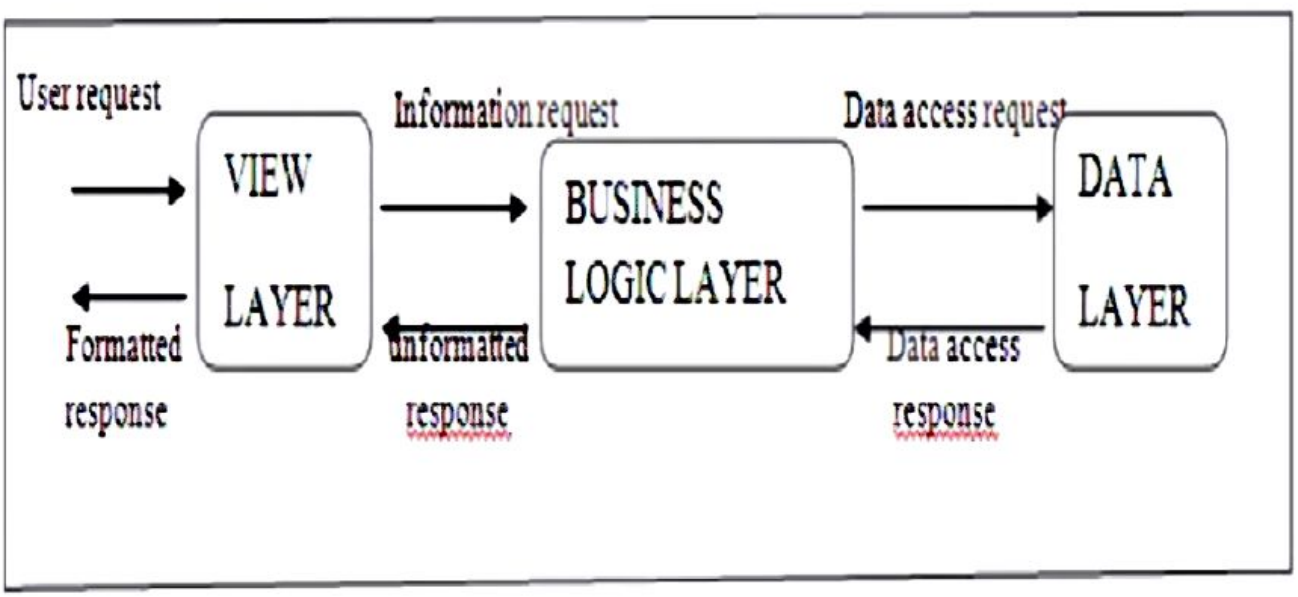
Sequence Diagram for Airline Administration



---

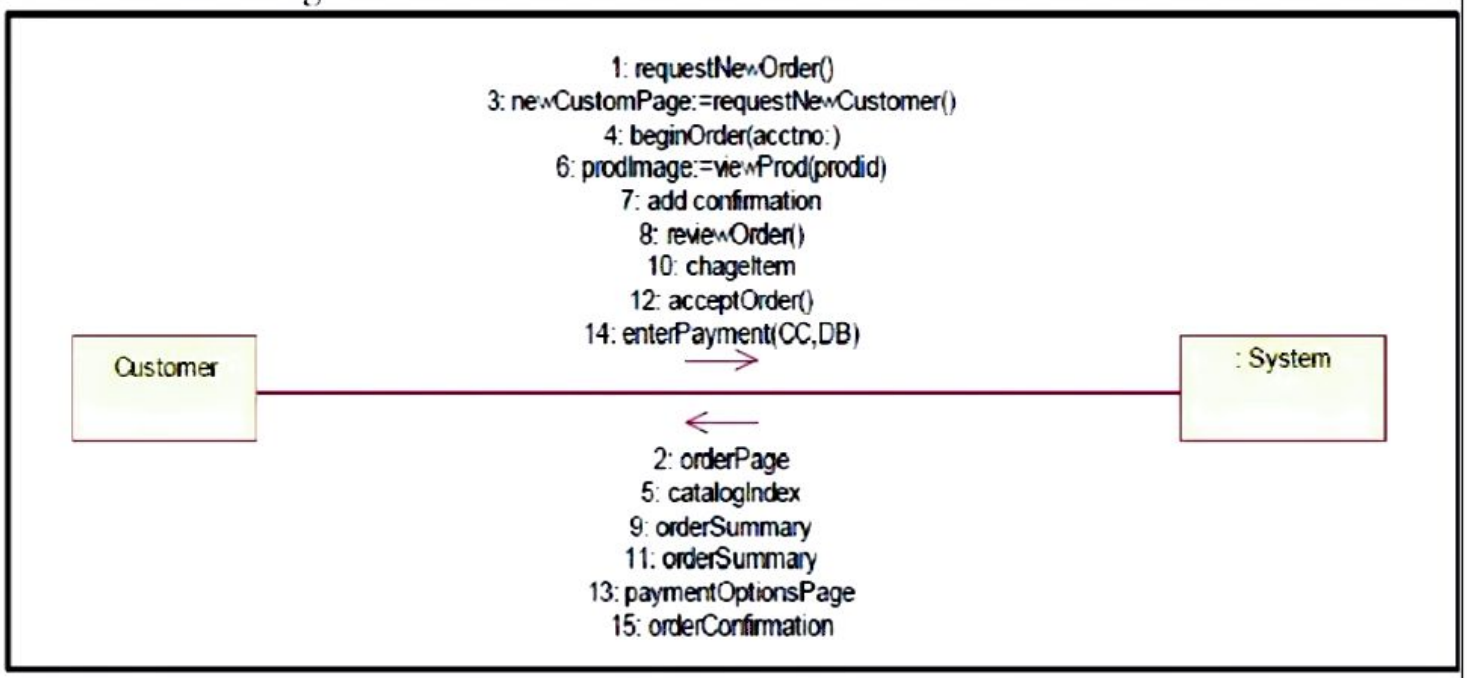
b)Identify MVC classes/objects for usecase

