**Experiment 7**

**Aim:** To perform the behavioral view diagram for the suggested system: Sequence diagram, Collaboration diagram.

**Description :**

**Sequence diagrams** show potential interactions between objects in the system being defined. Normally these are specified as part of a use case or use case flow and show how the use case will be implemented in the system. They include:

**Objects** - oblong boxes or actors at the top - either named or just shown as belonging to a class, from, or to which messages are sent to other objects.

**Messages** - solid lines for calls and dotted lines for data returns, showing the messages that are sent between objects including the order of the messages which is from the top to the bottom of the diagram.

**Object lifelines** - dotted vertical lines showing the lifetime of the objects.

**Activation** - the vertical oblong boxes on the object lifelines showing the thread of control in a synchronous system.

Sequence diagrams show a detailed flow for a specific use case or even just part of a specific use case. They are almost self-explanatory; they show the calls between the different objects in their sequence and can show, at a detailed level, different calls to different objects. A sequence diagram has two dimensions: The vertical dimension shows the sequence of messages/calls in the time order that they occur; the horizontal dimension shows the object instances to which the messages are sent.

**Collaboration Diagram**

They are the same as sequence diagrams but without a time axis:

* Their message arrows are numbered to show the sequence of message sending.
* They are less complex and less descriptive than sequence diagrams.
* These diagrams are very useful during design because you can figure out how objects communicate with each other.

**Performance Instruction:**

**To draw Sequence Diagram**

1)Identify the class instances (objects) by putting each class instance inside a box.

2)If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line.

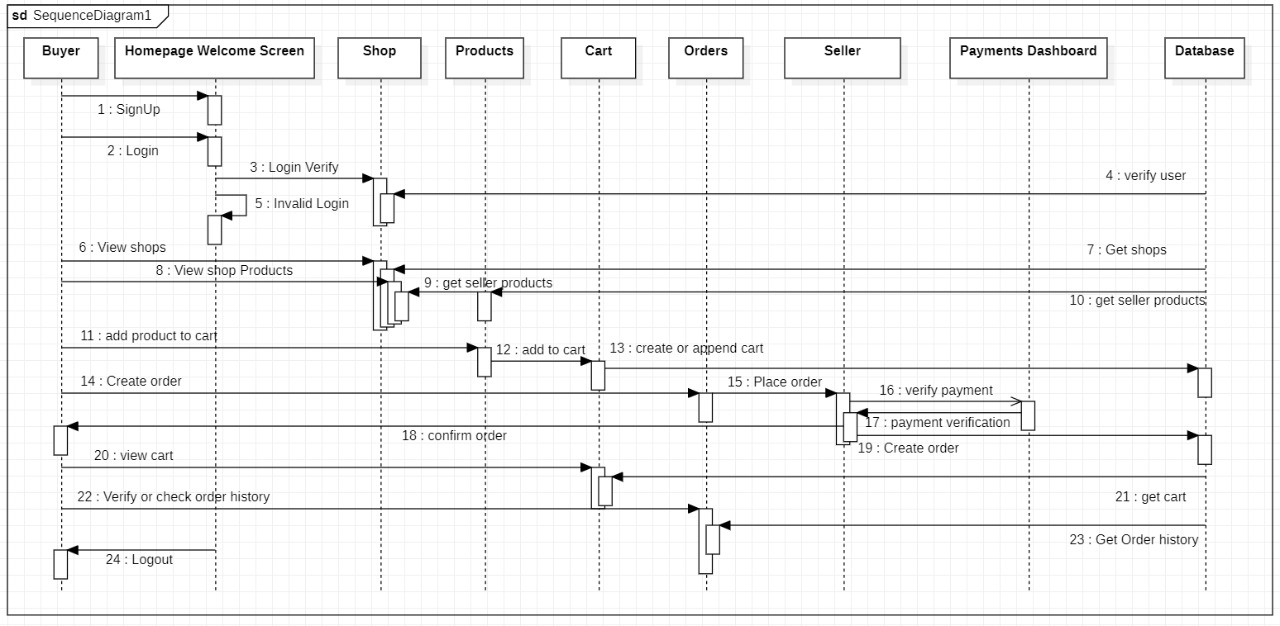
3)Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the return value above the dotted line.

