PRACTICAL NO 1

AIM :- Write down the problem statement for a suggested system of relevance.

1) LIBRARY MANAGEMENT SYSTEM

PROBLEM STATEMENT:-

A college library management is a project that manages and stores books information electronically according to student's needs. The system helps both students and library manager to keep a constant track of all the books available in the library. It allows both the admin and the student to search for the desired book. It becomes necessary for colleges to keep a continuous check on the books issued and returned and even calculate fine. This task if carried out manually will be tedious and includes chances of mistakes. These errors are avoided by allowing the system to keep track of information such as issue date, last date to return the book and even fine information and thus there is no need to keep manual track of this information which thereby avoids chances of mistakes.

Thus, this system reduces manual work to a great extent allows smooth flow of library activities by removing chances of errors in the details.

2) HOTEL MANAGEMENT SYSTEM

PROBLEM STATEMENT:-

A hotel system manages information about rooms, reservations, customers, and customer billing. A customer can make reservations, change, or cancel reservations through the hotel website. When a customer makes reservations, he/she needs to check if a room the customer want to reserve is available. If a room is available, the customer enters his/her information to the system and receives a confirmation number from the web site.

A desk clerk checks in a customer with only a prior reservation, change the checkout date, and check out the customer. A room is assigned to the customer at check-in time and a customer billing record is created at that time. The customer billing record is updated every night at 12. When a customer checks out, the desk clerk prints the bill. A customer can pay by cash, check, or credit card when he/she checks out.

- Develop the use case diagram showing actors and use cases for the hotel system.
- Describe Reserve Room use case using the template for use case description.

State your assumptions as needed.

PRACTICAL NO 2

AIM :- Draw the function oriented diagram: Data Flow Diagram (DFD)

INTRODUCTION :-

A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement. They are often elements of a formal methodology such as Structured Systems Analysis and Design Method (SSADM). Superficially, DFDs can resemble flow charts or Unified Modeling Language (UML), but they are not meant to represent details of software logic.

All data flow diagrams include four main elements: entity, process, data store and data flow.

External Entity

– Also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. These data flows are the inputs and outputs of the DFD.

Process

– An activity that changes or transforms data flows. Since they transform incoming data to outgoing data, all processes must have inputs and outputs on a DFD.

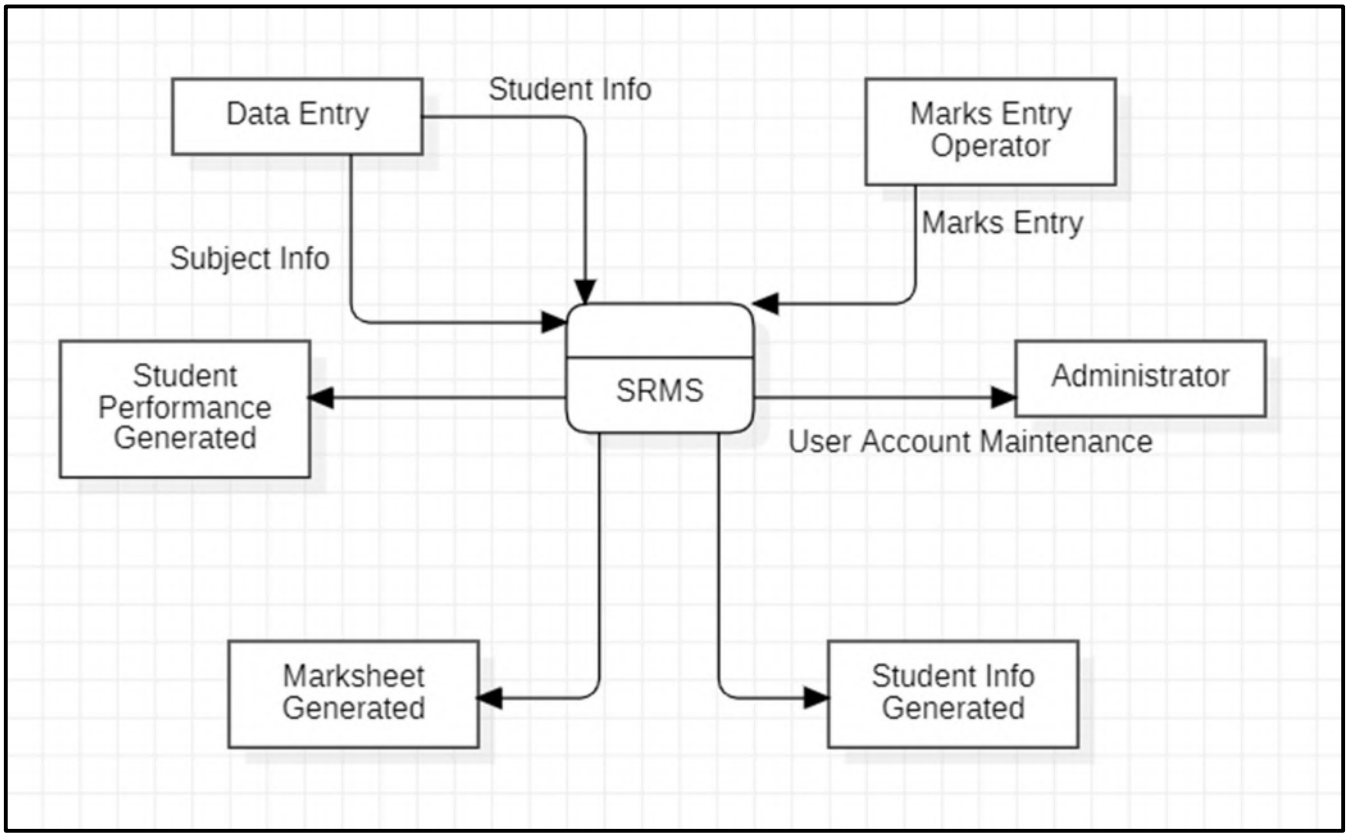
Data Store

– A data store does not generate any operations but simply holds data for later access. Data stores could consist of files held long term or a batch of documents stored briefly while they wait to be processed.

Data Flow

– Movement of data between external entities, processes and data stores is represented with an arrow symbol, which indicates the direction of flow.

