

CSE248: OBJECT ORIENTED ANALYSIS AND DESIGN USING UML**LIST OF EXPERIMENTS**

Sl.	Experiment Name	Pg
1	Frame a problem statement and Design SRS document	2
2	Identify Use Cases and develop the Use Case model	4
3	Identify the conceptual classes and develop a UML Class diagram	9
4	Using the identified scenarios find the interaction between objects and represent them using UML Interaction diagrams	12
5	Develop sequence and Collaboration diagram	17
6	Draw the State Chart diagram.	19
7	Identify the business activities and develop an UML Activity diagram	21
8	Draw Component and Deployment diagrams	25

EXPERIMENT 1

- **AIM**

Frame a problem statement and Design SRS document

- **THEORY**

A problem statement is a concise statement that summarises the current state (problem), the ideal state (goal), and the gap between them.

- ❖ **SAMPLE PROBLEM STATEMENTS AND OBJECTIVES**

1. Passport Automation System

To develop a passport automation system software using UML language. It is the interface between applicant and authority responsible for issuing of the passport. It aims at improving efficiency and reducing complexities.

2. Library Management System

The library management system is a software system that issues books and magazines to registered students only. The student has to login after getting registered to the system. The borrower of the book can perform various functions such as searching for desired book, get the issued book and return the book.

3. Course Reservation System

The system is built to be used by students and managed by an administrator. The student and employee have to login to the system before any processing can be done. The student can see the courses available to him/her and register to the course he/she wants. The administrator can maintain the course details and view all the students who have registered to any course.

- ❖ **SOFTWARE REQUIREMENTS SPECIFICATION(SRS)**

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based on the agreement between customer and contractors.

The essential properties of a good SRS document are the following:

- ❖ **Concise:** The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.
- ❖ **Structured:** It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.

- ❖ **Black-box view:** It should only define what the system should do and refrain from stating how to do these. This means that the SRS document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.
- ❖ **Conceptual integrity:** It should show conceptual integrity so that the reader can merely understand it. **Response to undesired events:** It should characterize acceptable responses to unwanted events. These are called system response to exceptional conditions.
- ❖ **Verifiable:** All requirements of the system, as documented in the SRS document, should be correct. This means that it should be possible to decide whether or not requirements have been met in an implementation.

❖ **TEMPLATE**

Table of Contents	ii
Revision History	ii
1. Introduction	1
1.1 Purpose.....	1
1.2 Document Conventions.....	1
1.3 Intended Audience and Reading Suggestions.....	1
1.4 Product Scope	1
1.5 References.....	1
2. Overall Description	2
2.1 Product Perspective.....	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment.....	2
2.5 Design and Implementation Constraints.....	2
2.6 User Documentation	2
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	3
3.1 User Interfaces	3
3.2 Hardware Interfaces	3
3.3 Software Interfaces	3
3.4 Communications Interfaces	3
4. System Features.....	4
4.1 System Feature 1.....	4
4.2 System Feature 2 (and so on).....	4
5. Other Nonfunctional Requirements	4
5.1 Performance Requirements	4
5.2 Safety Requirements	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
5.5 Business Rules	5
6. Other Requirements	5
Appendix A: Glossary.....	5
Appendix B: Analysis Models	5
Appendix C: To Be Determined List.....	6

• **EXPECTED RESULT/OUTCOME**

A document prepared by following the guidelines of SRS standard tailored to the particular problem statement that is to be carried through to project fruition.

EXPERIMENT 2

- **AIM**

Identify Use Cases and develop the Use Case model

- **THEORY**

A Use-Case describes functionality of the system from the user's point of view. A use case provides developers with a view of what the users want. It is free of technical or implementation details. A Use-Case is represented by an oval in UML. A use case always describes three things: an actor that initiates an event; the event that triggers a use case; and the use case that performs the actions triggered by the event.

A use case scenario is divided into three sections: identification and initiation, steps performed, and conditions, assumptions, and questions.

Use the following four steps to create use case descriptions:

1. Use agile stories, problem definition objectives, user requirements, or a features list as a starting point.
2. Ask about the tasks that must be done to accomplish the transaction. Ask if the use case reads any data or updates any tables.
3. Find out if there are any iterative or looping actions.
4. The use case ends when the customer goal is complete.

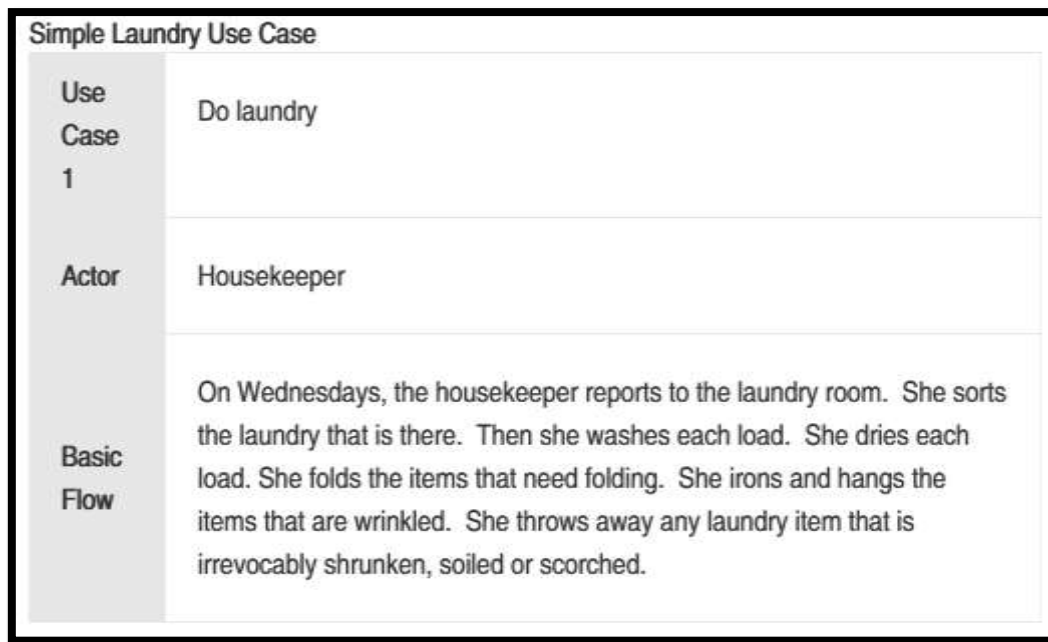
❖ Elements of a Use Case

Depending on how in depth and complex you want or need to get, use cases describe a combination of the following elements:

1. Actor – anyone or anything that performs a behaviour (who is using the system)
2. Stakeholder – someone or something with vested interests in the behaviour of the system under discussion (SUD)
3. Primary Actor – stakeholder who initiates an interaction with the system to achieve a goal
4. Preconditions – what must be true or happen before and after the use case runs.
5. Triggers – this is the event that causes the use case to be initiated.
6. Main success scenarios– use case in which nothing goes wrong.
7. Alternative paths [Alternative Flow] – these paths are a variation on the main theme. These exceptions are what happen when things go wrong at the system level.

