Gokhale Education Society's

R. H. Sapat College of Engineering, Management Studies & Research, Nasik -422005

DEPARTMENT OF COMPUTER ENGINEERING

LAB MANUAL

FOR

B.E. COMPUTER (SEM – VII)

Academic Year 2023-2024

LABORATORY PRACTICE IV
SUBJECT CODE:
410244(D): OBJECT ORIENTED MODELING AND DESIGN

Teaching Scheme:	Credit: 01	Examination Scheme:
PR: 02 Hrs /Week		TW: 50 Marks

Course Outcomes:

CO1: Apply android application development for solving real life problems
CO2: Design and develop system using various multimedia components.
CO3: Identify various vulnerabilities and demonstrate using various tools.
CO4: Apply information retrieval tools for natural language processing
CO5: Develop an application using open source GPU programming languages
CO6: Apply software testing tools to perform automated testing

Hardware & Software Requirements: Windows/Ubuntu OS, https://staruml.io/, https://www.visual-paradigm.com/tutorials/addclassattrtofoe.jsp, https://www.smartdraw.com/uml-diagram/uml-diagram-tool.htm, https://www.modelio.org/.

INDEX

Assignments: Group 1
(Any 5 assignments from group 1)

Sr. No	Assignments to be covered
1.	Draw state model for telephone line, with various activities.
2.	Draw basic class diagrams to identify and describe key concepts like classes, types in your system and their relationships.
3.	Draw one or more Use Case diagrams for capturing and representing requirements of the system. Use case diagrams must include template showing description and steps of the Use Case for various scenarios.
4.	Draw activity diagrams to display either business flows or like flow charts
5.	Draw component diagrams assuming that you will build your system reusing existing components along with a few new ones
6.	Draw deployment diagrams to model the runtime architecture of your system.

Assignments: Group 2
(1 Mini project from group 2 is mandatory)

Sr. No	Assignments to be covered
1.	Mini Project: Draw all UML diagrams for **your** project work.
2.	Mini Project - Develop a Blockchain based application for health related medical records Draw following UML Diagrams for Bank Management application a. Class Diagram b. Object Diagram c. ER Diagram d. Component Diagram

Prepared By
Mr. R. B. Mandlik

Approved By
Dr. D. V. Patil
HOD

Group 1

Assignment No- 01 **Date:__/__/_____**

Title: - . Draw state model for telephone line with various activities.

State Diagram are used to capture the behavior of a software system. UML State machine diagrams can be used to model the behavior of a class, a subsystem, a package, or even an entire system. It is also called a State chart or State Transition diagram.

There are a total of two types of state machine diagram in UML:

1. Behavioral State Machine Diagram

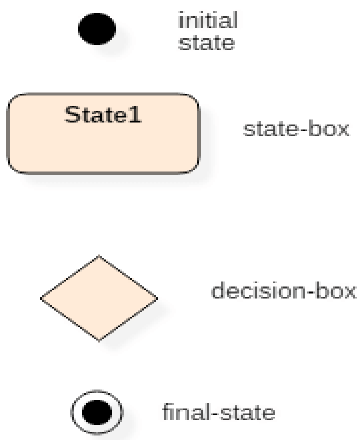
- It captures the behavior of an entity present in the system.
- It is used to represent the specific implementation of an element.
- The behavior of a system can be modelled using behavioral state machine diagram in OOAD.

2. Protocol State Machine Diagram

- These diagrams are used to capture the behavior of a protocol.
- It represents how the state of protocol changes concerning the event. It also represents corresponding changes in the system.
- They do not represent the specific implementation of an element.

Notation and Symbol for State Machine Diagram (Statechart Diagram)

Following are the various notations that are used throughout the state chart diagram. All these notations, when combined, make up a single diagram.



UML state diagram notations

Initial state

The initial state symbol is used to indicate the beginning of a state machine diagram.

Final state

This symbol is used to indicate the end of a state machine diagram.

Decision box

It contains a condition. Depending upon the result of an evaluated guard condition, a new path is taken for program execution.

Transition

A transition is a change in one state into another state which is occurred because of some event. A transition causes a change in the state of an object.

State box: It is a specific moment in the lifespan of an object. It is defined using some condition or a statement within the classifier body. It is used to represent any static as well as dynamic situations. The name of a state is written inside the rounded rectangle.

The name of a state can also be placed outside the rectangle. This can be done in case of composite or submachine states. One can either place the name of a state within the rectangle or outside the rectangle in a tabular box. One cannot perform both at the same time.

A state can be either active or inactive. When a state is in the working mode, it is active, as soon as it stops executing and transits into another state, the previous state becomes inactive, and the current state becomes active.