DATAFLOW DIAGRAM FOR STUDENT RESULT MANAGEMENT SYSTEM :-



PRACTICAL NO 3

AIM :- Draw the user‘s view analysis for the suggested system: Use case diagram.

INTRODUCTION :-

In UML, use-case diagrams model the behaviour of a system and help to capture the requirements of the system.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

A use case diagram is a graphical representation of a user's interactions with a system. The four main components of a use case diagram are:

Actors

- Any human or external system that interacts with the system. Actors can be users, organizations, or outside systems. Actors are often represented as stick figures.

Systems

- The main components of the system being modeled. A system can describe the hardware or software a person uses to complete a task. A box represents the system, which encompasses the use cases.

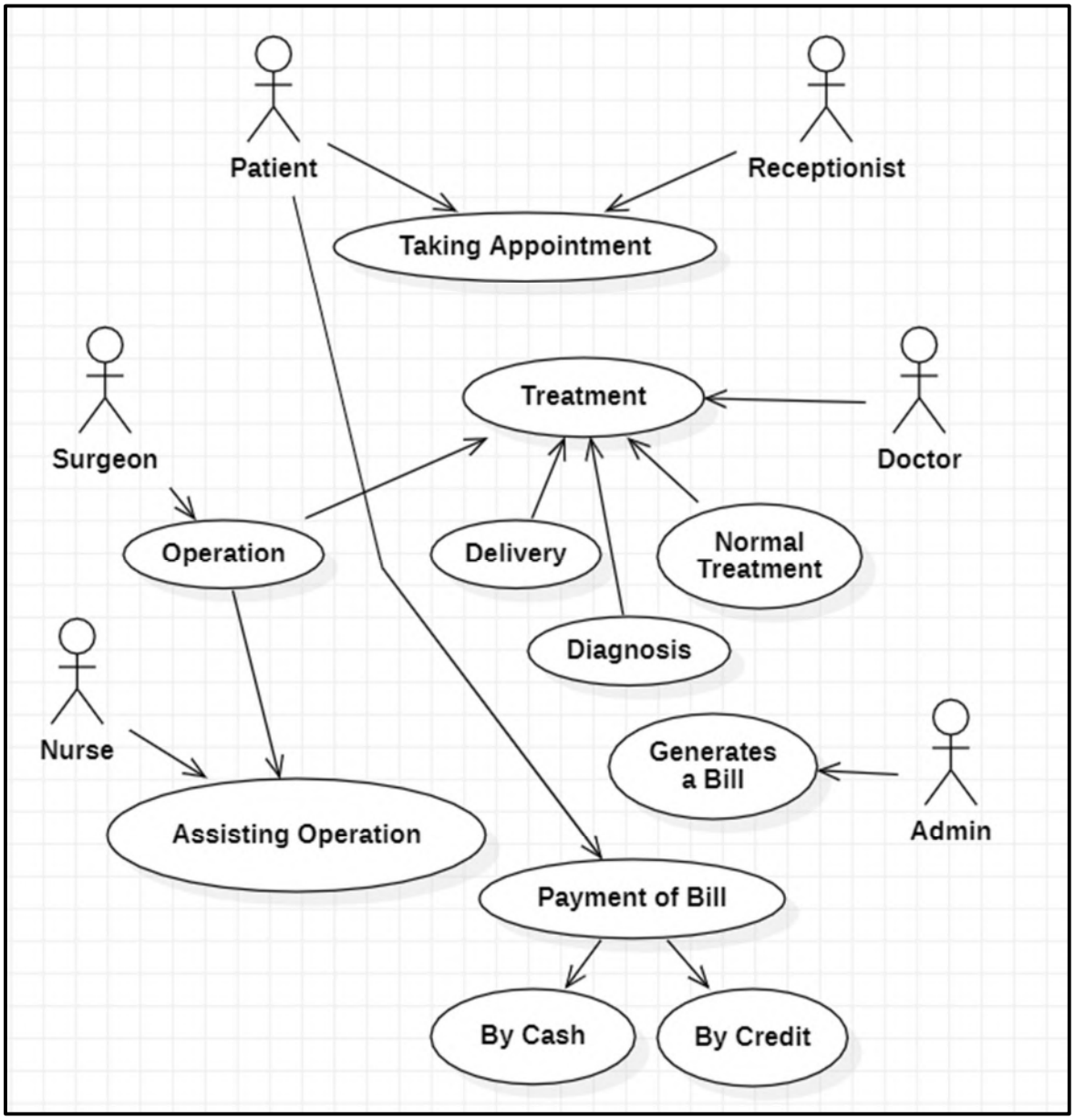
Use cases

- The specific things the system can do, such as "Place Order," "Track Delivery," or "Update Product Information". Use cases are represented by circles or ellipses.

Relationships

- Show how the actors and systems interact with each other.

USE CASE DIAGRAM FOR HOSPITAL MANAGEMENT SYSTEM :-



PRACTICAL NO 4

AIM :- Draw the structural view diagram for the system: Class diagram, object diagram.

INTRODUCTION :-

CLASS DIAGRAM :-

Class diagrams are a type of UML (Unified Modeling Language) diagram used in software engineering to visually represent the structure and relationships of classes within a system i.e. used to construct and visualize object-oriented systems. Class diagrams provide a high-level overview of a system's design, helping to communicate and document the structure of the software.

The standard class diagram is composed of three sections:

- Upper section:
  Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- Middle section:
  Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class.
- Bottom section:
  Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.

OBJECT DIAGRAM :-

Object diagrams are a visual representation in UML (Unified Modeling Language) that illustrates the instances of classes and their relationships within a system at a specific point in time. They display objects, their attributes, and the links between them, providing a snapshot of the system's structure during execution.

Object diagrams are simple to create: they're made from objects, represented by rectangles, linked together with lines. Take a look at the major elements of an object diagram.

Objects

- Objects are instances of a class.