MODEL VIEW CONTROLLER(MVC)

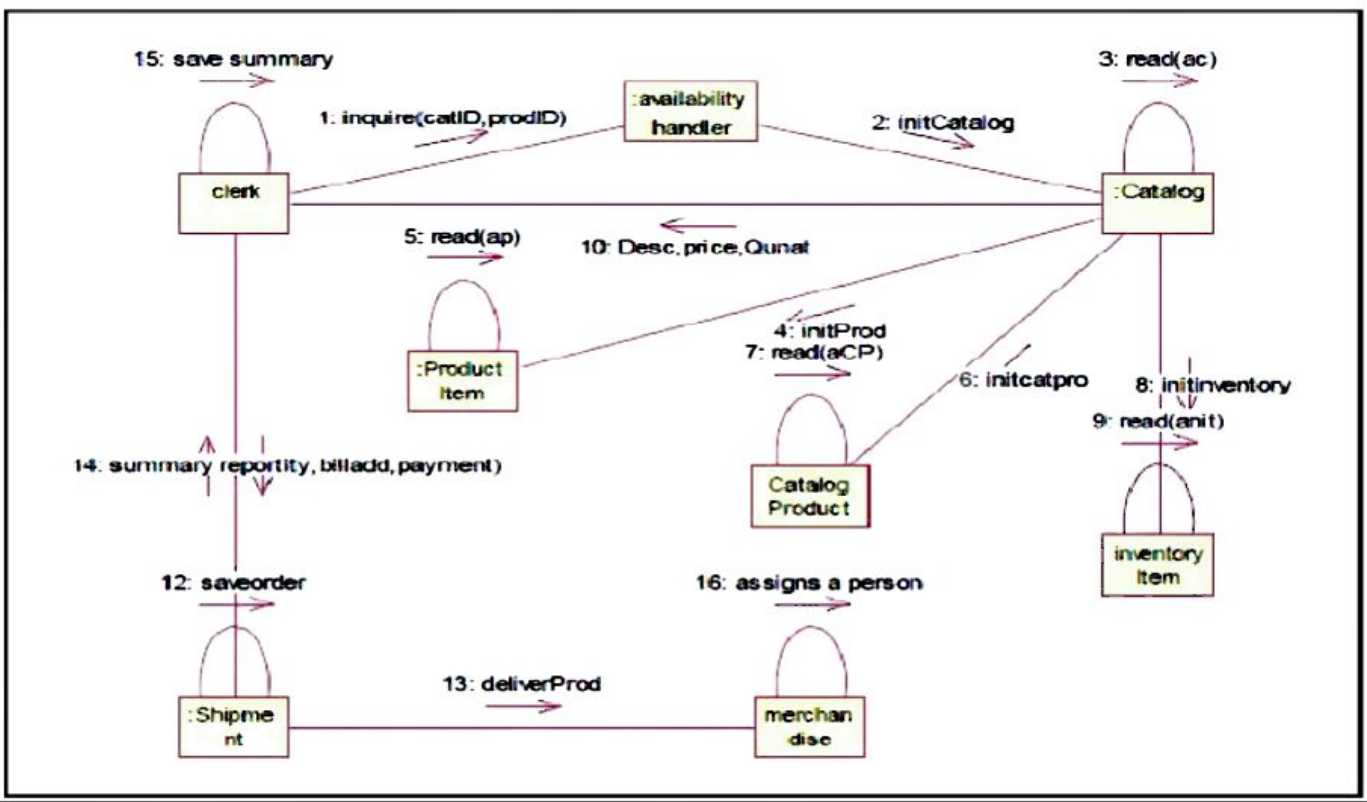


b) <u>Develop detailed sequence diagrams/ communication diagrams for each use case showing interactions</u>

Collobration daigram

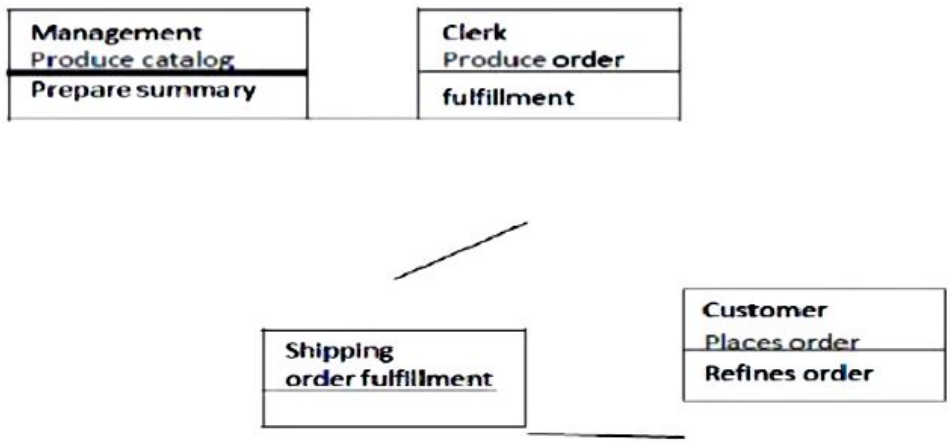


Communication for ech usecase showing with interactions:



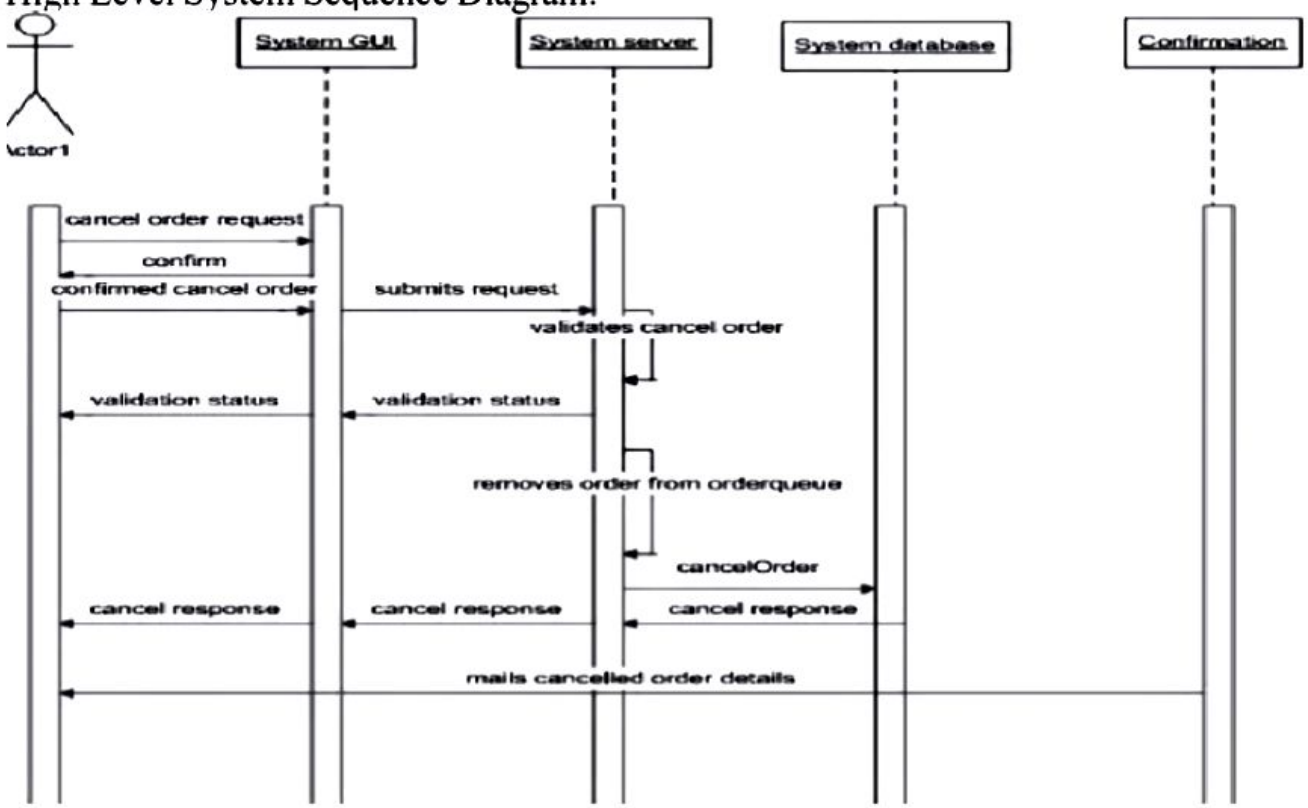
d)Develop detailed design class model

Pattern	Description
INFORMATION EXPERT	Assign a responsibility to the information expert — the class that has the information necessary to fulfill the responsibility



Case study 2: Point Of Sale Terminal

High Level System Sequence Diagram:



Viva Questions:

1. How to Apply the GRASP Patterns?
2. Define patterns.
3. How to Choosing the Initial Domain Object?
4. Explain Grasp: designing objects with responsibilities. -Responsibilities and Methods (RDD)
5. Explain GRASP: Patterns of General Principles in Assigning Responsibilities.
6. How to Determining the Visibility of the Design Model?

Week-11&12

a) Develop Use case Packages

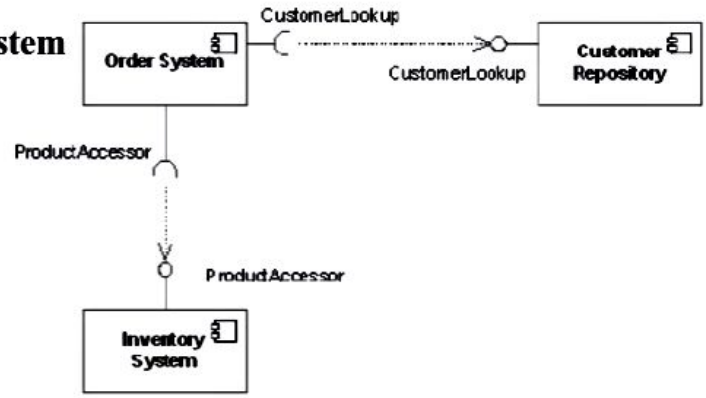
b) Develop Component Diagrams

c) Identify Relationships between use cases and represent them

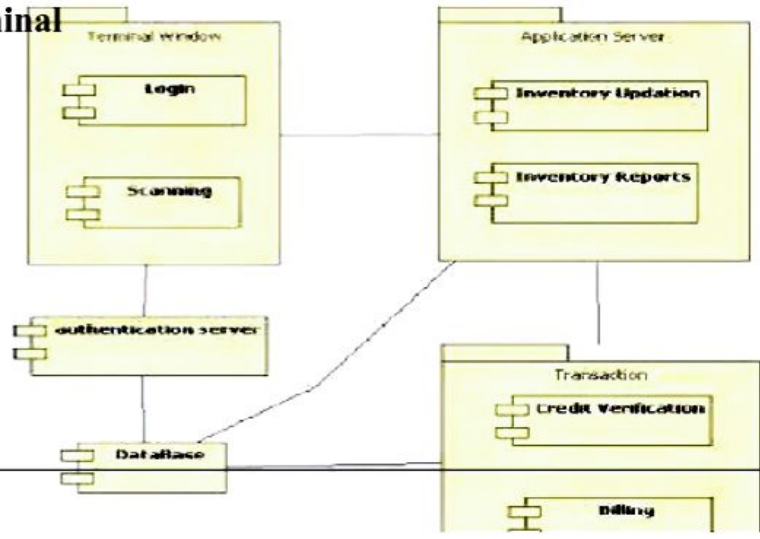
d) Refine domain class model by showing all the associations among classes