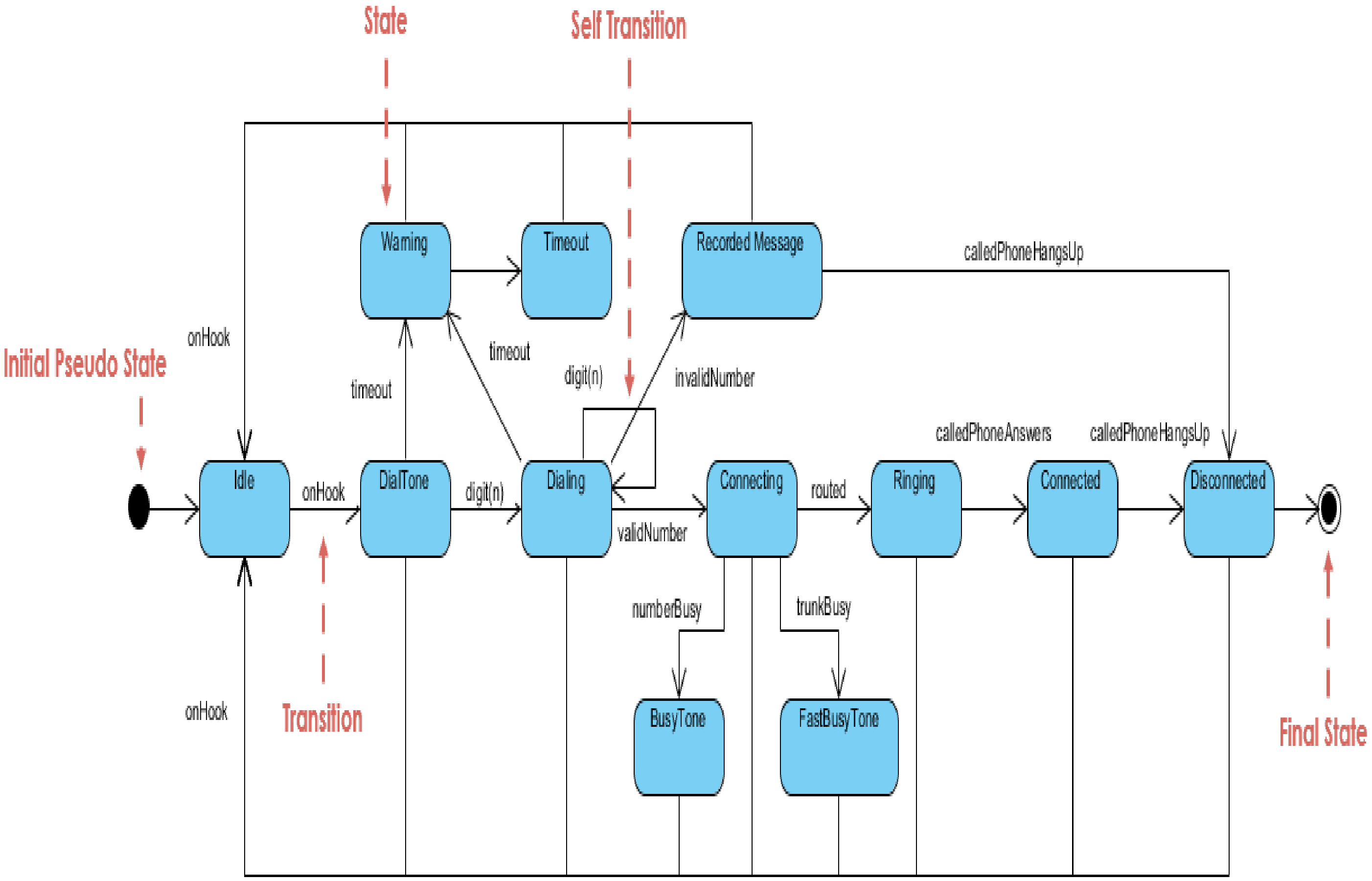
Types of State

Unified Modeling Language defines three types of states:

- **Simple state**
  They do not have any substrate.
- **Composite state**
  These types of states can have one or more than one substrate. A composite state with two or more substates is called an orthogonal state.
- **Submachine state**
  These states are semantically equal to the composite states. Unlike the composite state, we can reuse the submachine states.

Consider the class for telephone line with following activities and states:

As a start of a call, the telephone line is idle. When the phone receiver is picked from hook, it gives a dial tone and can accept the dialing of digits. If after getting dial tone, if the user doesn't dial number within time interval then time out occurs and phone line gets idle. After dialing a number, if the number is invalid then some recorded message is played. Upon entry of a valid number, the phone system tries to connect a call & routes it to proper destination. If the called person answers the phone, the conversation can occur. When called person hangs up, the phone disconnects and goes to idle state.



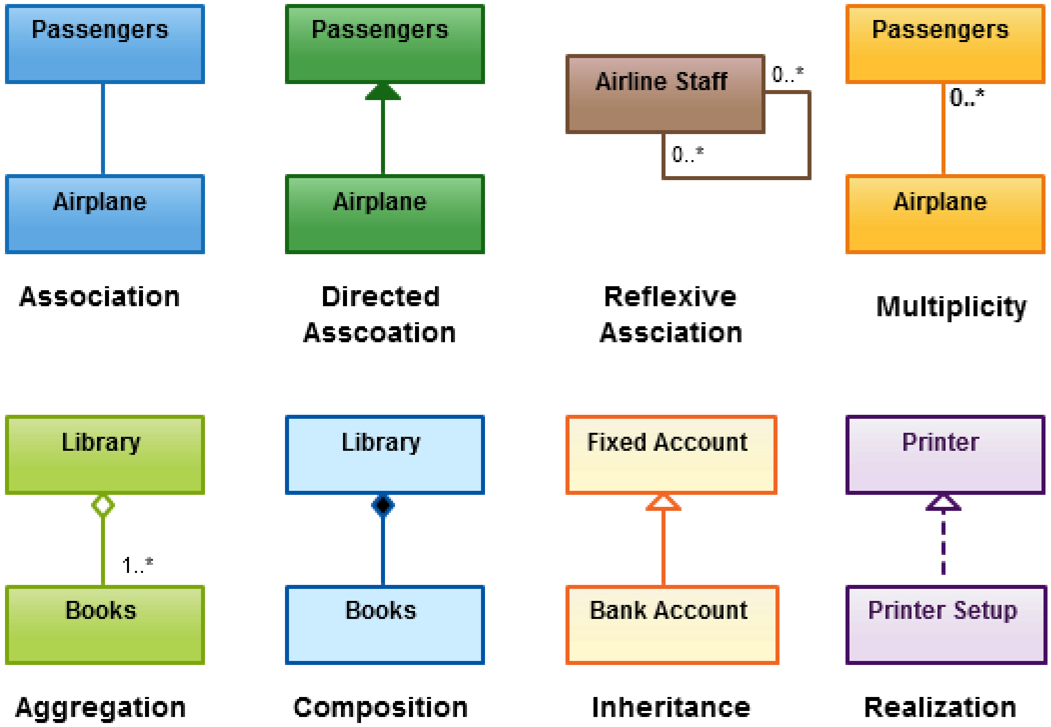
Outcomes:-

Assignment No- 02 **Date:__/__/_____**

Title: -Draw basic class diagrams to identify and describe key concepts like classes, types in your system and their relationships.

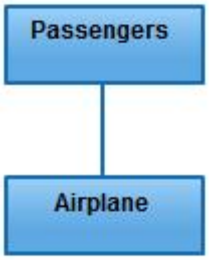
Classes are interrelated to each other in specific ways. In particular, relationships in class diagrams include different types of logical connections. The following are such types of logical connections that are possible in UML:

- Association
- Directed Association
- Reflexive Association
- Multiplicity
- Aggregation
- Composition
- Inheritance/Generalization
- Realization



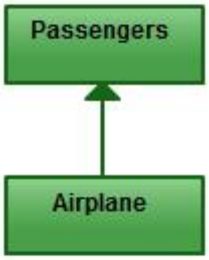
Relationships in UML class diagrams

Association



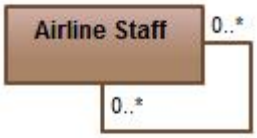
Association is a broad term that encompasses just about any logical connection or relationship between classes. For example, passengers and airline may be linked as above.

Directed Association



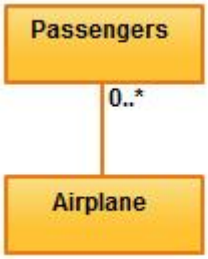
Directed Association refers to a directional relationship represented by a line with an arrowhead. The arrowhead depicts a container-contained directional flow.

Reflexive Association



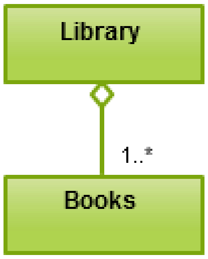
Reflexive Association:This occurs when a class may have multiple functions or responsibilities. For example, a staff member working in an airport may be a pilot, aviation engineer, ticket dispatcher, guard, or maintenance crew member. If the maintenance crew member is managed by the aviation engineer there could be a managed by relationship in two instances of the same class.

Multiplicity



Multiplicity is the active logical association when the cardinality of a class in relation to another is being depicted. For example, one fleet may include multiple airplanes, while one commercial airplane may contain zero to many passengers. The notation 0..* in the diagram means “zero to many”.