Class titles

- Class titles are the specific attributes of a given class

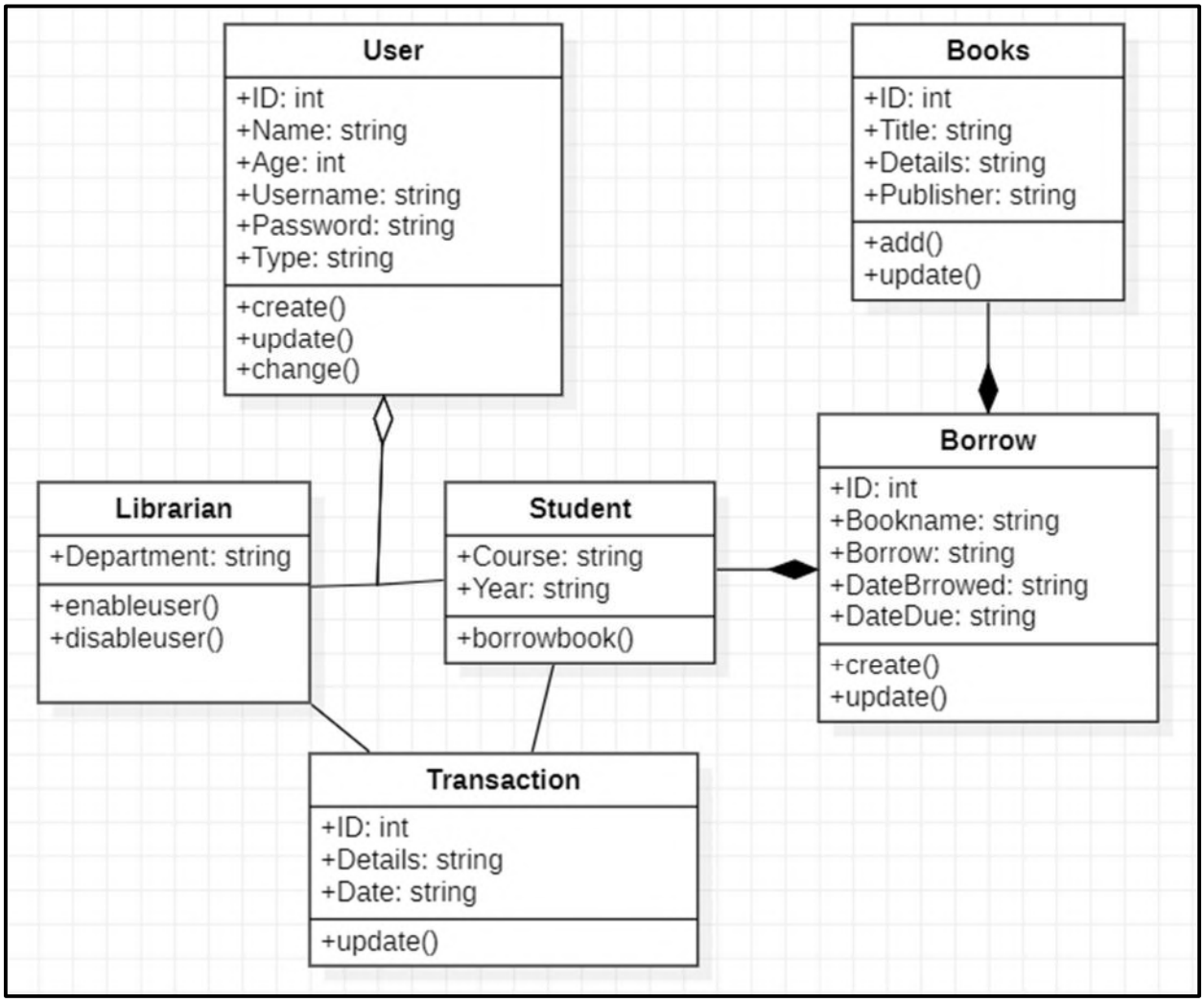
Class attributes

- Class attributes are represented by a rectangle with two tabs that indicates a software element.