❖ SAMPLE USE CASE

Context: Laundry



❖ UML for use cases

1. Naming Use Cases

UML Specification provides no guidelines on use case names. The only requirement is that each use case must have a name. We should just follow use case definition to give some name to that unit of functionality performed by a system which provides some observable and useful result to an actor. Examples of use case names:

Make Purchase
Place Order
Update Subscription

2. Notation

Use case is usually shown as an ellipse containing the name of the use case.



If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note, that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier. Use cases have no standard keywords or stereotypes.

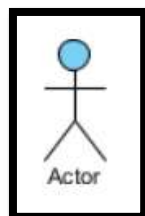
Note:

A business use case defines what happens in the business when the use case is requested by business actor, it describes complete workflow or business process that produces results required or in need of business actor. Business use case is represented in RUP with use case oval and a line crossing it as shown below.



Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc.

Note that actor represents a role that a user can play but not a specific user. So, in a hospital information system, you may have doctor and patient as actors but not Dr. John, Mrs. Brown as actors.



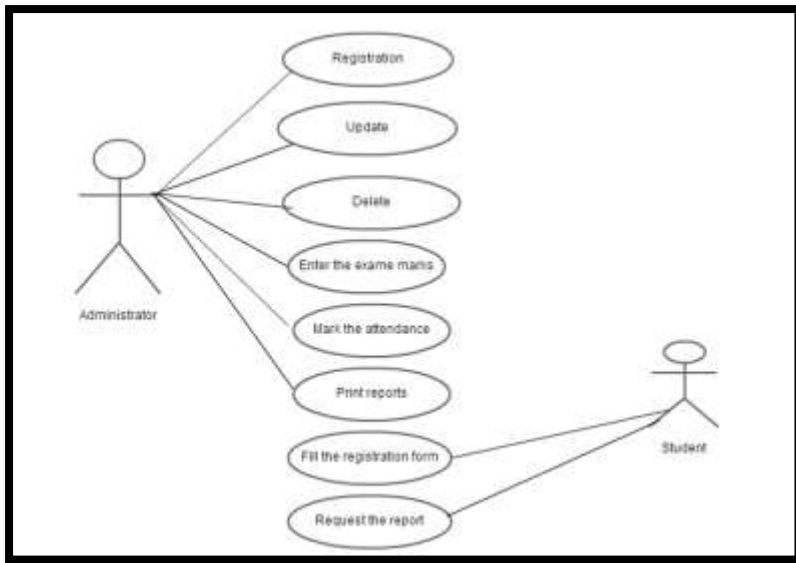
❖ SAMPLE USE CASE SCENARIO

Context: Change information of student

Use case name:	Change Student Information	UniqueID:	Student UC 005
Area:	Student System		
Actor(s):	Student		
Description:	Allow student to change his or her own information, such as name, home address, home telephone, campus address, campus telephone, cell phone, and other information using a secure Web site.		
Triggering Event:	Student uses Change Student Information Web site, enters student ID and password, and clicks the Submit button.		
Trigger type:	<input checked="" type="checkbox"/> External <input type="checkbox"/> Temporal		
Steps Performed (Main Path)	Information for Steps		
1. Student logs on to the secure Web server.	Student ID, Password		
2. Student record is read and password is verified.	Student Record, Student ID, Password		
3. Current student personal information is displayed on the Change Student Web page.	Student Record		
4. Student enters changes on the Change Student Web form and clicks Submit button.	Change Student Web Form		
5. Changes are validated on the Web server.	Change Student Web Form		
6. Change Student Journal record is written.	Change Student Web Form		
7. Student record is updated on the Student Master.	Change Student Web Form, Student Record		
8. Confirmation Web page is sent to the student.	Confirmation Page		
Preconditions:	Student is on the Change Student Information Web page.		
Postconditions:	Student has successfully changed personal information.		
Assumptions:	Student has a browser and a valid user ID and password.		
Requirements Met:	Allow students to be able to change personal information using a secure Web site.		
Outstanding Issues:	Should the number of times a student is allowed to logon be controlled?		
Priority:	Medium		
Risk:	Medium		

❖ SAMPLE USE CASE DIAGRAM

Context: Student DB system



- **EXPECTED RESULT/OUTCOME**

Write use cases for the chosen problem statement and then follow up with use case scenarios and use case diagram.

EXPERIMENT 3

- **AIM**

Identify the conceptual classes and develop a UML Class diagram

- **THEORY**

In UML, class diagrams are one of six types of structural diagram. Class diagrams are fundamental to the object modelling process and model the static structure of a system. Depending on the complexity of a system, a single class diagram may be used to model an entire system, or several class diagrams may also be used to model the components of a system.

Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help in the understanding of the requirements of the problem domain and to identify its components.

- ❖ UML class diagram: is a picture of

- the classes in an OO system
- their fields and methods
- connections between the classes (that interact or inherit from each other)

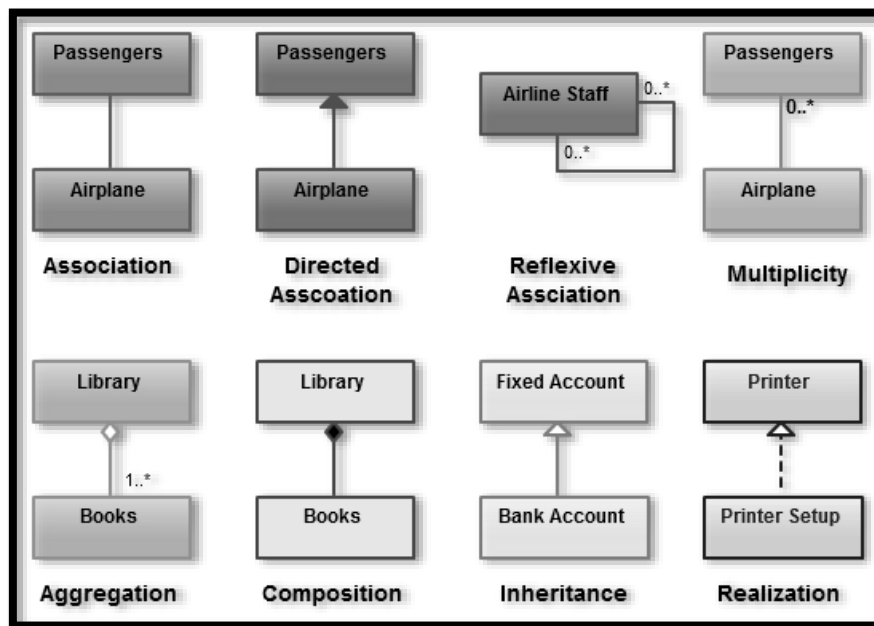
- ❖ Creating a class diagram

- Class name in top of box
 - write <<interface>> on top of interfaces' names
 - use italics for an abstract class name
- Attributes (optional)
 - should include all fields of the object
 - underline static attributes
- Operations / methods (optional)
 - may omit trivial (get/set) methods
 - underline static methods
- But don't omit any methods from an interface!
 - should not include inherited methods
- Comments
 - represented as a folded note, attached to the appropriate class/method/etc by a dashed line