**To draw collaboration Diagram**

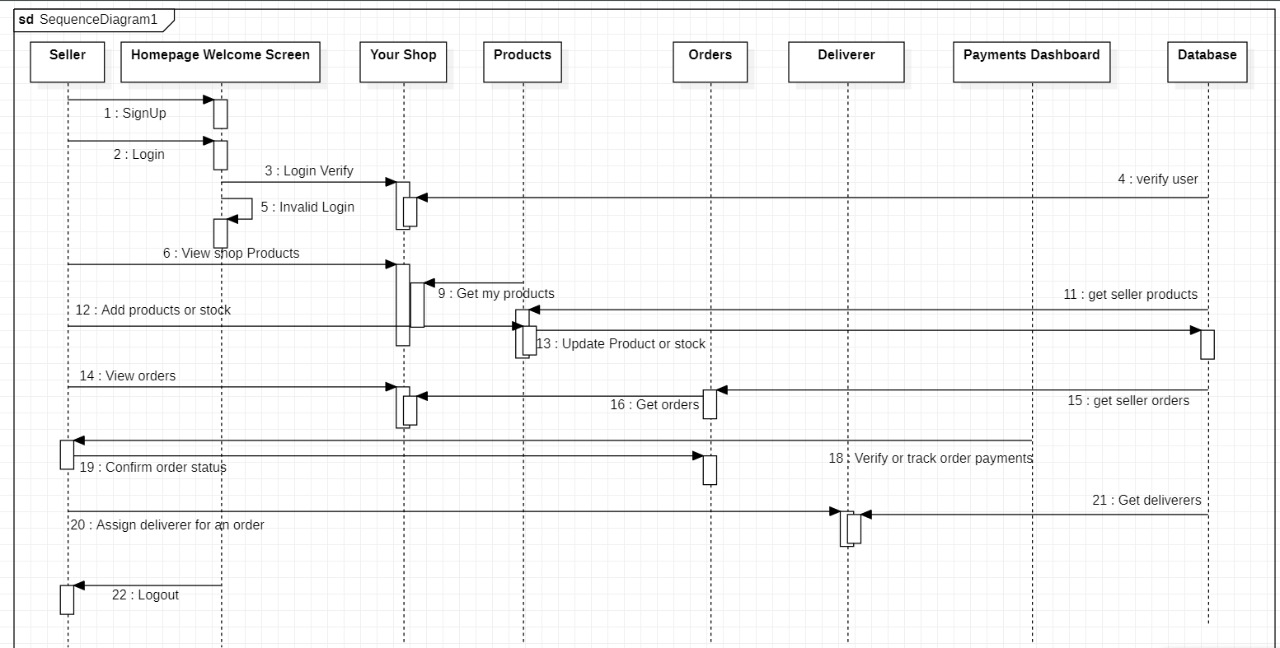
1)By simply pressing combination of keys, we can design collaboration diagram from sequence diagram.