Aggregation

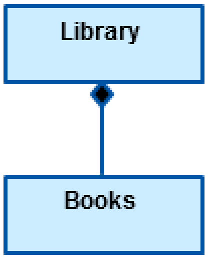


Aggregation refers to the formation of a particular class as a result of one class being aggregated or built as a collection. For example, the class “library” is made up of one or more books, among other materials. In aggregation, the contained classes are not strongly dependent on the lifecycle of the container.

In the same example, books will remain so even when the library is dissolved. To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.

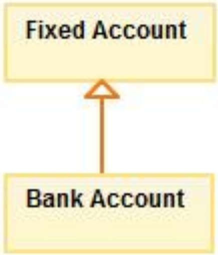
To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.

Composition



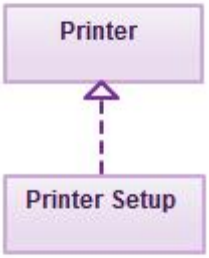
Composition: The composition relationship is very similar to the aggregation relationship. with the only difference being its key purpose of emphasizing the dependence of the contained class to the life cycle of the container class. That is, the contained class will be obliterated when the container class is destroyed. For example, a shoulder bag's side pocket will also cease to exist once the shoulder bag is destroyed.
To show a composition relationship in a UML diagram, use a directional line connecting the two classes, with a filled diamond shape adjacent to the container class and the directional arrow to the contained class.

Inheritance / Generalization



Inheritance refers to a type of relationship wherein one associated class is a child of another by virtue of assuming the same functionalities of the parent class. In other words, the child class is a specific type of the parent class. To show inheritance in a UML diagram, a solid line from the child class to the parent class is drawn using an unfilled arrowhead.

Realization

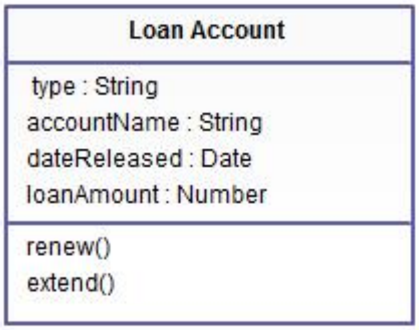


Realization denotes the implementation of the functionality defined in one class by another class. To show the relationship in UML, a broken line with an unfilled solid arrowhead is drawn from the class that defines the functionality of the class that implements the function. In the example, the printing preferences that are set using the printer setup interface are being implemented by the printer.

Class diagrams:

are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations, and the relationships among them.

The following figure is an example of a simple class:



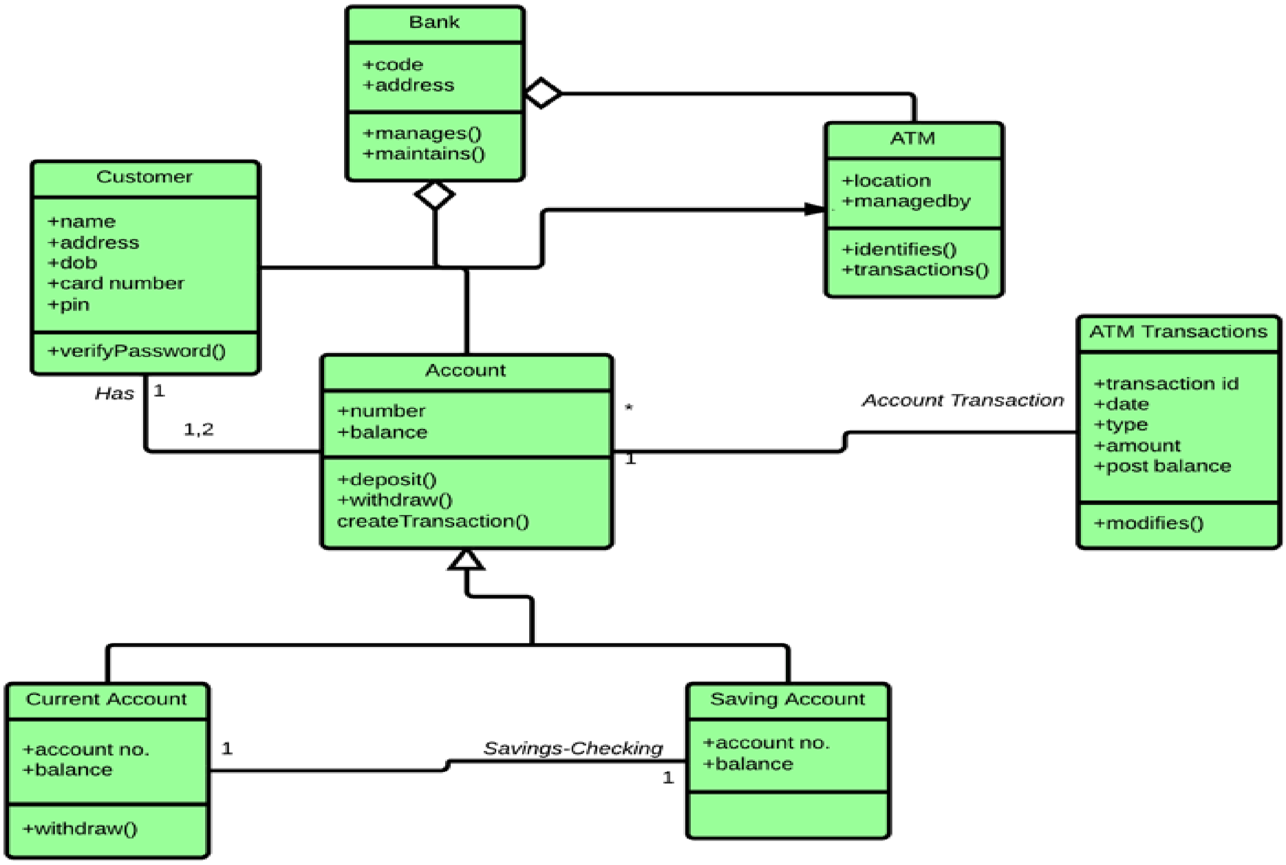
Simple class diagram with attributes and operations

In the example, a class called “loan account” is depicted. Classes in class diagrams are represented by boxes that are partitioned into three:

1. The top partition contains the name of the class.
2. The middle part contains the class’s attributes.
3. The bottom partition shows the possible operations that are associated with the class.

The example shows how a class can encapsulate all the relevant data of a particular object in a very systematic and clear way. A class diagram is a collection of classes similar to the one above.

UML Class Diagram example:



Outcomes:-

Assignment No- 03 **Date:__/__/_____**

Title: - Draw one or more Use Case diagrams for capturing and representing requirements of the system. Use case diagrams must include template showing description and steps of the Use Case for various scenarios.

A use case describes how a user uses a system to accomplish a particular goal. A use case diagram consists of the system, the related use cases and actors and relates these to each other to visualize: what is being described? (**system**), who is using the system? (**actors**) and what do the actors want to achieve? (**use cases**), thus, use cases help ensure that the correct system is developed by capturing the requirements from the user's point of view.