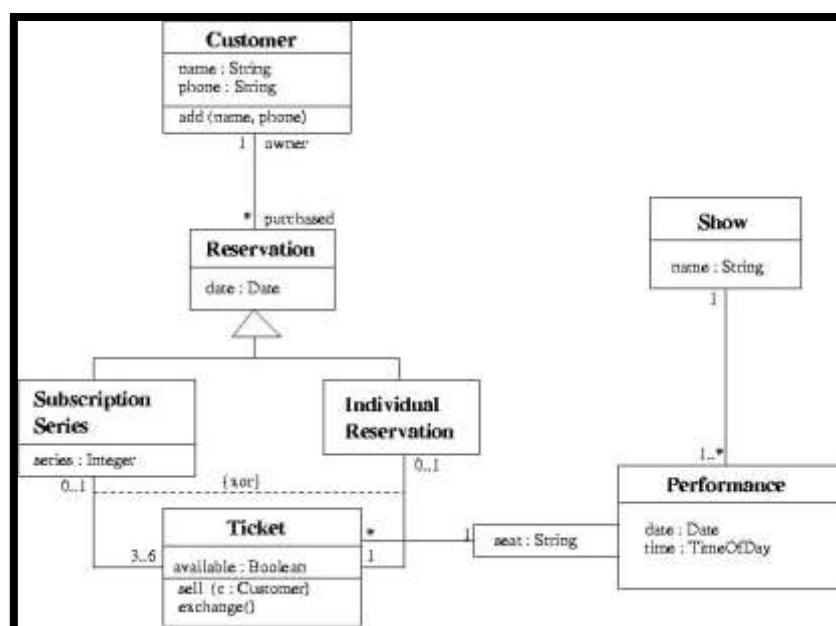
- ❖ RELATIONSHIPS BETWEEN CLASSES

- Generalization: an inheritance relationship

- inheritance between classes
 - interface implementation
 - Association: a usage relationship; An association relation is established when two classes are connected to each other in any way.
 - dependency
 - aggregation
 - composition
- ❖ DENOTING CLASS RELATIONSHIPS



❖ SAMPLE CLASS DIAGRAM



- **EXPECTED RESULT/OUTCOME**

Create conceptual classes for problem of your choice and draw the corresponding class diagrams.

EXPERIMENT 4

AIM

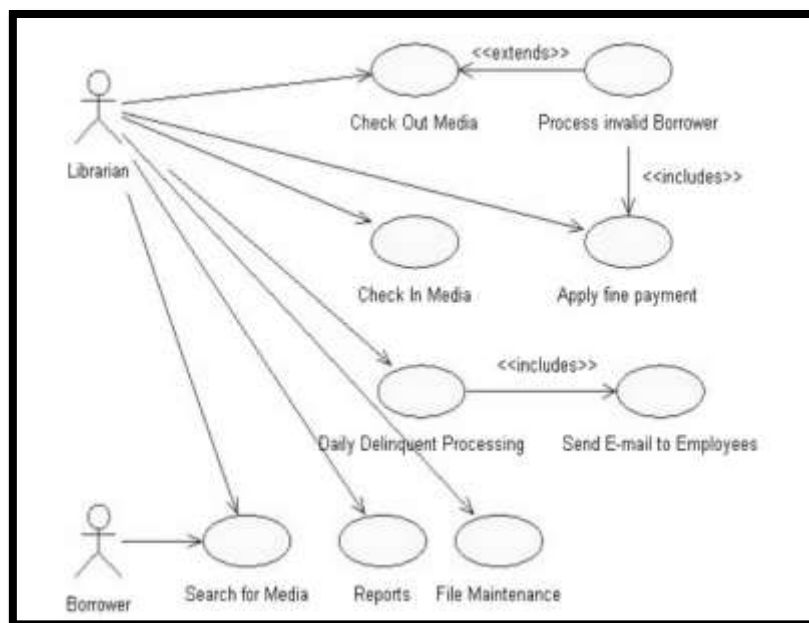
Using the identified scenarios find the interaction between objects and represent them using UML Interaction diagrams

THEORY

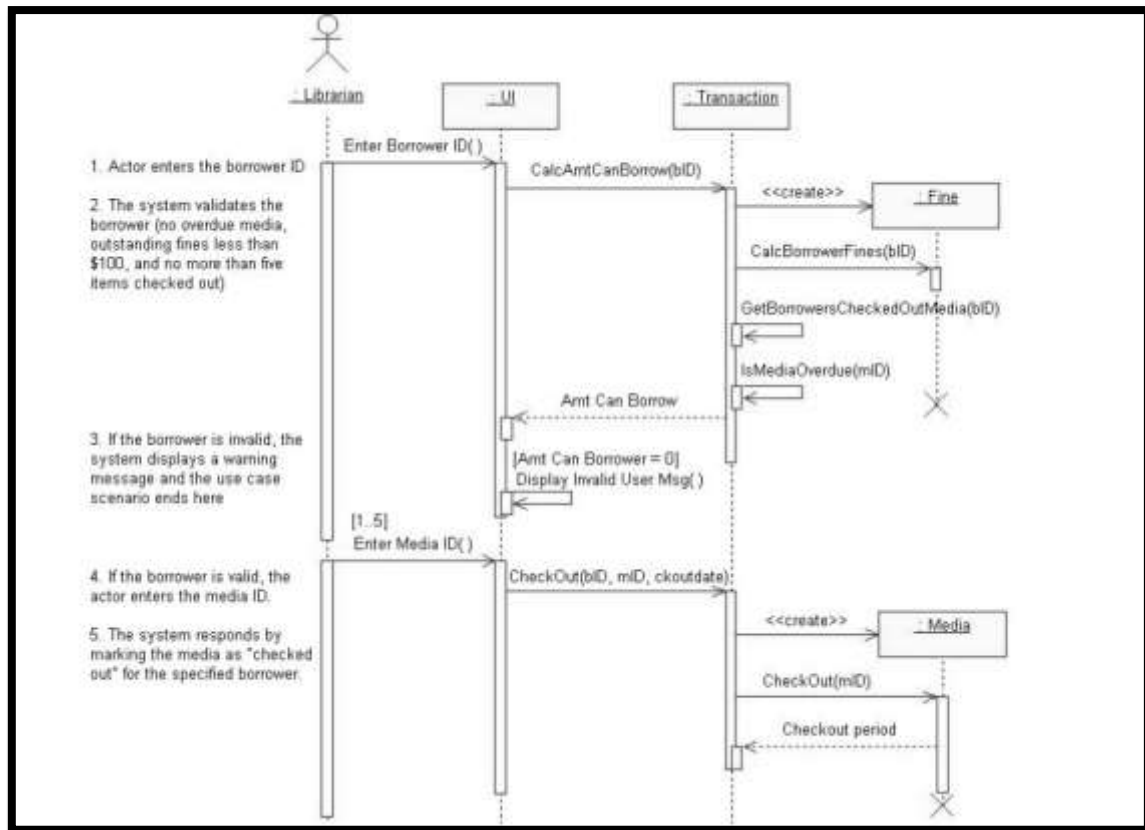
Interaction diagrams are models that describe how a group of objects collaborate in some behaviour - typically a single use-case. Interaction diagrams come in two forms, both present in the UML. The first form is the sequence diagram. The second form of the interaction diagram is the collaboration diagram.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