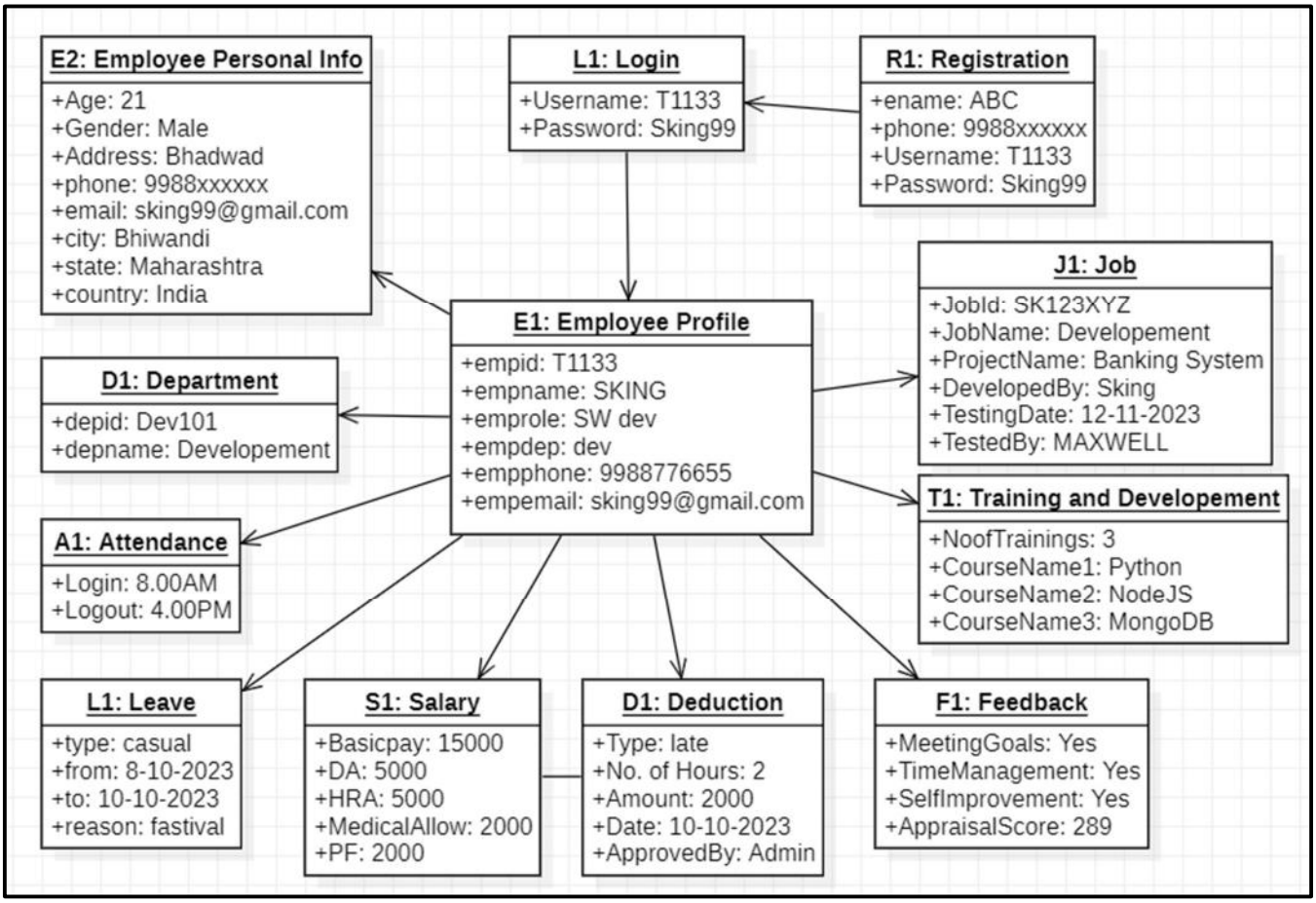
Links

- Links are the lines that connect two shapes of an object diagram to each other.

CLASS DIAGRAM :-



OBJECT DIAGRAM:-



PRACTICAL NO 5

AIM :- Draw the behavioural view diagram : State-chart diagram, Activity diagram

INTRODUCTION :-

STATE-CHART DIAGRAM :-

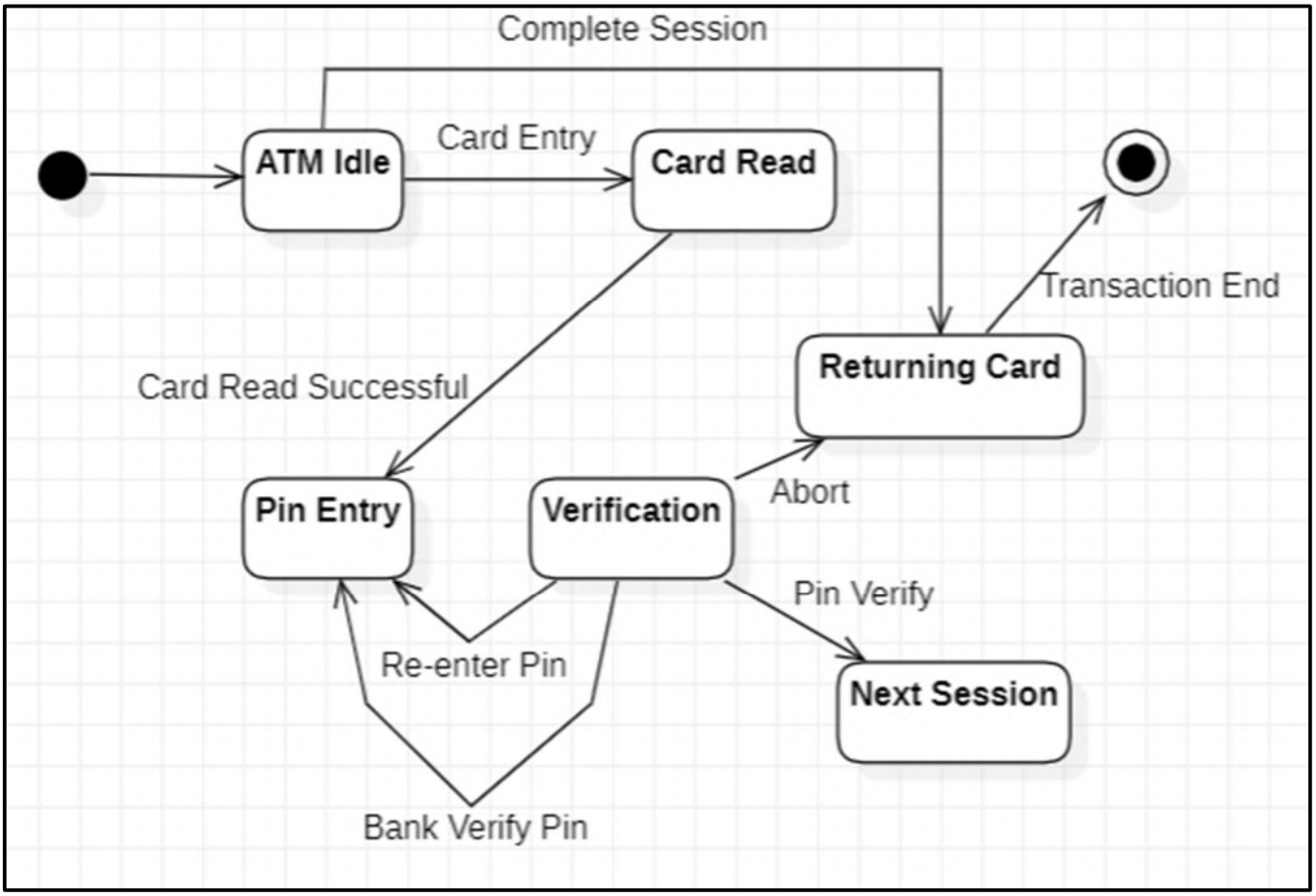
State-chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State-chart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.
While the exact structure depends heavily on the specific system being modelled, a state diagram typically includes the following basic denotations:
Initial state. A solid black circle that represents the initial state of a system or a class.
States. Individual states are portrayed as boxes with rounded corners that contain the name of the state (usually written in CamelCase).
State actions. Any actions that a particular state is expected to invoke are designated by a box with two sections: An upper section containing the name of the state and a lower section containing the actions it will initiate (as well as any relevant variable names).
Transitions. Straight lines each with an arrow at one end connect various pairs of state boxes, providing an ordered designation of the state transitions that will occur.
Final state. Final state is portrayed as a large black dot with a circle around it.
In addition to these basic components, state diagrams can also indicate forked, joined, self-transitioning, composite and historical states.