A use case is a list of actions or event steps typically defining the interactions between a role of an actor and a system to achieve a goal. A use case is a useful technique for identifying, clarifying, and organizing system requirements. A use case is made up of a set of possible sequences of interactions between systems and users that defines the features to be implemented and the resolution of any errors that may be encountered.

A use case (or set of use cases) has these characteristics:

1. Organizes functional requirements
2. Models the goals of system/actor (user) interactions
3. Describes one main flow of events (main scenarios) and possibly other exceptional flows (alternatives), also called paths or user scenarios

Use Case Diagram Notations

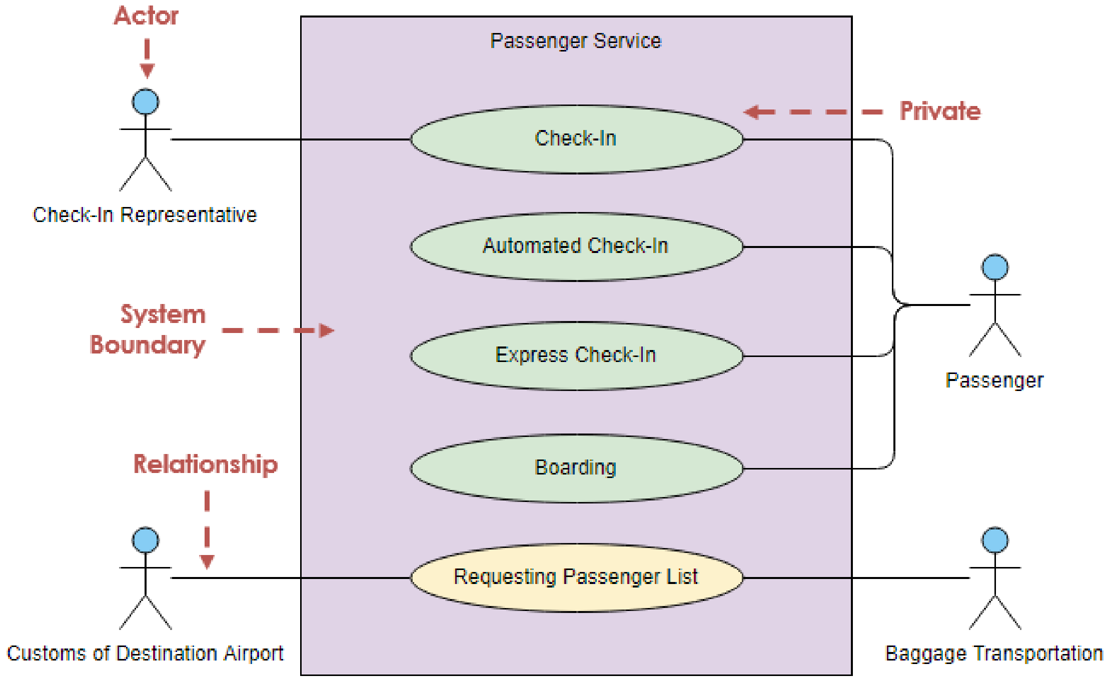
Use cases define interactions between external actors and the system to attain particular goals. A use case diagram contains four main components

Actor

Actors are usually individuals involved with the system defined according to their roles. The actor can be a human or other external system.

Use Case

A use case describes how an actor uses a system to accomplish a particular goal. Use cases are typically initiated by a user to fulfill goals describing the activities and variants involved in attaining the goal.



Relationship

The relationships between and among the actors and the use cases.

System Boundary

The system boundary defines the system of interest in relation to the world around it.

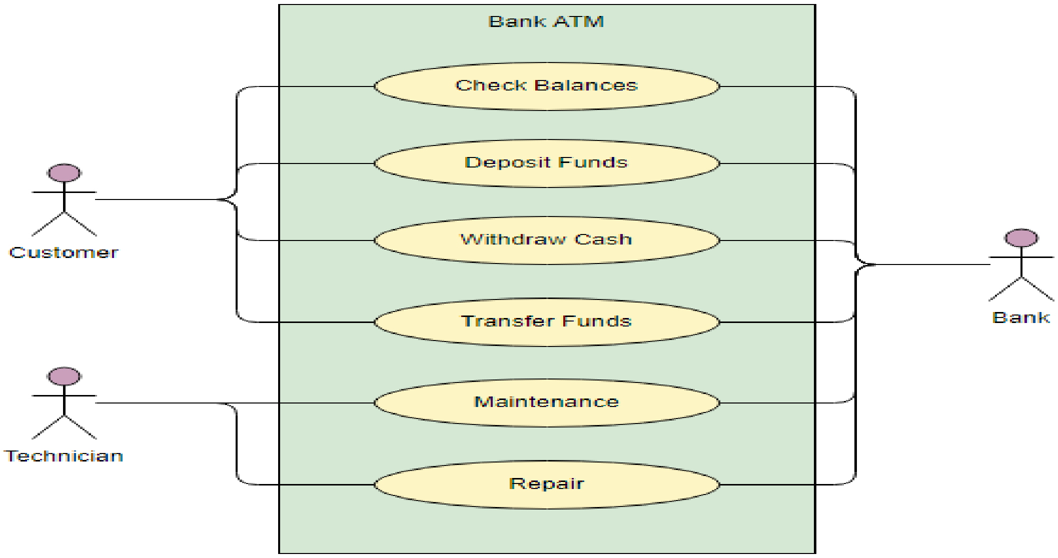
Benefits of Use Case Diagram

1. Use cases are a powerful technique for the elicitation and documentation of black-box functional requirements.
2. Because, use cases are easy to understand and provide an excellent way for communicating with customers and users as they are written in natural language.
3. Use cases can help manage the complexity of large projects by partitioning the problem into major user features (i.e., use cases) and by specifying applications from the users' perspective.
4. A use case scenario, often represented by a sequence diagram, involves the collaboration of multiple objects and classes, use cases help identify the messages (operations and the information or data required - parameters) that glue the objects and classes together.
5. Use cases provide a good basis to link between the verification of the higher-level models (i.e. interaction between actors and a set of collaborative objects), and subsequently, for the validation of the functional requirements (i.e. blueprint of white-box test).
6. Use case driven approach provides an traceable links for project tracking in which the key development activities such as the use cases implemented, tested, and delivered fulfilling the goals and objectives from the user point of views.