❖ SEQUENCE DIAGRAM



One of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams. In addition to their use in designing new systems, sequence diagrams can be used to document how objects in an existing (call it "legacy") system currently interact. This documentation is very useful when transitioning a system to another person or organization.

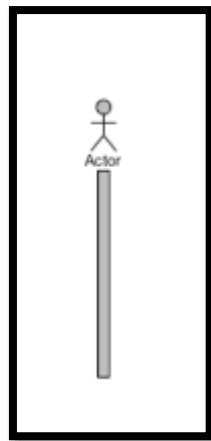


There are four primary elements of a sequence diagram:

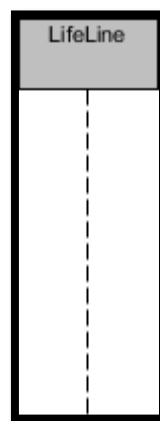
- Objects: Objects that are involved in the sequence of events should be placed at the top of the sequence diagram across its horizontal axis.
- Lifelines: The lifeline is the dotted line that extends down the vertical axis from the base of each object. The lifeline indicates the life span of an object over a period of time.
- Messages: Messages are the most important elements of a sequence diagram. They indicate when one object calls an operation on another object (or itself). They are also used to indicate return values.
- Focus of control: Focus of control (FOC) is used in sequence diagrams to show the period of time during which an object performs an action. FOC is rendered as a thin, rectangular object that sits on top of object lifelines. The top of the FOC rectangle coincides with the receipt of a message. The bottom of the rectangle coincides with the completion of an action and can be marked with a return message.

❖ NOTATION FOR SEQUENCE DIAGRAM

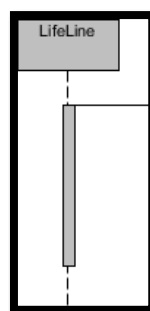
Actor: a type of role played by an entity that interacts with the subject by exchanging signals and data. Represent roles played by human users, external hardware, or other subjects.



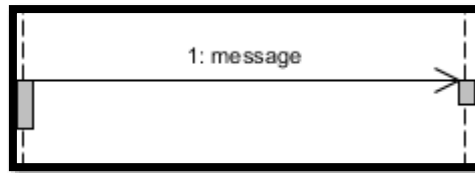
Lifeline: A lifeline represents an individual participant in the Interaction.



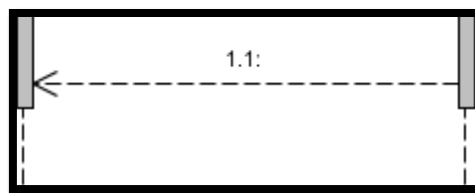
Activation: A thin rectangle on a lifeline) represents the period during which an element is performing an operation. The top and the bottom of the of the rectangle are aligned with the initiation and the completion time respectively.



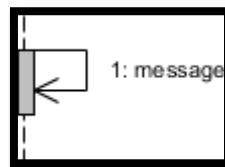
Call Message: A message defines a particular communication between Lifelines of an Interaction.



Return Message: Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message



Self-message: It is a kind of message that represents the invocation of message of the same lifeline.



❖ COLLABORATION DIAGRAM

Collaboration diagrams are often used to:

- Provide a birds-eye view of a collection of collaborating objects, particularly within a real-time environment.
- Allocate functionality to classes by exploring the behavioural aspects of a system.
- Model the logic of the implementation of a complex operation, particularly one that interacts with a large number of other objects.
- Explore the roles that objects take within a system, as well as the different relationships they are involved with when in those roles.

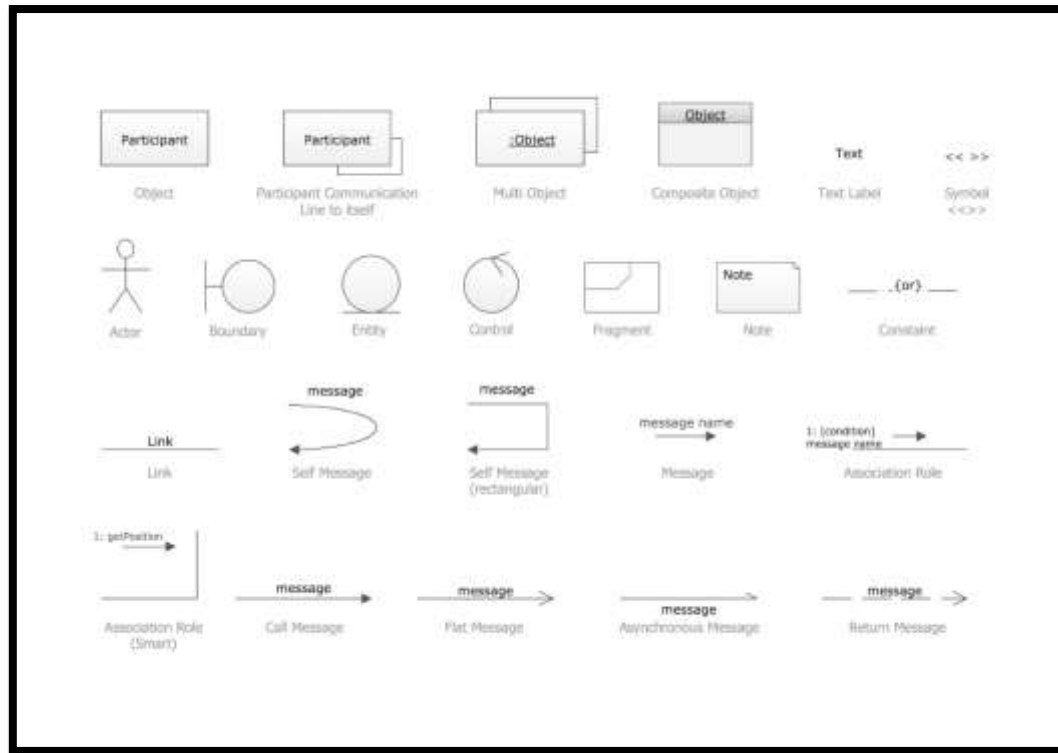
Often, business objects need to call on the services of other business objects to accomplish a particular task. Collaboration diagrams focus on the relationships between the collaborating objects.