ACTIVITY DIAGRAM :-

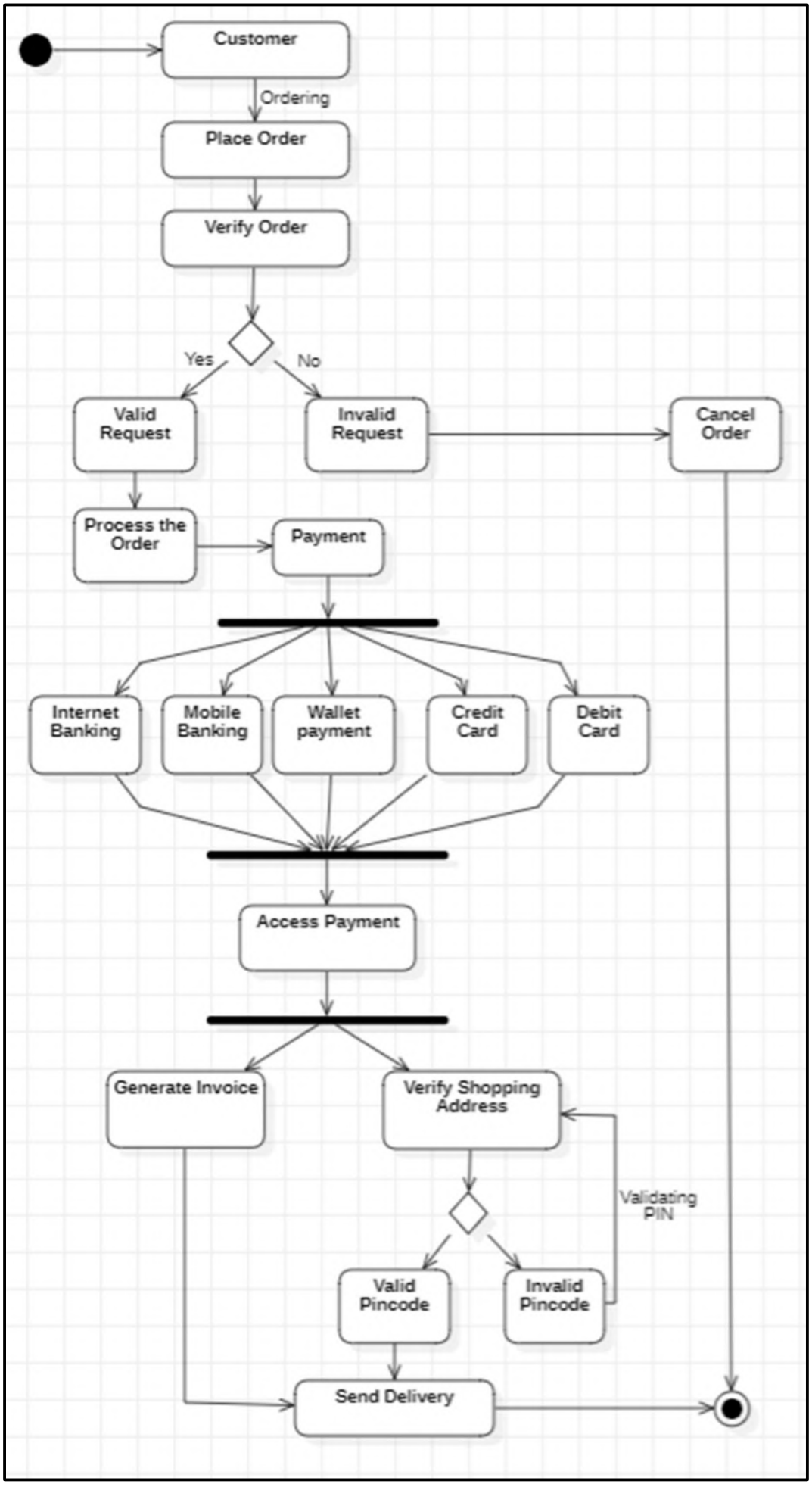
An activity diagram is a type of Unified Modelling Language (UML) flowchart that shows the flow from one activity to another in a system or process. It's used to describe the different dynamic aspects of a system and is referred to as a 'behaviour diagram' because it describes what should happen in the modelled system.

Initial node	Represents the starting point of an activity.
Activity state	Represents the activities within the process.
Action	Represents the executable sub-areas of an activity.
Control flow	Represents the flow of control from one action to another.
Object flow	Represents the path of the objects moving through the activity.
Activity final node	Represents the end of all control flows within the activity.
Flow final node	Represents the end of a single control flow.
Decision node	Represents a conditional branch point with a single input and multiple outputs.
Merge node	Represents the merging of flows with several inputs and only one output.
Fork	Represents a flow that can branch into two or more parallel flows.
Merge	Represents two inputs merging into a single output.

STATE-CHART DIAGRAM :-



ACTIVITY DIAGRAM :-



PRACTICAL NO 6

AIM :- Draw the behavioural view diagram for the suggested system: Sequence diagram, Collaboration diagram

INTRODUCTION :-

SEQUENCE DIAGRAM :-

A sequence diagram is the most commonly used interaction diagram.
A sequence diagram is used to show the interactive behaviour of a system. Since visualizing the interactions in a system can be difficult, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

Actors

An actor in a UML diagram represents a type of role where it interacts with the system and its objects.

Lifelines

A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline.

Messages

Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline.

Synchronous messages

A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message.

Asynchronous Messages

An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not.

Reply Message

Reply messages are used to show the message being sent from the receiver to the sender. We represent a return/reply message using an **open arrow head with a dotted line**.