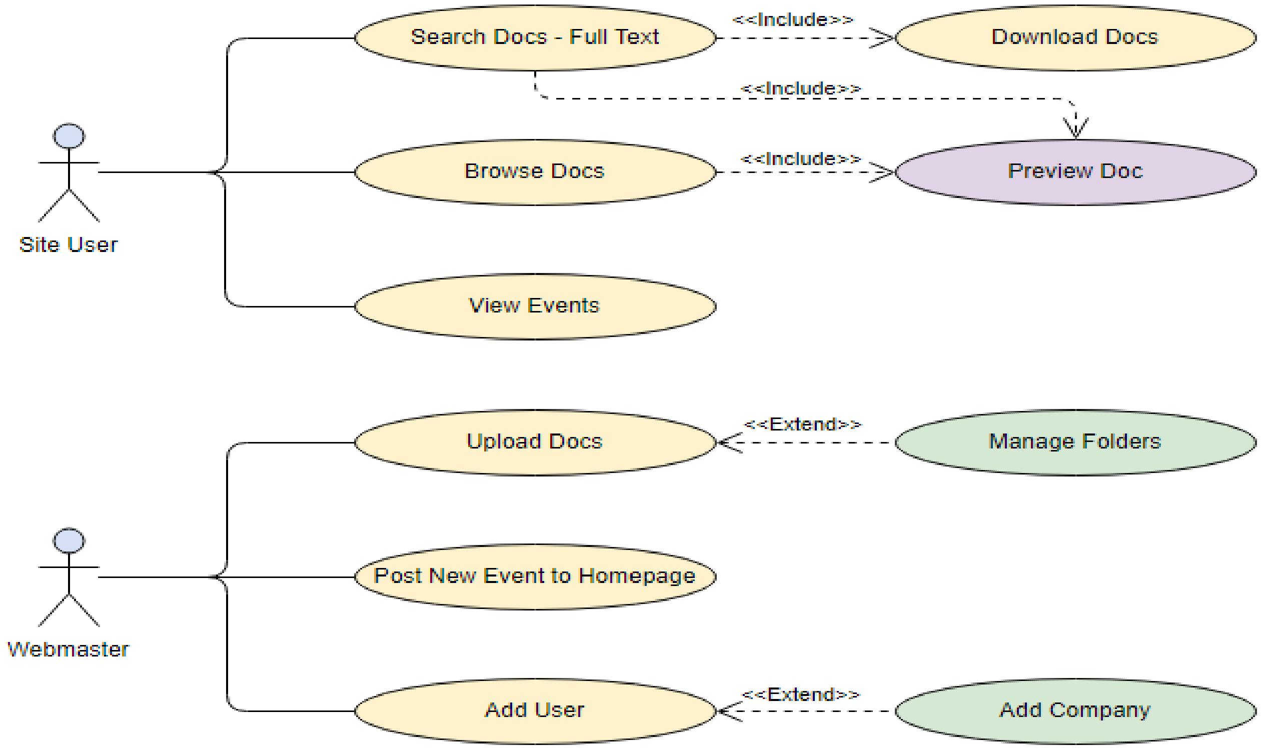
How to Draw a Use Case Diagram?

1. Identify the Actors (role of users) of the system.
2. For each category of users, identify all roles played by the users relevant to the system.
3. Identify what are the users required the system to be performed to achieve these goals.
4. Create use cases for every goal.
5. Structure the use cases.
6. Prioritize, review, estimate and validate the users.

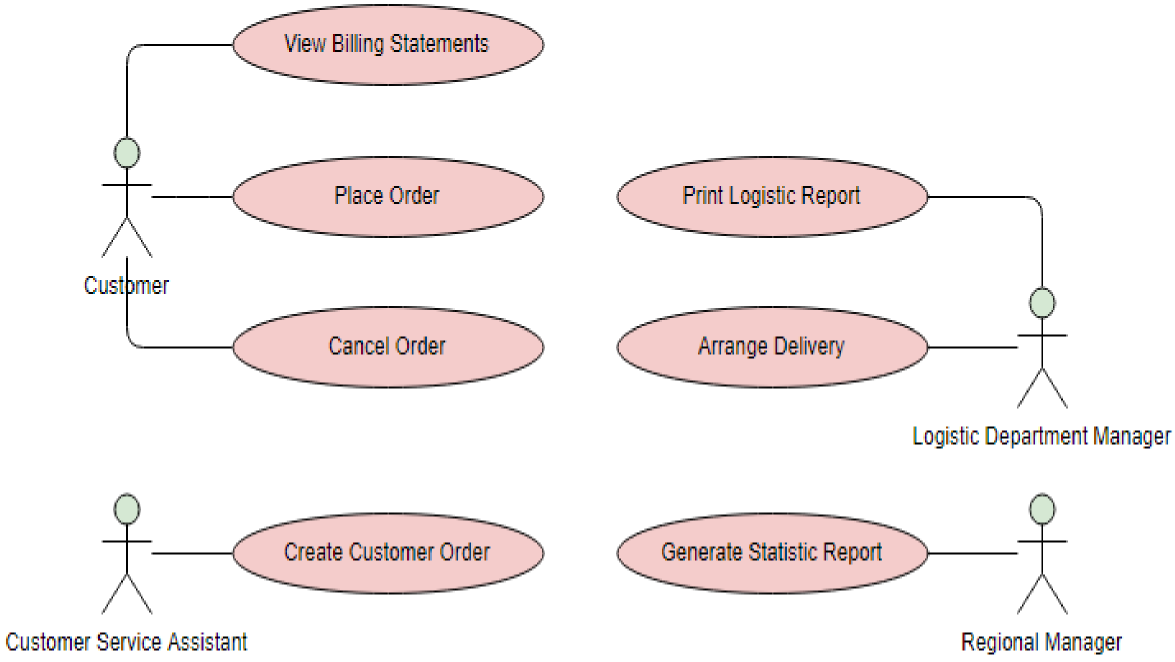
The figure below shows an ATM use case diagram example, which is quite a classic example to use in teaching use case diagram.



The **Document Management System (DMS)** use case diagram example below shows the actors and use cases of the system. In particular, there are include and extend relationships among use cases.



The **Order System** use case diagram example below shows the actors and use cases involved in the system:

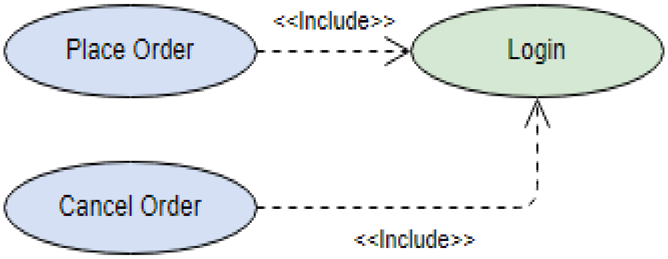


Structuring Use Cases

UML defines three stereotypes of association between Use Cases:

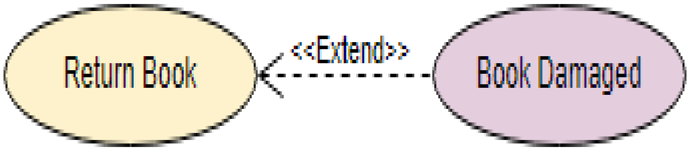
<<include>> Use Case

The time to use the <<include>> relationship is after you have completed the first cut description of all your main Use Cases. You can now look at the Use Cases and identify common sequences of user-system interaction.