There are three primary elements of a collaboration diagram:

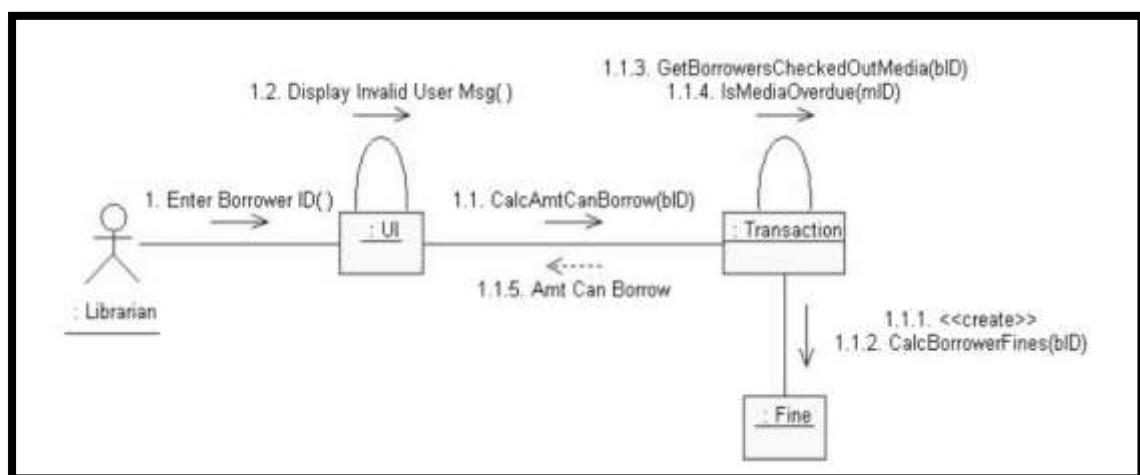
- Objects: Supplier objects are the objects that supply the method that is being called, and therefore receive the message. Client objects call methods on supplier objects, and therefore send messages.

- Links: The connecting lines drawn between objects in a collaboration diagram are links. A single link can support one or more messages sent between objects.
- Messages: arrows pointing from the Client object to the Supplier object. Typically, messages represent a client invoking an operation on a supplier object.

❖ NOTATION FOR COLLABORATION DIAGRAM



❖ SAMPLE COLLABORATION DIAGRAM



EXPECTED RESULT/OUTCOME

Analyse your chosen problem statement and create appropriate sequence, collaboration diagrams.

EXPERIMENT 5

AIM

To draw the diagrams[use case, activity, sequence, collaboration, class] for the Automated Trading House System.

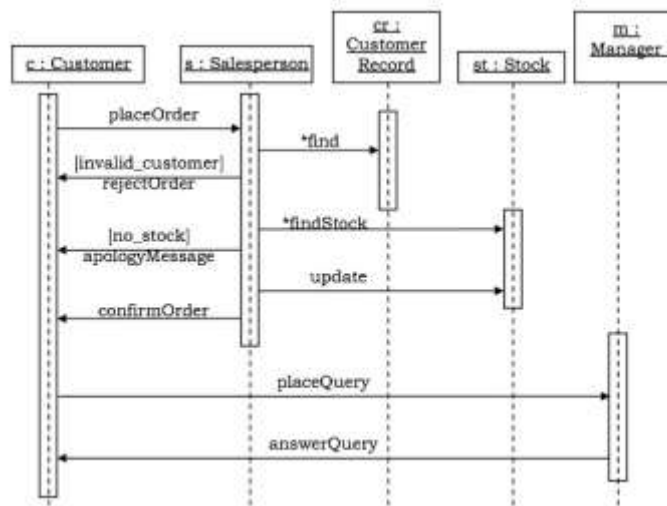
THEORY:

Sequence diagram:

Sequence diagrams are interaction diagrams that illustrate the ordering of messages according to time.

Notations – These diagrams are in the form of two-dimensional charts. The objects that initiate the interaction are placed on the x-axis. The messages that these objects send and receive are placed along the y-axis, in the order of increasing time from top to bottom.

Example – A sequence diagram for the Automated Trading House System is shown in the following figure.

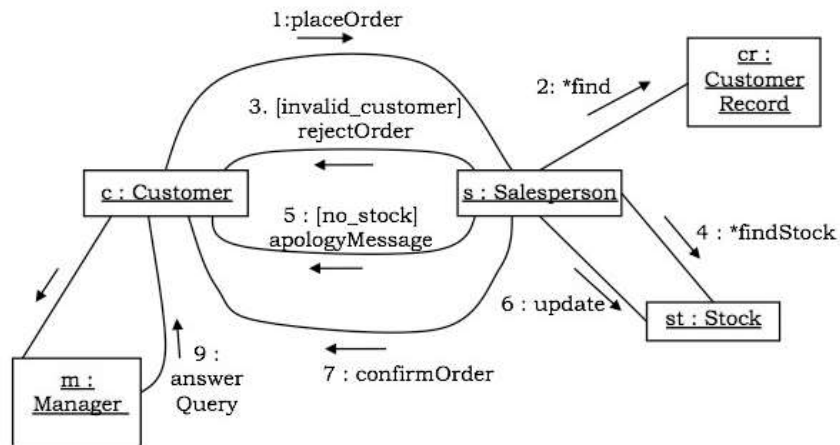


Collaboration Diagrams

Collaboration diagrams are interaction diagrams that illustrate the structure of the objects that send and receive messages.

Notations – In these diagrams, the objects that participate in the interaction are shown using vertices. The links that connect the objects are used to send and receive messages. The message is shown as a labeled arrow.

Example – Collaboration diagram for the Automated Trading House System is illustrated in the figure below.

**EXPECTED RESULT/OUTCOME:**

Analyse your chosen problem statement and create appropriate sequence, collaboration diagrams.