Case study 1: Customer Support System

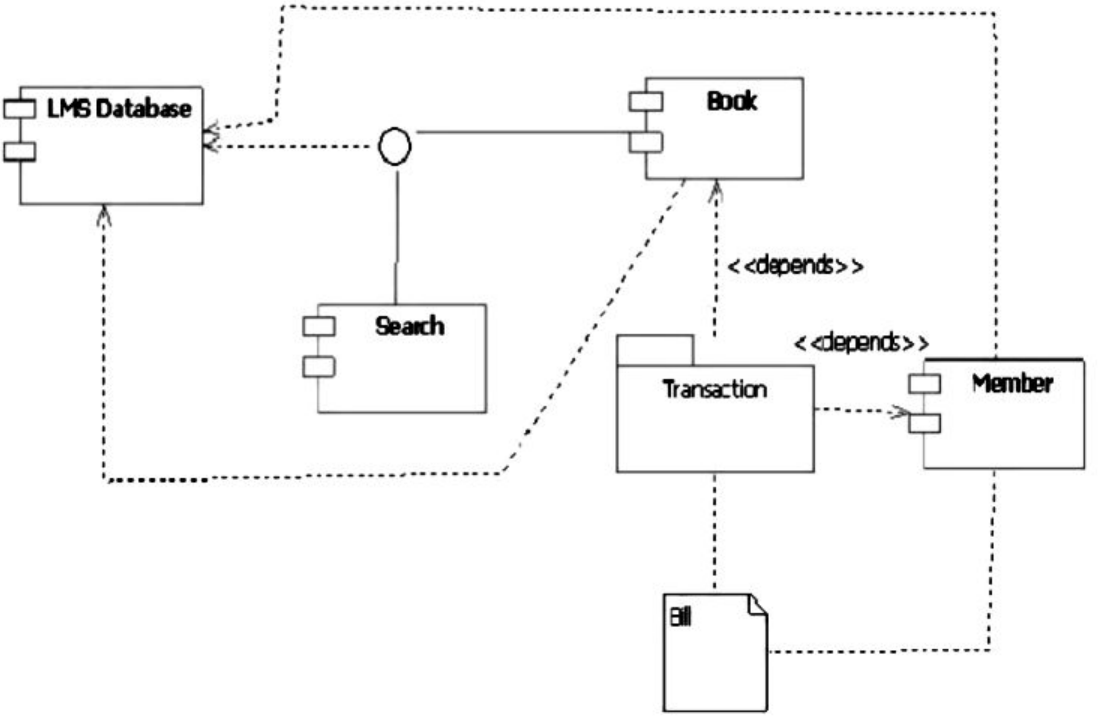
Component Diagram:



Case study 2: Point Of Sale Terminal



Case study 3: Library Management System



Viva questions:

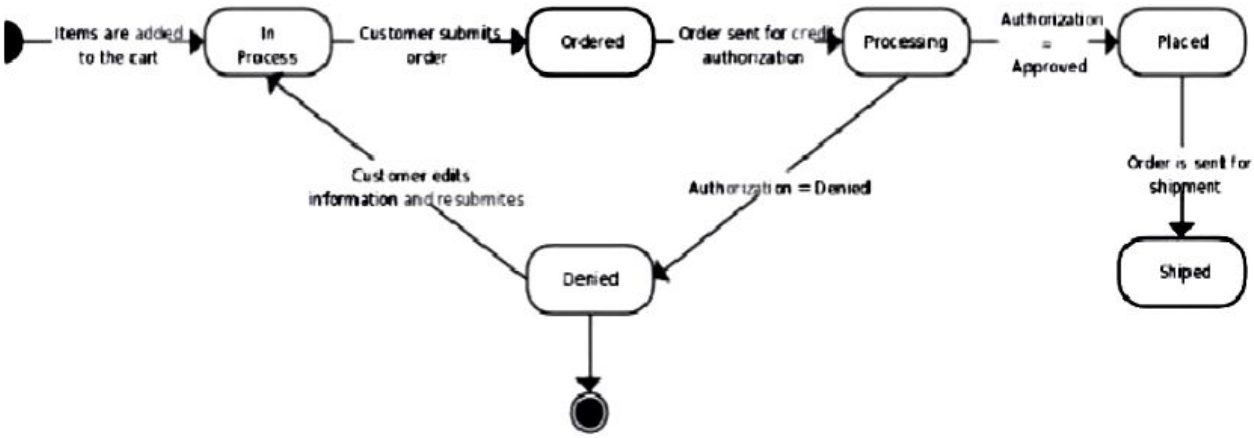
Q1. Define package.
Q2. Can you explain component diagrams?
Q3.Explan use of Component diagrams.
Q4. What are the different views that are considered when building an object-oriented software system?
Q5: What are diagrams?
Q6: What are the major three types of modeling used?
Q7: Mention the different kinds of modeling diagrams used?
Q8: What is Architecture?
Q9: What is SDLC?
Q10: What are Relationships?
Q11: How are the diagrams divided?
Q12: What is Static Diagrams?
Q13: What is Dynamic Diagrams?
Q14: What are Messages?