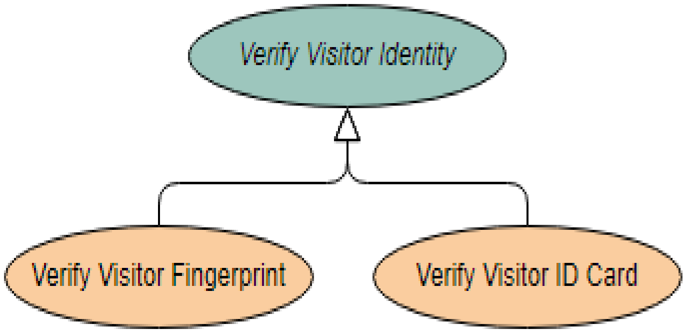
<<extend>> Use Case

An extending use case is, effectively, an alternate course of the base use case. The <<extend>> use case accomplishes this by conceptually inserting additional action sequences into the base use-case sequence.



Abstract and generalized Use Case

The general use case is abstract. It cannot be instantiated, as it contains incomplete information. The title of an abstract use case is shown in italics.



Outcomes:-

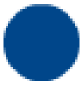
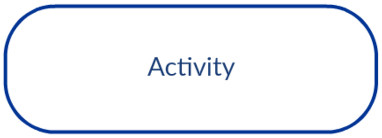
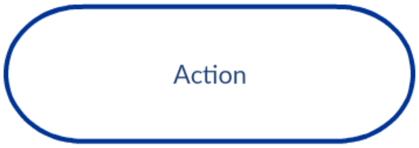
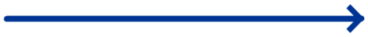
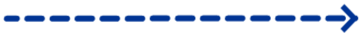
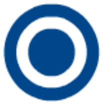
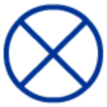
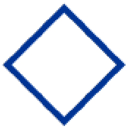
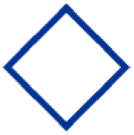
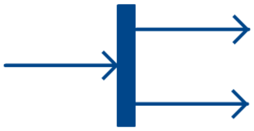
Assignment No- 04 **Date:__/__/_____**

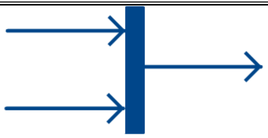
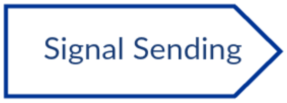
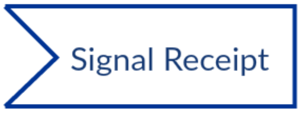
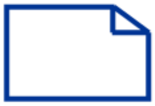
Title: -Draw activity diagrams to display either business flows or like flowcharts

A UML activity diagram helps to visualize a certain use case at a more detailed level. It is a behavioral diagram that illustrates the flow of activities through a system.UML activity diagrams can also be used to depict a flow of events in a business process. They can be used to examine business processes in order to identify their flow and requirements.

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

How to Draw an Activity Diagram

Activity diagrams can be used to model business requirements, create a high-level view of a system's functionalities, analyze use cases, and for various other purposes. In each of these cases, here's how to draw an activity diagram from the beginning.

Step 1: Figure out the action steps from the use case

Here you need to identify the various activities and actions your business process or system is made up of.

Step 2: Identify the actors who are involved

If you already have figured out who the actors are, then it's easier to discern each action they are responsible for.

Step 3: Find a flow among the activities

Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others?

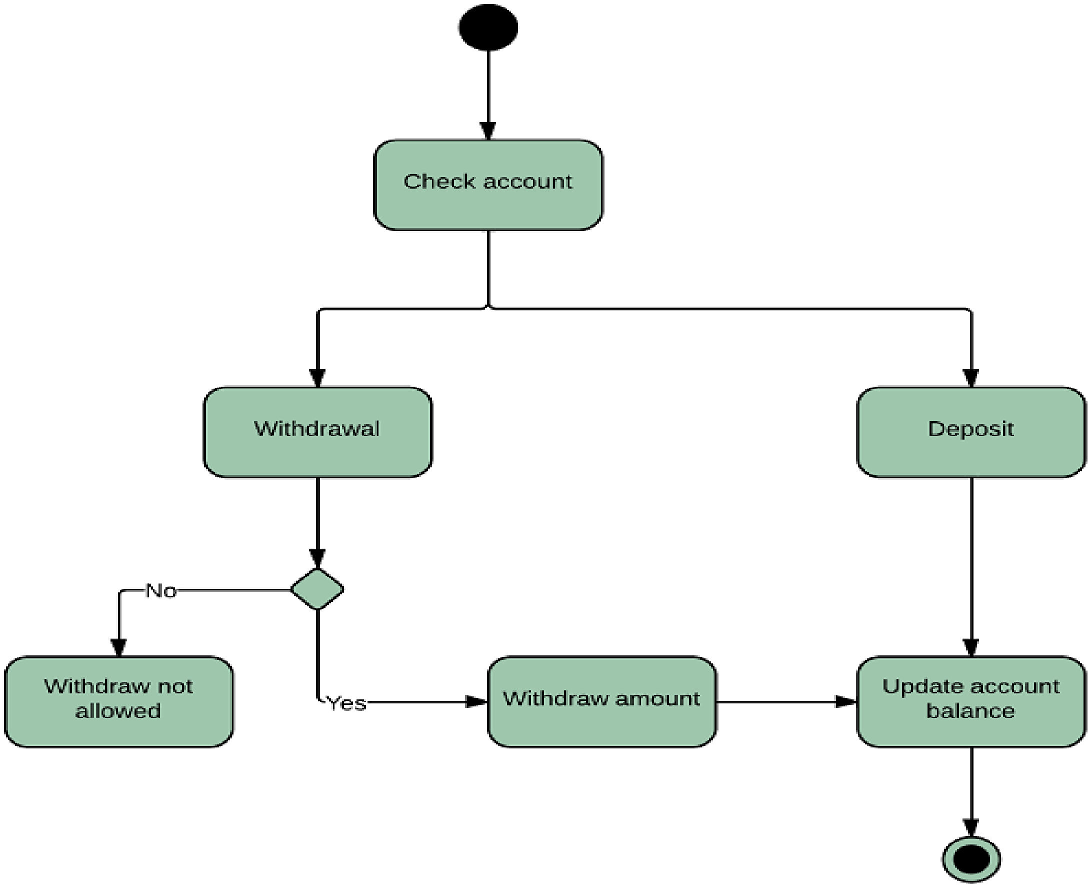
Step 4: Add swimlanes

You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.