**Sample Output:**

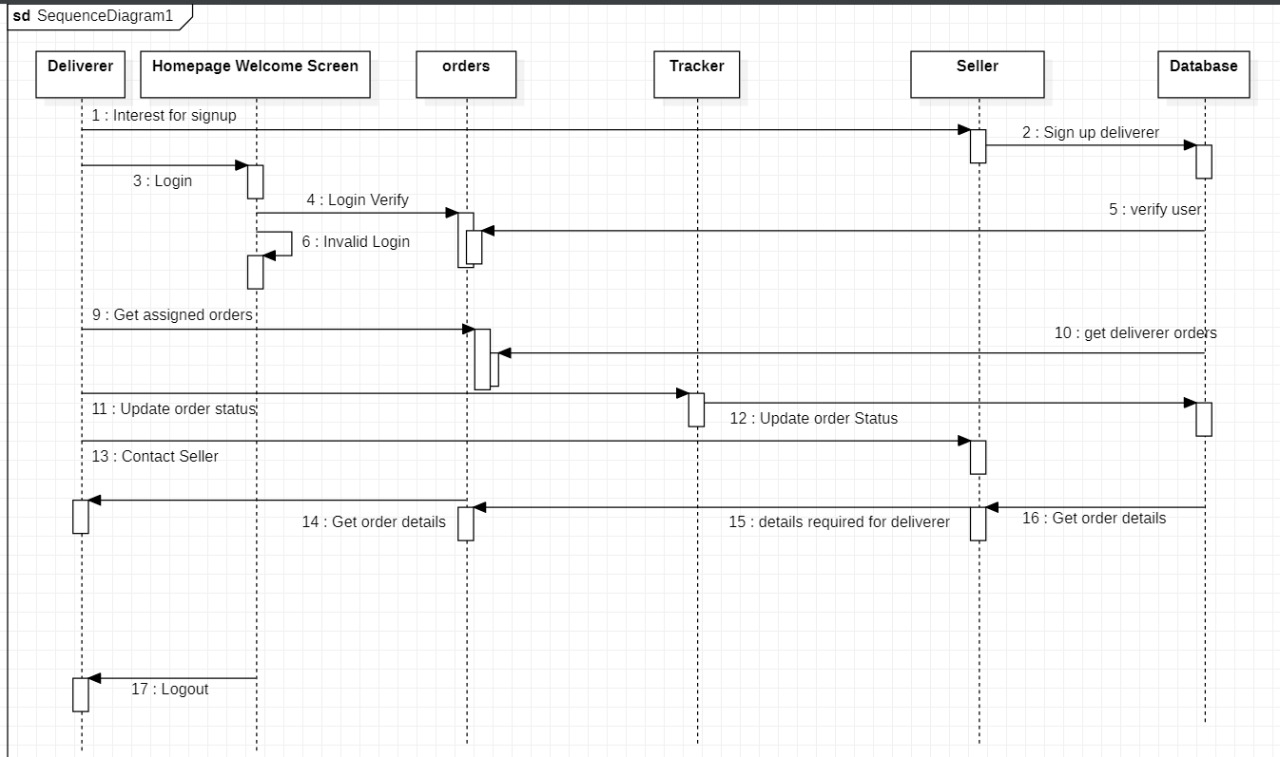
**Sequence Diagram for Buyer**



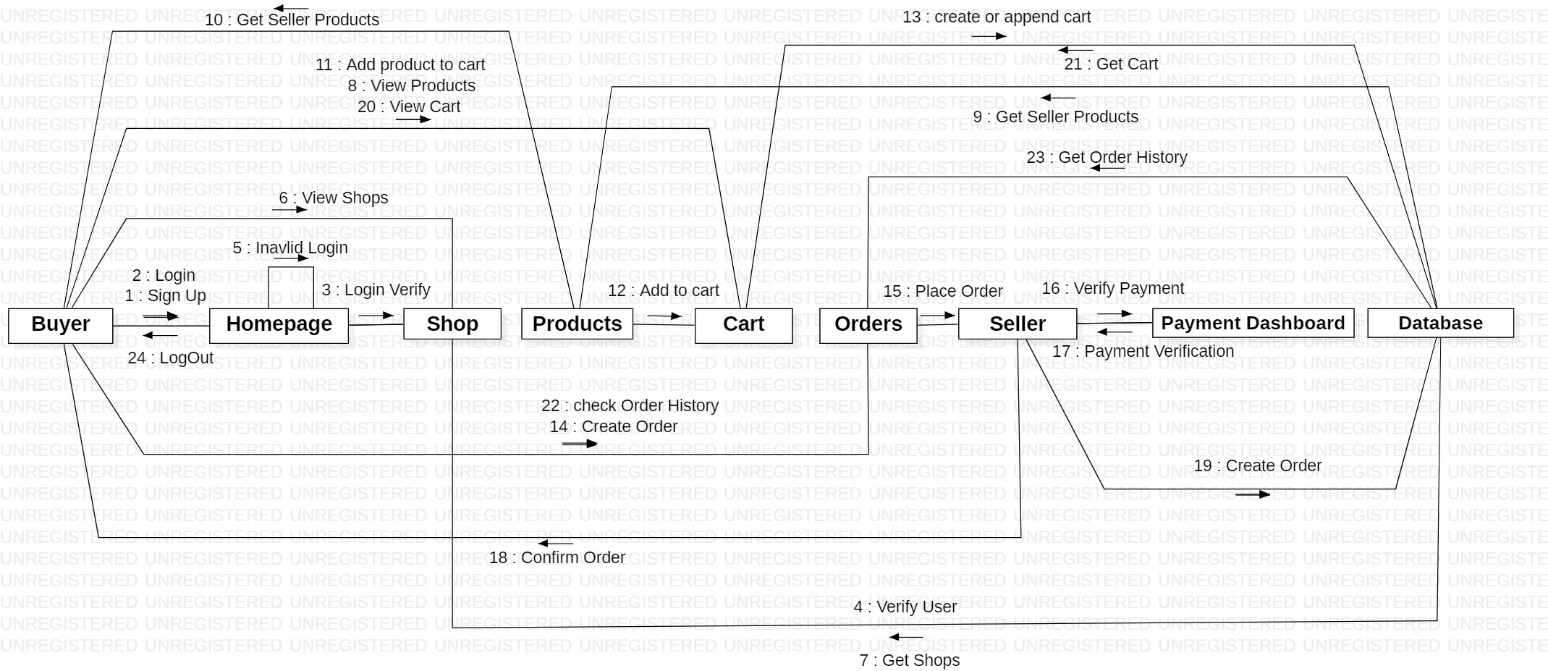
**Sequence Diagram for Seller**



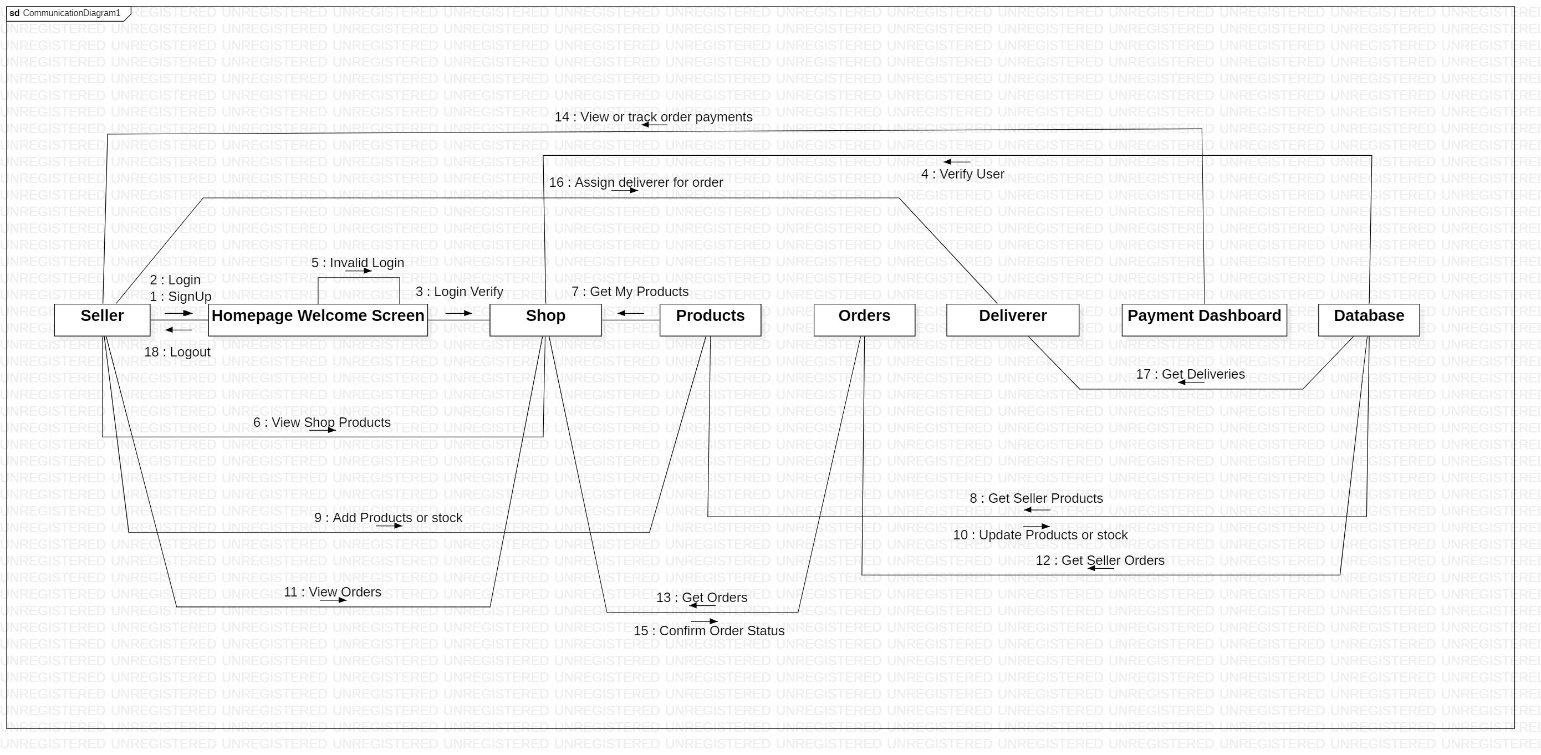
**Sequence Diagram for Deliverer**



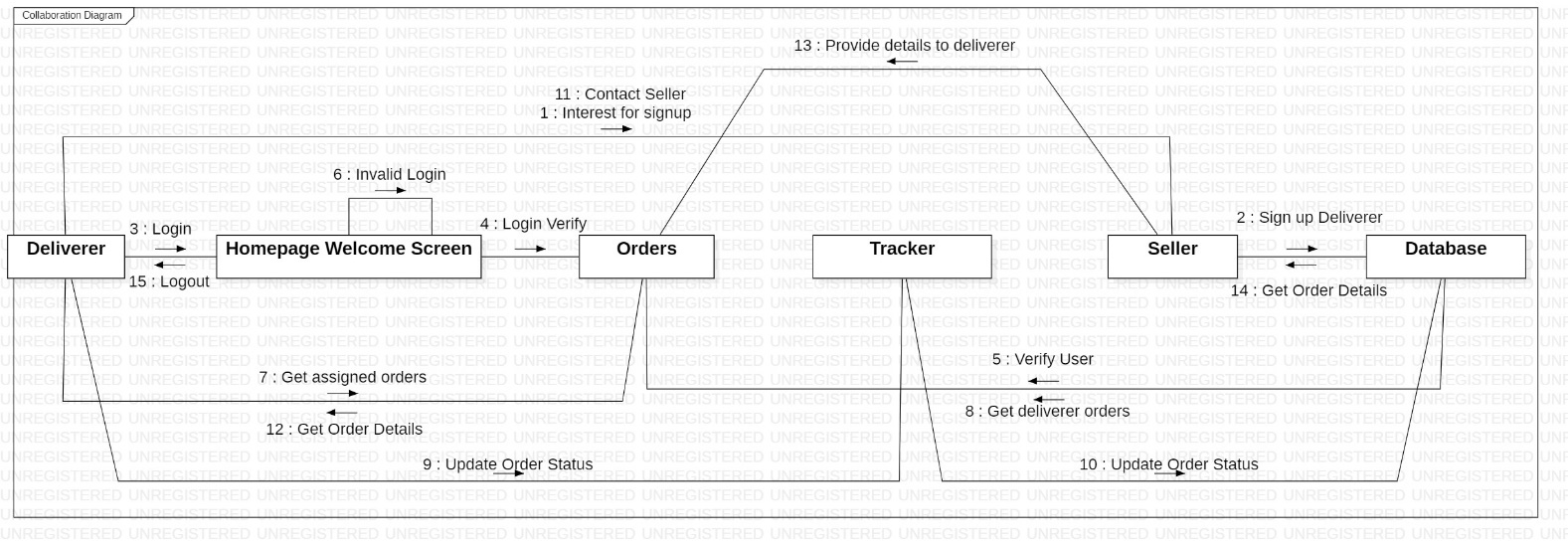
**Collaboration Diagram for Buyer**



**Collaboration Diagram for Seller**



**Collaboration Diagram for Deliverer**



**Conclusion:** Sequence diagram and Collaboration Diagram were made successfully by following above steps.

**Viva - Questions:**

**Q-1. Explain use of sequence diagram?**

**Ans.** The following scenarios are ideal for using a sequence diagram:

* **Usage scenario:** A usage scenario is a diagram of how your system could potentially be used. It's a great way to make sure that you have worked through the logic of every usage scenario for the system.
* **Method logic:** Just as you might use a UML sequence diagram to explore the logic of a use case, you can use it to explore the logic of any function, procedure, or complex process.