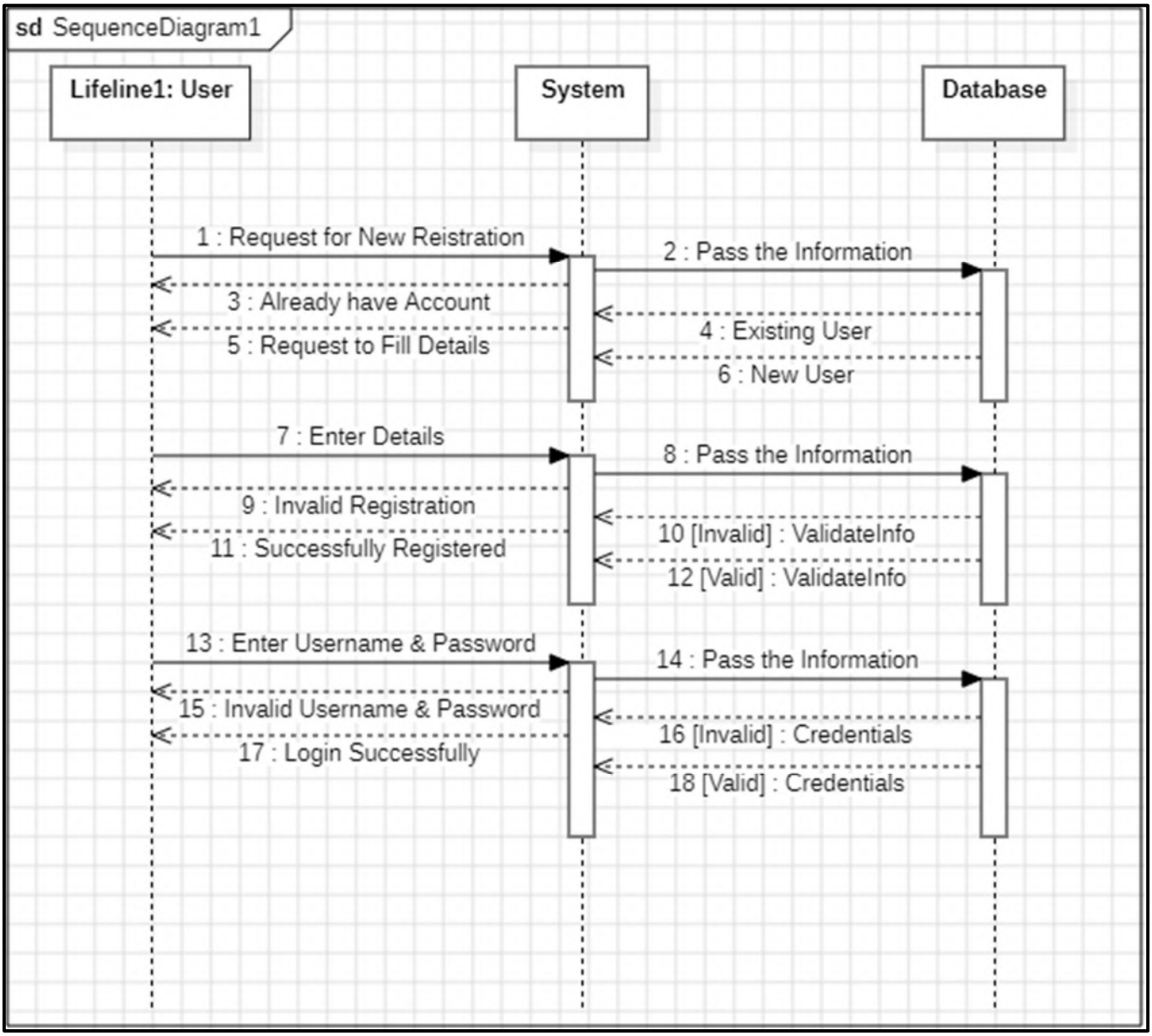
COLLABORATION DIAGRAM :-

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.

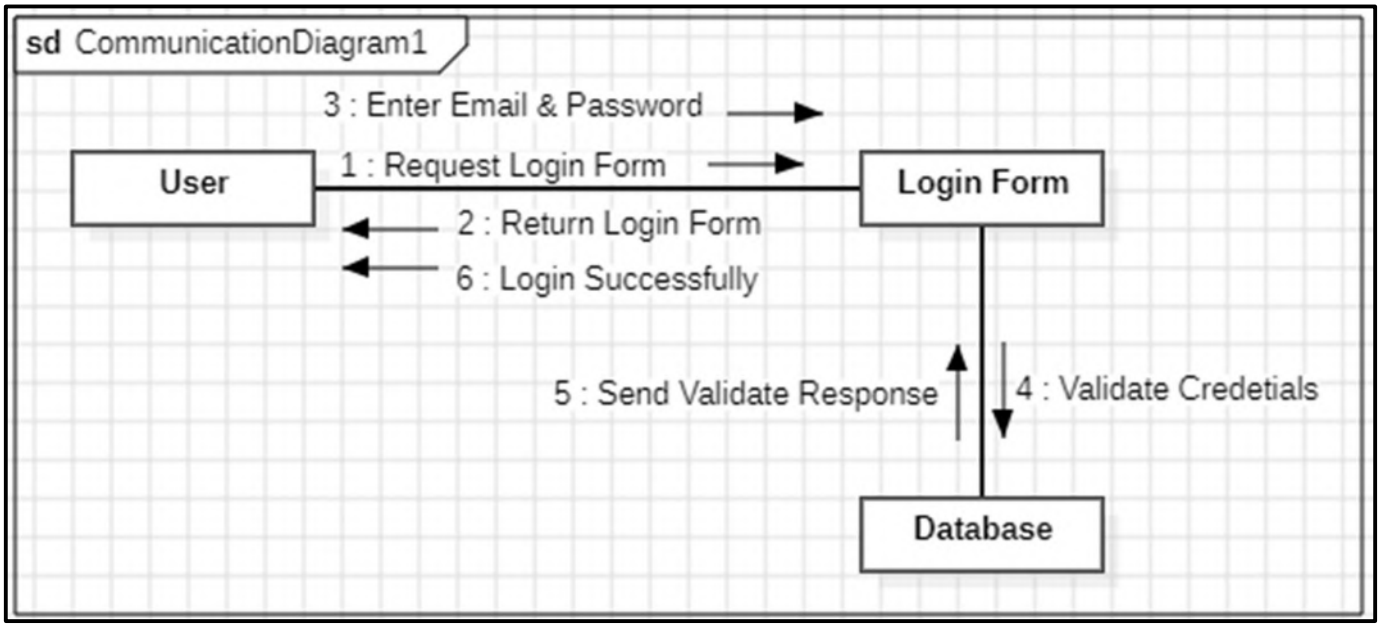
Following are the components of a component diagram that are enlisted below:

1. **Objects:** The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.
2. **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
3. **Links:** The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line.
4. **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link.