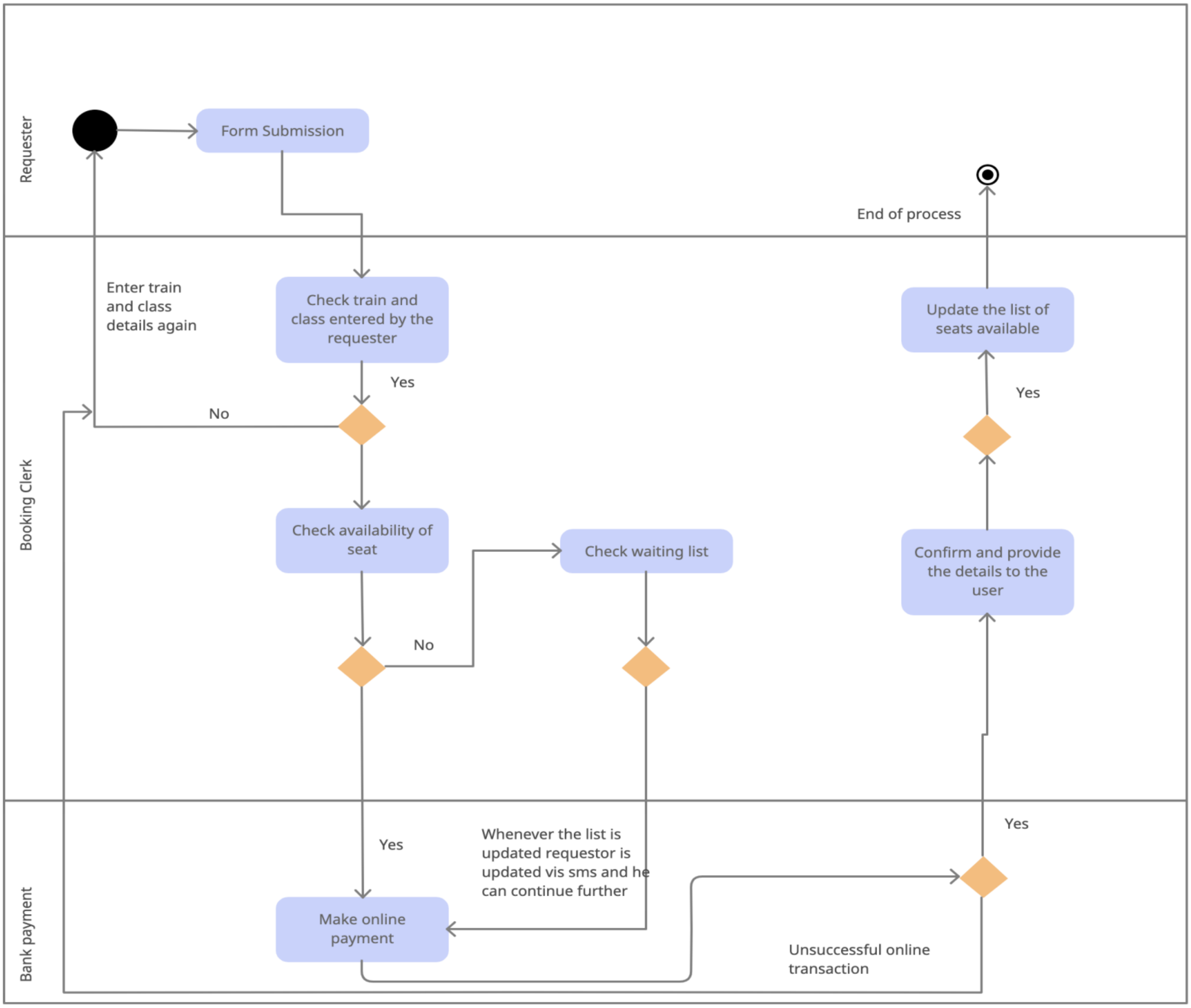
Activity Diagram Examples:

1. Activity diagram for a banking system

This diagram shows the process of either withdrawing money from or depositing money into a bank account. An advantage of representing the workflow visually in UML is the ability to show withdrawals and deposits on one chart.



2. Activity Diagram for a Railway Reservation System


Requester
Booking Clerk
Bank payment
Form Submission
End of process
Enter train and class details again
Check train and class entered by the requester
Yes
No
Update the list of seats available
Yes
Check availability of seat
Check waiting list
Confirm and provide the details to the user
No
Yes
Whenever the list is updated requestor is updated vis sms and he can continue further
Yes
Make online payment
Unsuccessful online transaction

Assignment No- 05 **Date:__/__/_____**

Title: - Draw component diagrams assuming that you will build your system reusing existing components along with a few new ones.

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system.

The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.

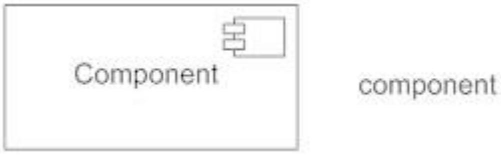
Component diagrams

- Are used in Component-Based-Development to describe systems with Service-Oriented-Architecture
- Show the structure of the code itself
- Can be used to focus on the relationship between components while hiding specification detail
- Help communicate and explain the functions of the system being built to stakeholders

Component Diagram Symbols:

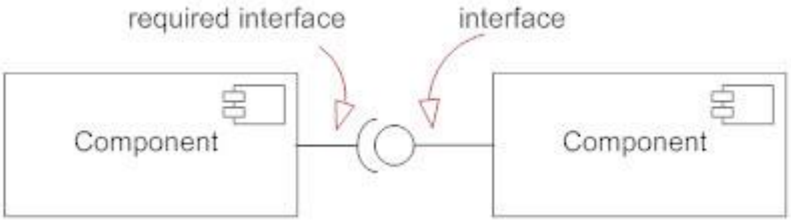
Component

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



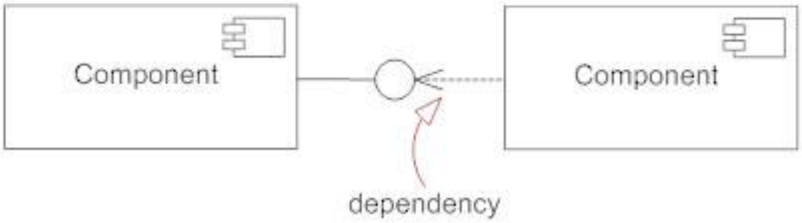
Interface

An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.



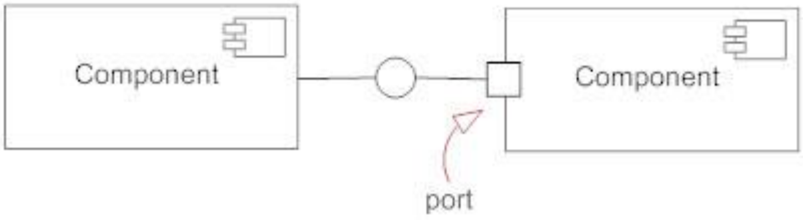
Dependencies

Draw dependencies among components using dashed arrows.