Week-13 Onwards:

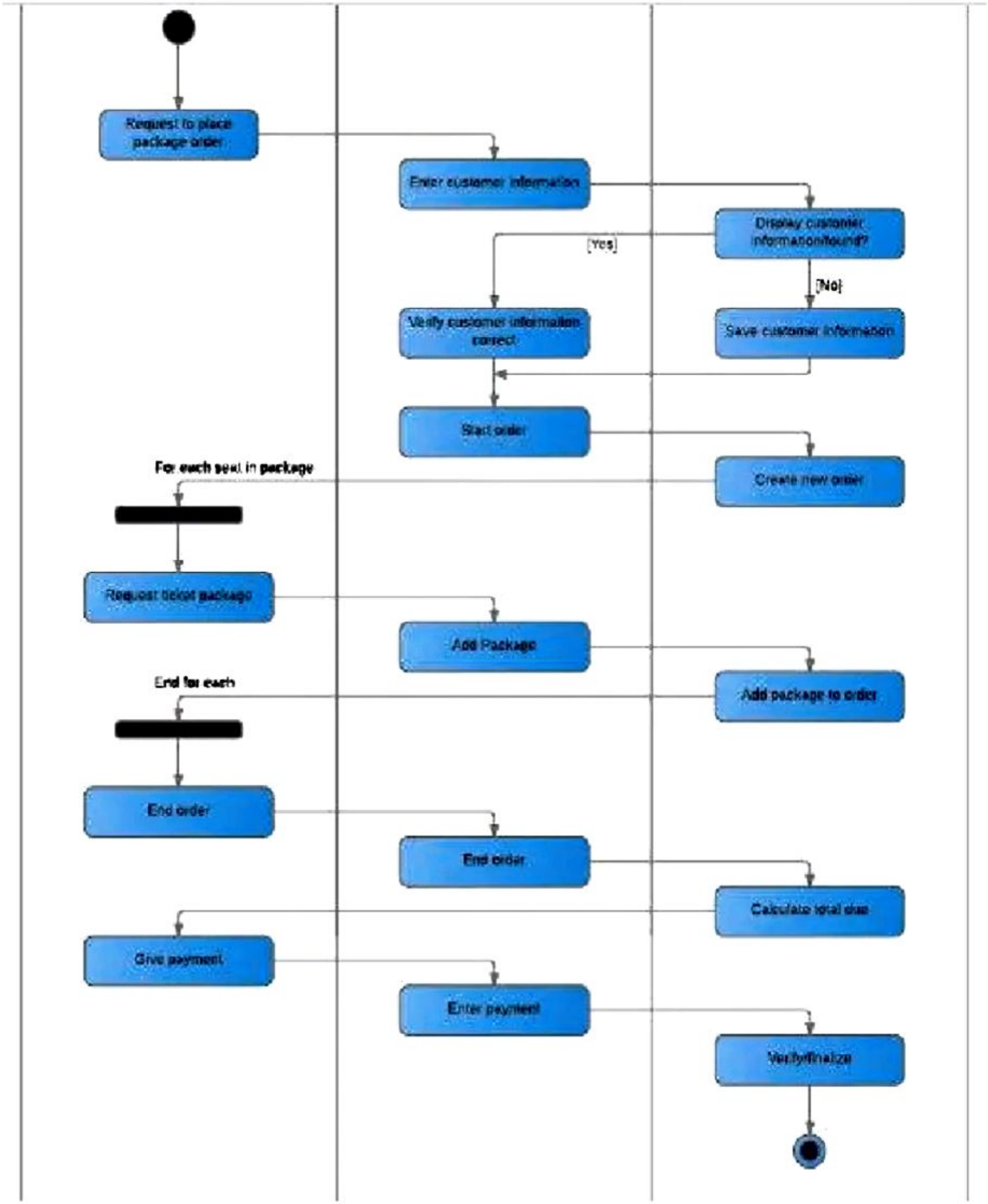
Develop Sample diagrams for state chart diagrams, activity diagrams and deployment diagrams

Case study 1: Customer Support System

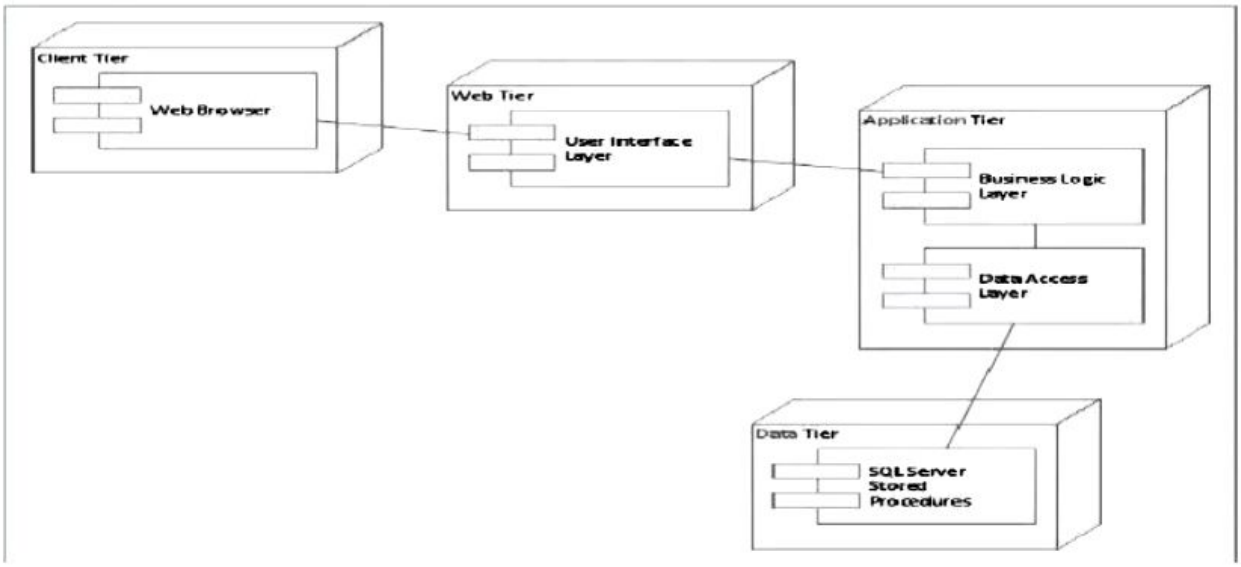
State- chart diagram:



Activity diagram:

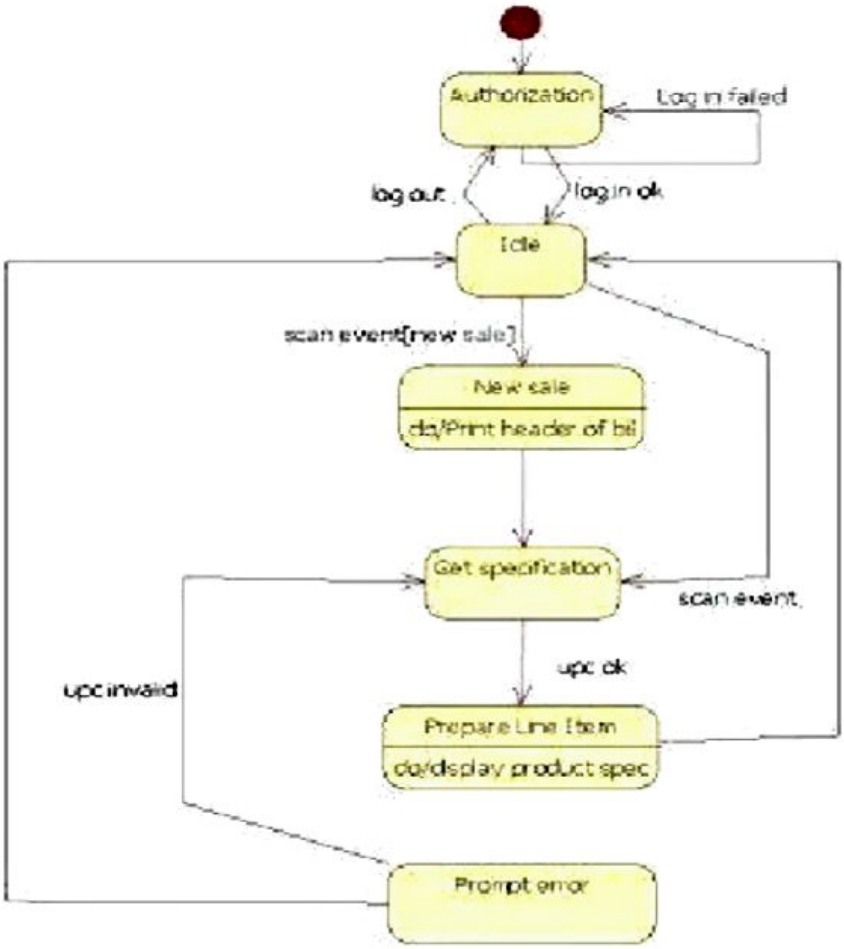


Deployment diagram :