SEQUENCE DIAGRAM :-



COLLABORATION DIAGRAM :-



PRACTICAL NO 7

AIM :- Draw the implementation and environmental view diagram: Component diagram, Deployment diagram

INTRODUCTION :-

COMPONENT DIAGRAM :-

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

Component symbol	An entity required to execute a stereotype function. A component provides and consumes behavior through interfaces, as well as through other components.
Node symbol	Represents hardware or software objects, which are of a higher level than components.
Interface symbol	Shows input or materials that a component either receives or provides.
Port symbol	Specifies a separate interaction point between the component and the environment
Dependency symbol	Shows that one part of your system depends on another. Dependencies are represented by dashed lines

DEPLOYEMENT DIAGRAM :-

The deployment diagram visualizes the physical hardware on which the software will be deployed. It portrays the static deployment view of a system. It involves the nodes and their relationships.

It ascertains how software is deployed on the hardware. It maps the software architecture created in design to the physical system architecture, where the software will be executed as a node.

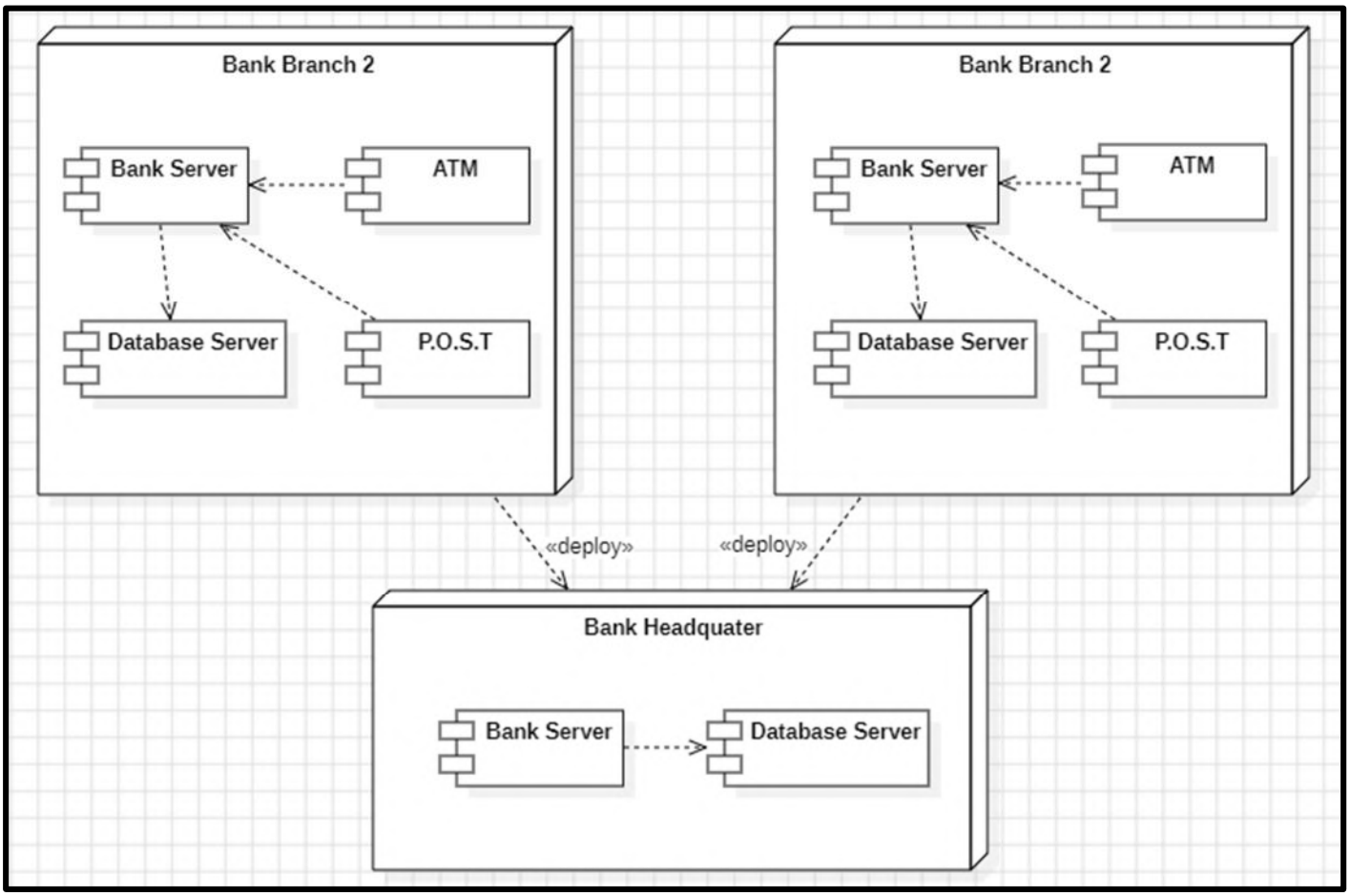
A variety of shapes make up deployment diagrams. This list offers an overview of the basic elements you may encounter, and you can see most of these items illustrated in the image below.