Port

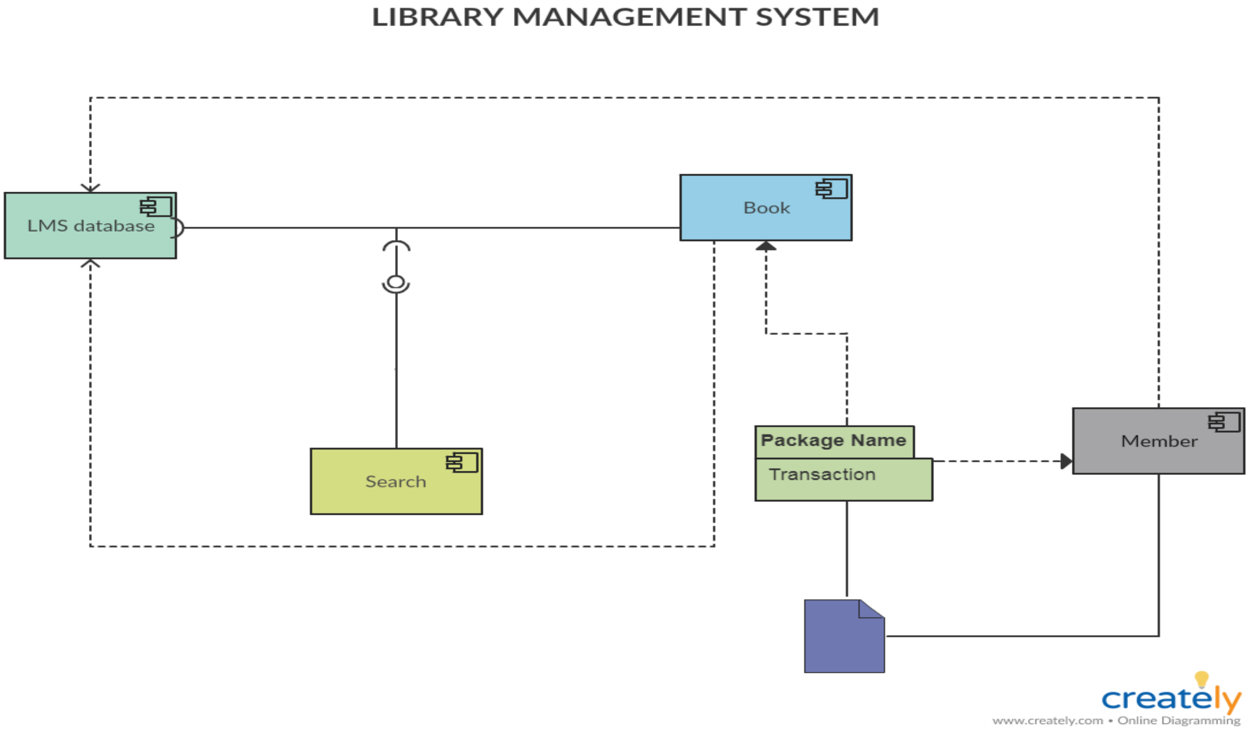
Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



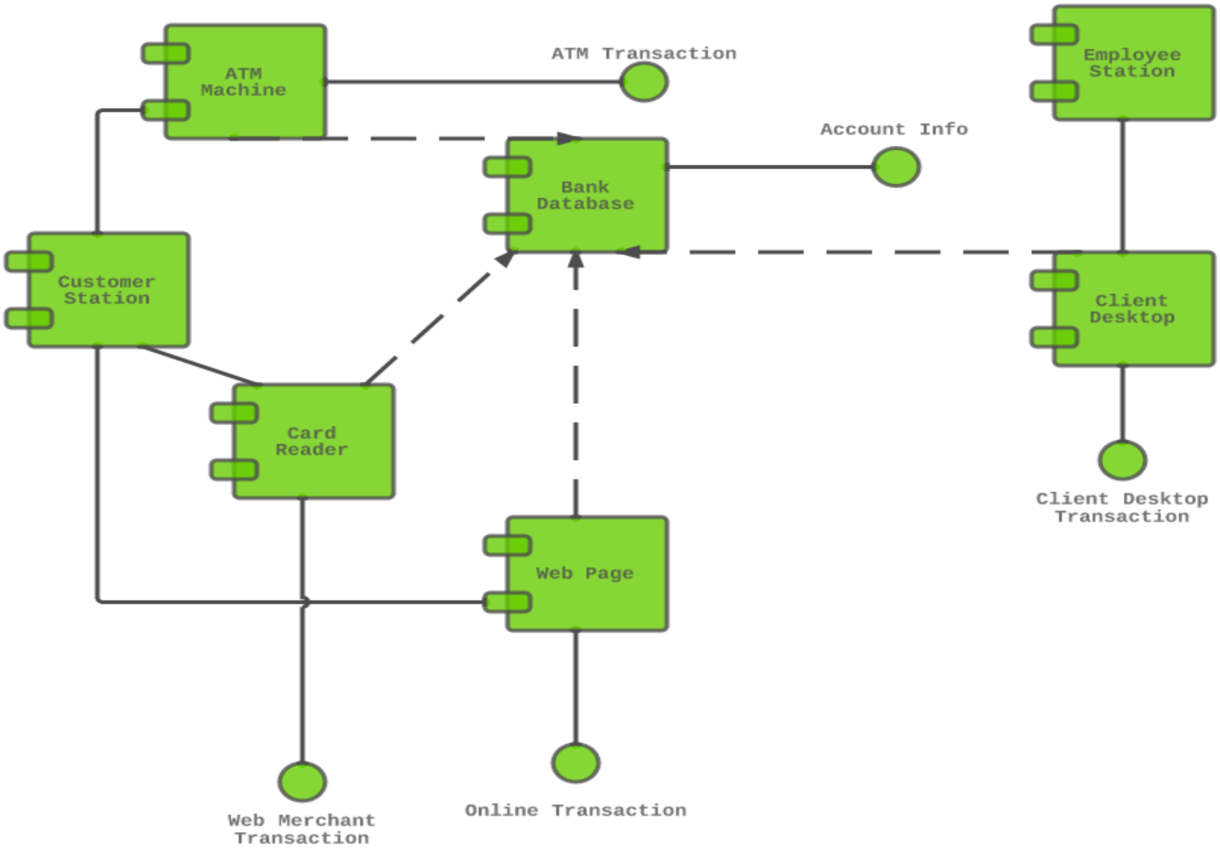
How to Draw a Component Diagram

- Take stock of everything needed to implement the planned system. For example, for a simple e-commerce system, you'll need components that describe products, orders, and customer accounts.
- Create a visual for each of the components.
- Describe the organization and relationships between components using interfaces, ports, and dependencies.

Examples:


LIBRARY MANAGEMENT SYSTEM
LMS database
Book
Search
Package Name
Transaction
Member
creately
www.creately.com • Online Diagramming

Component diagram for an ATM system:



Outcomes:

Group 2

1. Mini Project: Draw all UML diagrams for **your** project work.

OR

2. Mini Project - Develop a Blockchain based application for health related medical records
   Draw following UML Diagrams for Bank Management application
   a. Class Diagram
   b. Object Diagram
   c. ER Diagram
   d. Component Diagram

   Link1: https://www.freeprojectz.com/uml-diagram/banking-management-system-uml-diagram
   Link2: https://www.startertutorials.com/uml/uml-diagrams-online-banking-system.html
   Link3: https://itsourcecode.com/uml/bank-management-system-uml-diagrams/