EXPERIMENT 6

AIM

Draw the State Chart diagram.

THEORY:

The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.

A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.

Activity diagram explained in the next chapter, is a special kind of a Statechart diagram. As Statechart diagram defines the states, it is used to model the lifetime of an object.

Purpose of Statechart Diagrams:

1. Statechart 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. Statechart diagrams are useful to model the reactive systems.
2. Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.
3. Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

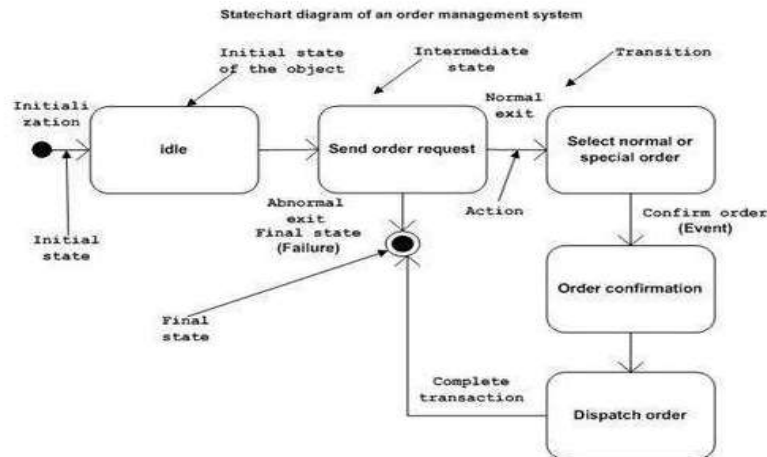
Following are the main purposes of using Statechart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

How to Draw a Statechart Diagram?

1. Statechart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.
2. Before drawing a Statechart diagram we should clarify the following points –
 - Identify the important objects to be analyzed.
 - Identify the states.
 - Identify the events.

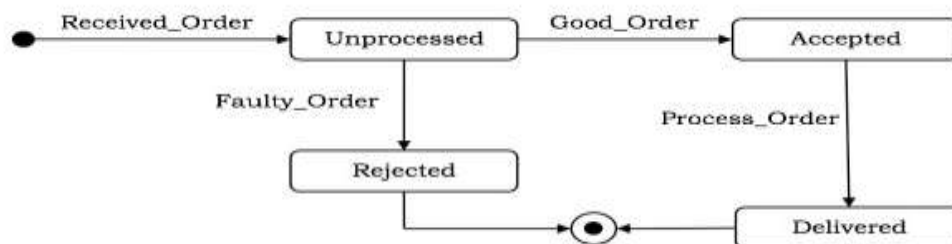
Following is an example of a Statechart diagram where the state of Order object is analyzed:



1. The first state is an idle state from where the process starts.
2. The next states are arrived for events like send request, confirm request, and dispatch order. These events are responsible for the state changes of order object.
3. During the life cycle of an object (here order object) it goes through the following states and there may be some abnormal exits.
4. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete, it is considered as a complete transaction as shown in the following figure.
5. The initial and final state of an object is also shown in the following figure.

Example:

In the Automated Trading House System, let us model Order as an object and trace its sequence. The following figure shows the corresponding state-chart diagram.



A state machine portrays the sequences of states which an object undergoes due to events and their responses to events.

State-Chart Diagrams comprise of –

- States: Simple or Composite
- Transitions between states
- Events causing transitions
- Actions due to the events

EXPECTED RESULT/OUTCOME: Analyse your chosen problem statement and created appropriate state chart diagram.

EXPERIMENT 7

AIM

Identify the business activities and develop an UML Activity diagram

THEORY

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.

Purpose of Activity Diagrams:

1. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques.
2. The only missing thing in the activity diagram is the message part. It does not show any message flow from one activity to another.
3. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –











- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

How to Draw an Activity Diagram?

- ✓ Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swimlane, etc.
- ✓ Before drawing an activity diagram, we should identify the following elements –
 - Activities
 - Association
 - Conditions
 - Constraints

Activity Diagram Symbols:

UML has specified a set of symbols and rules for drawing [activity diagrams](#). Following are the commonly used activity diagram symbols with explanations.

Symbol	Name	Use
	Start/ Initial Node	Used to represent the starting point or the initial state of an activity
	Activity / Action State	Used to represent the activities of the process
	Action	Used to represent the executable sub-areas of an activity
	Control Flow / Edge	Used to represent the flow of control from one action to the other
	Object Flow / Control Edge	Used to represent the path of objects moving through the activity
	Activity Final Node	Used to mark the end of all control flows within the activity
	Flow Final Node	Used to mark the end of a single control flow
	Decision Node	Used to represent a conditional branch point with one input and multiple outputs
	Merge Node	Used to represent the merging of flows. It has several inputs, but one output.
	Fork	Used to represent a flow that may branch into two or more parallel flows
	Merge	Used to represent two inputs that merge into one output
	Signal Sending	Used to represent the action of sending a signal to an accepting activity

	Signal Receipt	Used to represent that the signal is received
	Note/ Comment	Used to add relevant comments to elements

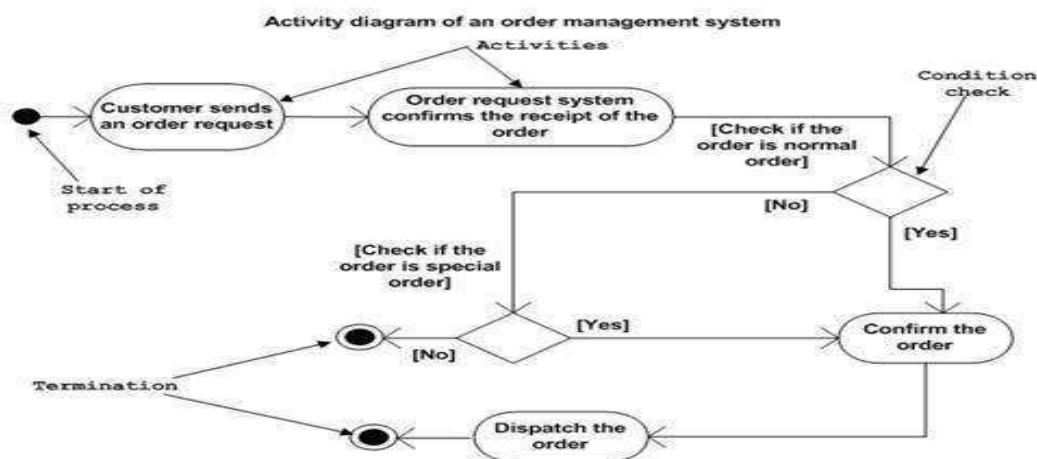
Following is an example of an activity diagram for order management system. In the diagram, four activities are identified which are associated with conditions.