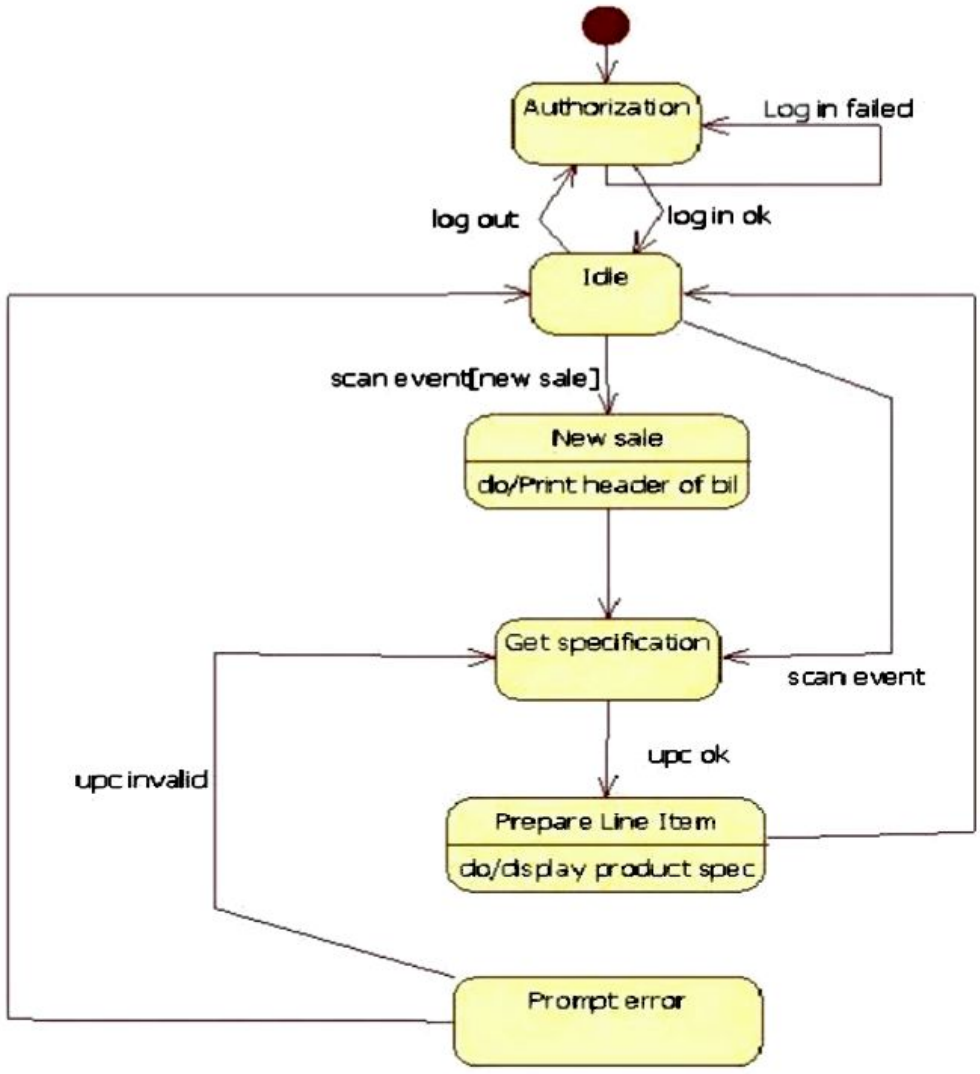


Case study 2: Point Of Sale Terminal

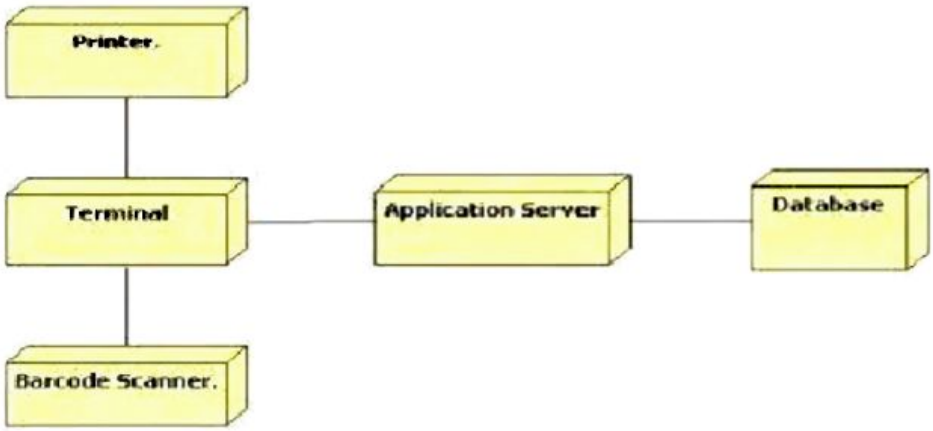
State chart diagram



Activity Diagram:

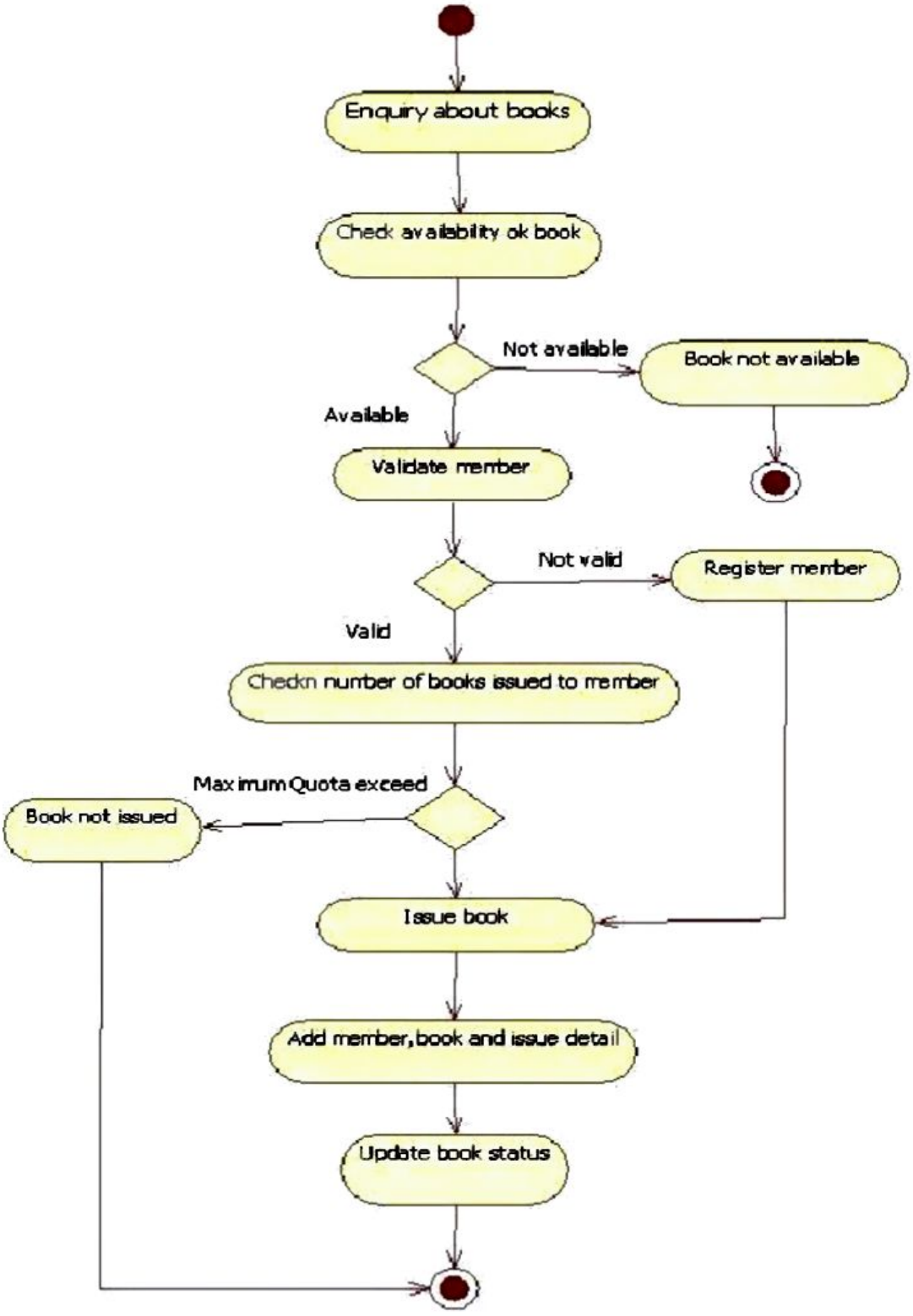


Deployment diagram:



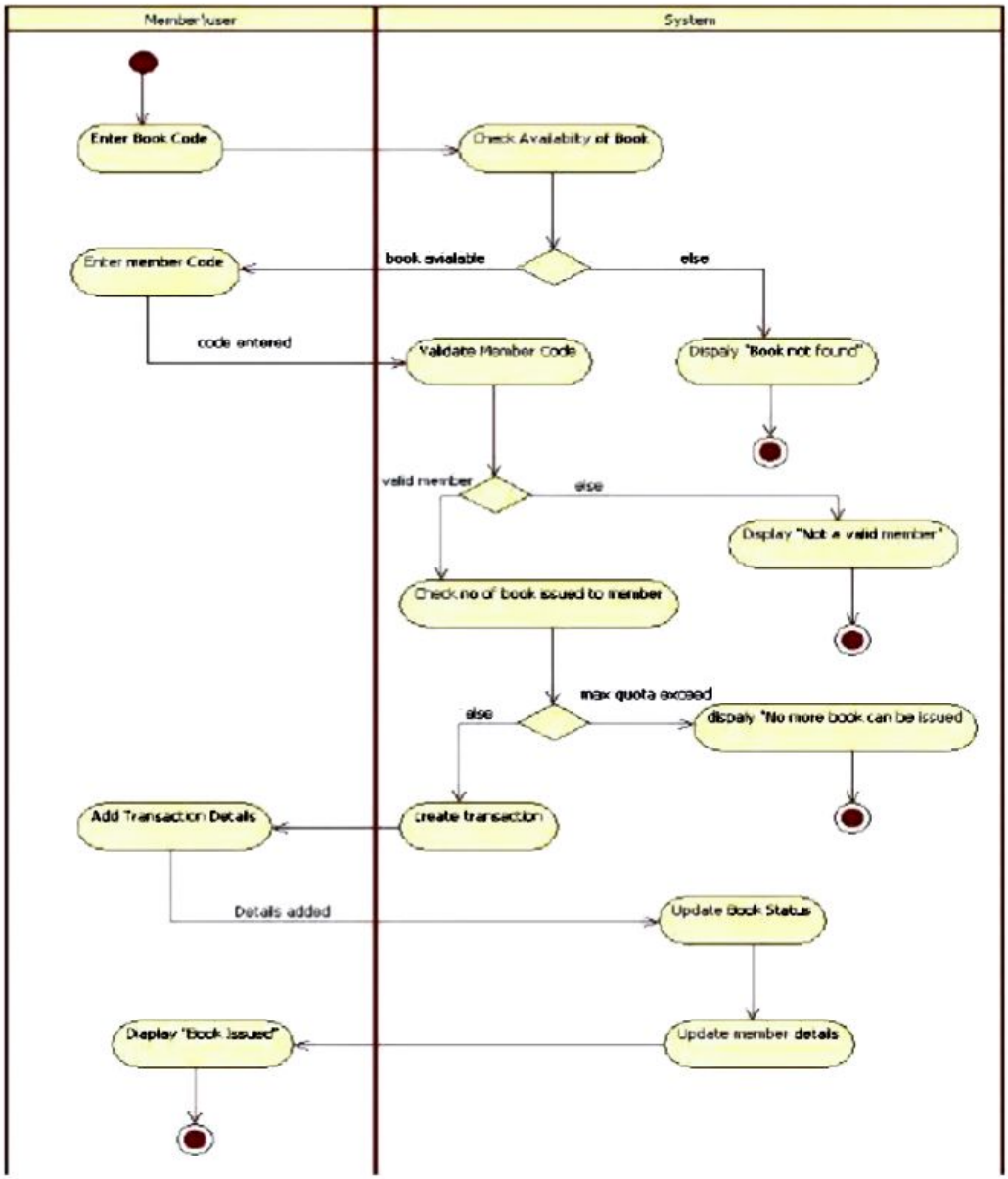
Case study3: Library Management Systems

State chart diagram: State chart diagram is one of the five UML **diagrams** used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events.

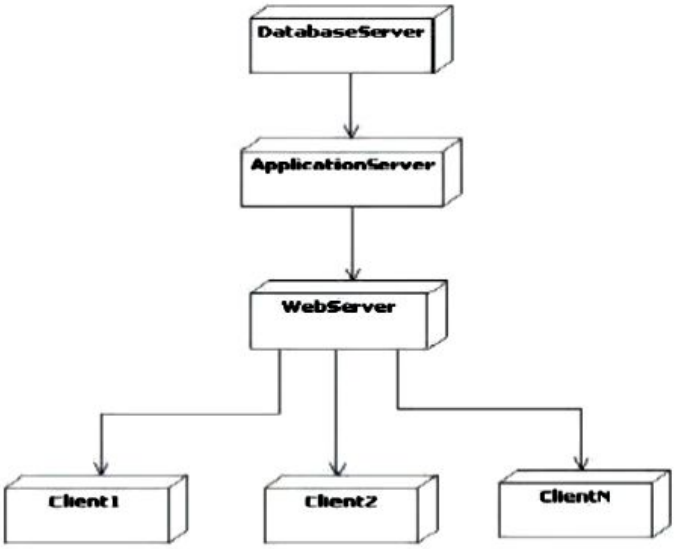


Activity Diagrms:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc



Deployment diagram: Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed. Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.



Viva Questions

Q1 How are the diagrams divided?

Q2.What is Static Diagrams?

Q3.What is Dynamic Diagrams?

Q4.What is Message?

Q5. Define sate chart diagrams.

Q6. What is the purpose of state chart diagrams?

Q7. Explain the use of state chart diagrams?

Q8.Define Activity diagram

Q9. What is the purpose of an activity diagram?

Q10. Explain the use of Activity diagram

Q11. Define Deployment diagrams.