- Artifact:
  A product developed by the software, symbolized by a rectangle with the name and the word "artifact" enclosed by double arrows.
- Association
  : A line that indicates a message or other type of communication between nodes.
- Component:
  A rectangle with two tabs that indicates a software element.
- Dependency:
  A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.
- Interface:

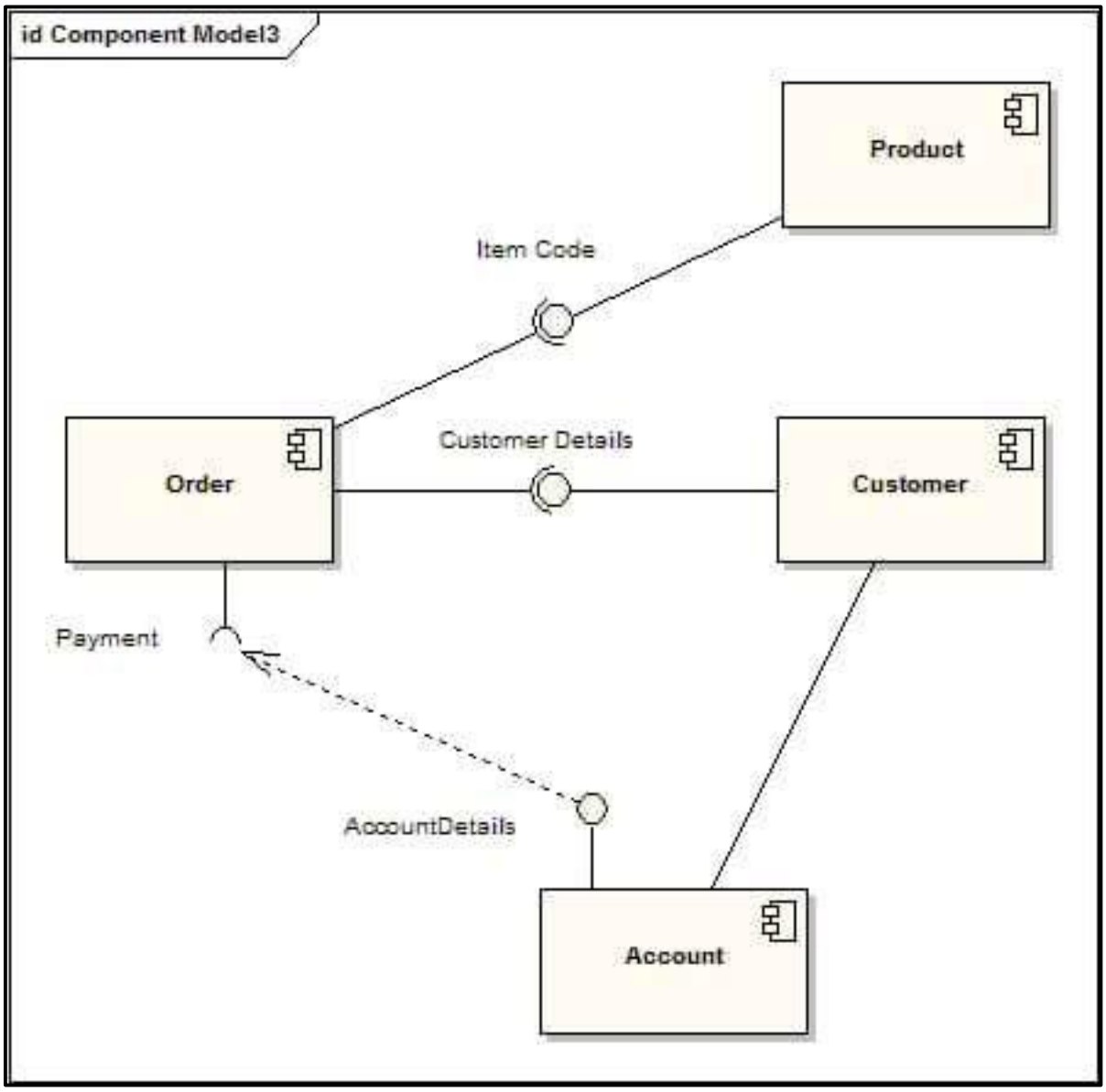
A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.

- Node:
  A hardware or software object, shown by a three-dimensional box.
- Node
  as container:
  A node that contains another node inside of it—such as in the example below, where the nodes contain components.
- Stereotype:
  A device contained within the node, presented at the top of the node, with the name bracketed by double arrows.

DEPLOYEMENT DIAGRAM :-



COMPONENT DIAGRAM :-



PRACTICAL NO 8

AIM :- Prepare time line chart/Gantt Chart/PERT Chart

INTRODUCTION :-

Generalized Activity Normalization Time Table (GANTT) chart is type of chart in which series of horizontal lines are present that show the amount of work done or production completed in given period of time in relation to amount planned for those projects. It is horizontal bar chart developed by Henry L. Gantt (American engineer and social scientist) in 1917 as production control tool. It is simply used for graphical representation of schedule that helps to plan in an efficient way, coordinate, and track some particular tasks in project.

Gantt charts are made up of nine components.

1. **Dates.** One of the main components of a Gantt chart, the dates allow project managers to see not only when the entire project will begin and end, but also when each task will take place.
2. **Tasks.** Large projects always consist of a large number of sub-tasks. A Gantt chart helps project managers keep track of all of the sub-tasks in a project, so nothing is forgotten or delayed.
3. **Bars.** Once the sub-tasks have been listed, bars are used to show the time frame in which each task should be completed.
4. **Milestones.** Milestones are those tasks that are instrumental to a project's completion and success. Unlike the minor details, which also have to be done, completing a milestone offers a sense of satisfaction and forward motion.
5. **Arrows.** While some of your tasks can be done at any time, others must be completed before or after another sub-task can begin or end.
6. **Taskbars.** While many sub-tasks can be completed fairly quickly, there will be plenty of times when you will want to see at a glance exactly how your project is coming along.
7. **Vertical Line Marker.** Another way to monitor your project's progress, a vertical line marker indicates the current date on the chart.
8. **Task ID.** In today's fast-paced business world, you likely have several tasks going on at the same time.
9. **Resources.** While not every Gantt chart lists the names of the people who will be working on it, if your project will be completed by a number of individuals, listing names and the tasks that are assigned to them can be incredibly helpful.

GANTT CHART :-

Task	Start Date	End Date	Duration
Project Kick Off	01-01-2017	01-08-2017	7
Gather Project Requirement	01-08-2017	01-22-2017	14
Develop Database	01-22-2017	02-05-2017	14
Develop UI	02-05-2017	02-19-2017	14
Testing	02-19-2017	03-12-2017	21
Implementation	03-12-2017	03-19-2017	7
Custom Acceptance	03-19-2017	03-26-2017	7
Maintanence	03-26-2017	04-25-2017	30