The activity diagram is made to understand the flow of activities and is mainly used by the business users

Following diagram is drawn with the four main activities –

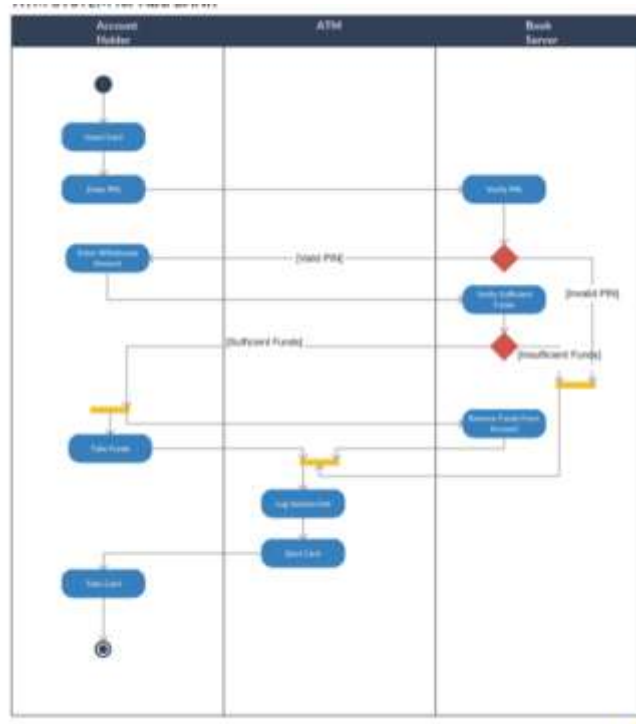
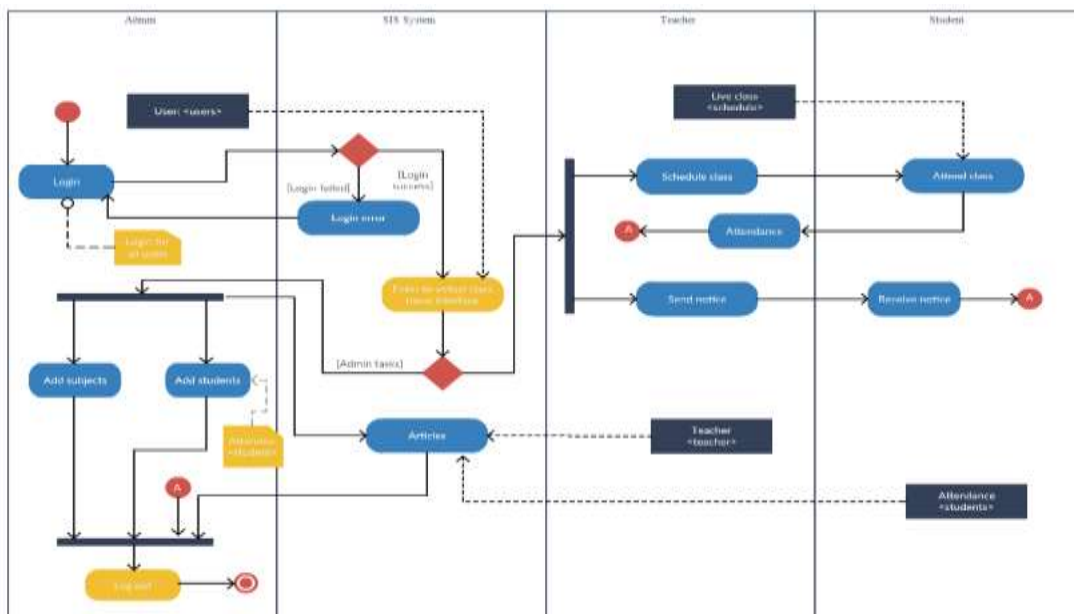
- Send order by the customer
- Receipt of the order
- Confirm the order
- Dispatch the order

After receiving the order request, condition checks are performed to check if it is normal or special order. After the type of order is identified, dispatch activity is performed and that is marked as the termination of the process.



EXAMPLE 1:

Activity Diagram for ATM:

**EXAMPLE 2 :****Activity Diagram for College Management System:****EXAMPLE 3:**

An activity diagram depicts the flow of activities which are ongoing non-atomic operations in a state machine. Activities result in actions which are atomic operations.

Activity diagrams comprise of –

- Activity states and action states
- Transitions

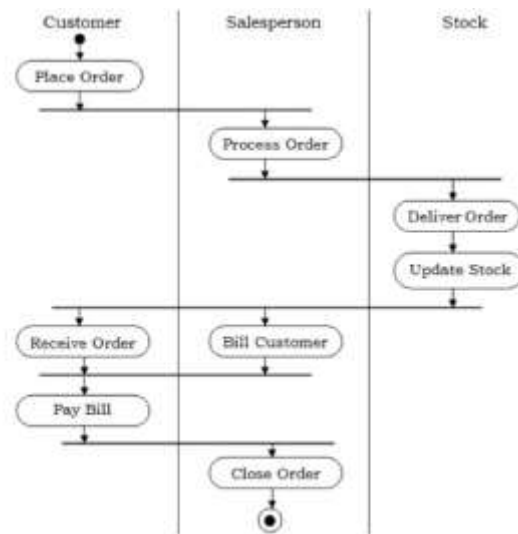
- Objects

Activity diagrams are used for modeling –

- workflows as viewed by actors, interacting with the system.
- details of operations or computations using flowcharts.

Example

The following figure shows an activity diagram of a portion of the Automated Trading House System.



EXPECTED RESULT/OUTCOME

Analyse your chosen problem statement and created appropriate Activity diagram.

EXPERIMENT 8

AIM:

Draw Component and Deployment diagrams

THEORY

Component diagrams:

1. Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system.
2. Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.
3. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

A single component diagram cannot represent the entire system but a collection of diagrams is used to represent the whole.

The purpose of the component diagram can be summarized as –

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

Component Diagram Symbols:

There are three ways the component symbol can be used.

1) Rectangle with the component stereotype (the text <<component>>). The component stereotype is usually used above the component name to avoid confusing the shape with a class icon.



2) Rectangle with the component icon in the top right corner and the name of the component.



3) Rectangle with the component icon and the component stereotype.

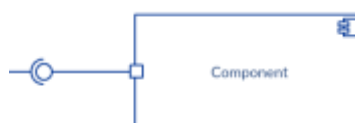


Provided Interface and the Required Interface



Interfaces in component diagrams show how components are wired together and interact with each other. The assembly connector allows linking the component's required interface (represented with a semi-circle and a solid line) with the provided interface (represented with a circle and solid line) of another component. This shows that one component is providing the service that the other is requiring.

Port



Port (represented by the small square at the end of a required interface or provided interface) is used when the component delegates the interfaces to an internal class.

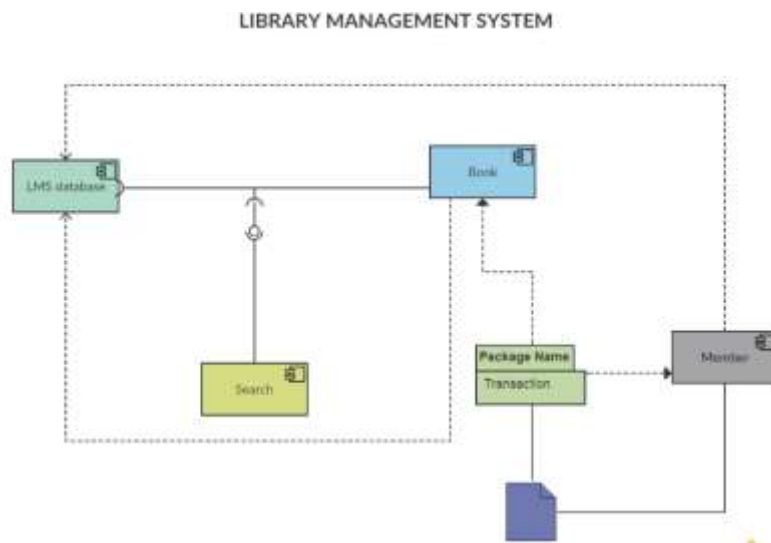
Dependencies



Although you can show more detail about the relationship between two components using the ball-and-socket notation (provided interface and required interface), you can just as well use a dependency arrow to show the relationship between two components.

EXAMPLES:

Component Diagram for Library Management System



Component Diagram for Online Shopping System



Deployment diagrams:

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.

Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important as it controls the following parameters –

- Performance
- Scalability
- Maintainability
- Portability

Before drawing a deployment diagram, the following artifacts should be identified –

- Nodes
- Relationships among nodes

Following is a sample deployment diagram to provide an idea of the deployment view of order management system. Here, we have shown nodes as –

- Monitor
- Modem
- Caching server
- Server

Deployment Diagram Notations:

In order to [draw a deployment diagram](#), you need to first become familiar with the following deployment diagram notations and deployment diagram elements.

Nodes



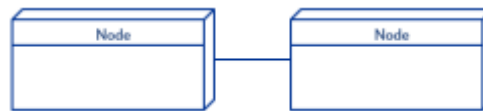
A node, represented as a cube, is a physical entity that executes one or more components, subsystems or executables. A node could be a hardware or software element.

Artifacts



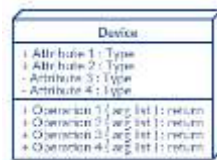
Artifacts are concrete elements that are caused by a development process. Examples of artifacts are libraries, archives, configuration files, executable files etc.

Communication Association



This is represented by a solid line between two nodes. It shows the path of communication between nodes.

Devices



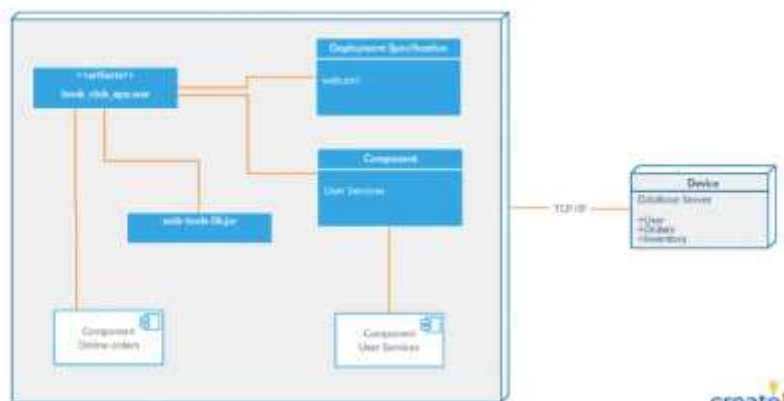
A device is a node that is used to represent a physical computational resource in a system. An example of a device is an application server.

Deployment Specifications

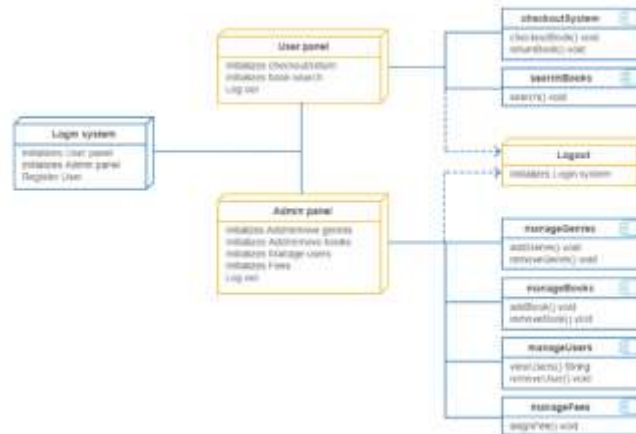


Deployment specifications is a configuration file, such as a text file or an XML document. It describes how an artifact is deployed on a node.

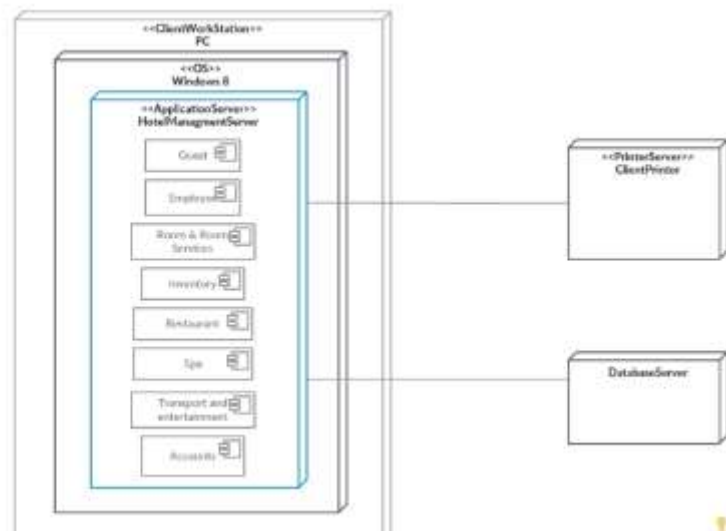
Deployment Diagram for Online Shopping System:



Deployment Diagram for Library Management System



Deployment Diagram for Hotel Management System



EXPECTED RESULT/OUTCOME

Analyse your chosen problem statement and created appropriate Component and Deployment diagrams.