* **Service logic:** If you consider a service to be a high-level method used by different clients, a sequence diagram is an ideal way to map that out.
* **Sequence diagram Visio** - Any sequence diagram that you create with Visio can also be uploaded into Lucidchart. Lucidchart supports .vsd and .vdx file import and is a great Microsoft Visio alternative. Almost all of the images you see in the UML section of this site were generated using Lucidchart.

**Q-2. Explain steps to draw sequence diagram?**

**Ans.**

1. Identify the class instances (objects) by putting each class instance inside a box.
2. If a class instance sends a message to another class instance, draw a line with an open arrowhead pointing to the receiving class instance; place the name of the message/method above the line.
3. Optionally, for important messages, you can draw a dotted line with an arrowhead pointing back to the originating class instance; label the return value above the dotted line.

**Q-3. Explain need of collaboration diagram?**

**Ans.**

1. Modeling collaborations, mechanisms or the structural organization within a system design.
2. Providing an overview of collaborating objects within an object-oriented system.
3. Exhibiting many alternative scenarios for the same use case.
4. Demonstrating forward and reverse engineering.
5. Capturing the passage of information between objects.
6. Visualizing the complex logic behind an operation.

**Q-4. Explain steps to draw collaboration diagram?**

**Ans.** By simply pressing combination of keys, we can design collaboration diagram from sequence diagram.

**Q-5. Explain terms- entity objects, interface objects and control objects?**

**Ans. Entity Objects :** Entity objects are classes that encapsulate the business model, including rules, data, relationships, and persistence behavior, for items that are used in your business application.

**Interface objects:** Interface objects provide all details on how one process component exchanges data with another, for example, the mode of communication and the data structures.

**Control objects :** Control objects are also known as widgets or gadgets and they can be used in windows and dialog boxes. They cannot exist outside a window or dialog box, so you have to define a window or dialog box and select it before you can define controls.

**Experiment 8**

**Aim:** To perform the implementation view diagram: Component diagram for the system.

**Description:** A component diagram provides a physical view of the system. Its purpose is to show the dependencies that the software has on the other software components (e.g., software libraries) in the system. The diagram can be shown at a very high level, with just the large-grain components, or it can be shown at the component package level. [Note: The phrase component package level is a programming language-neutral way of referring to class container levels such as .NET's namespaces (e.g., System.Web.UI) or Java's packages (e.g., java.util).

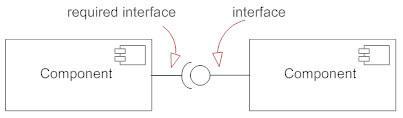
Basic Component Diagram Symbols and Notations **Component**

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.

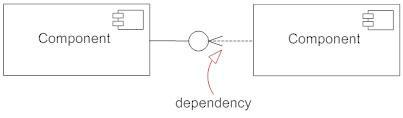


**Interface**

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.

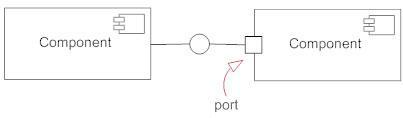


***Dependencies***

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.

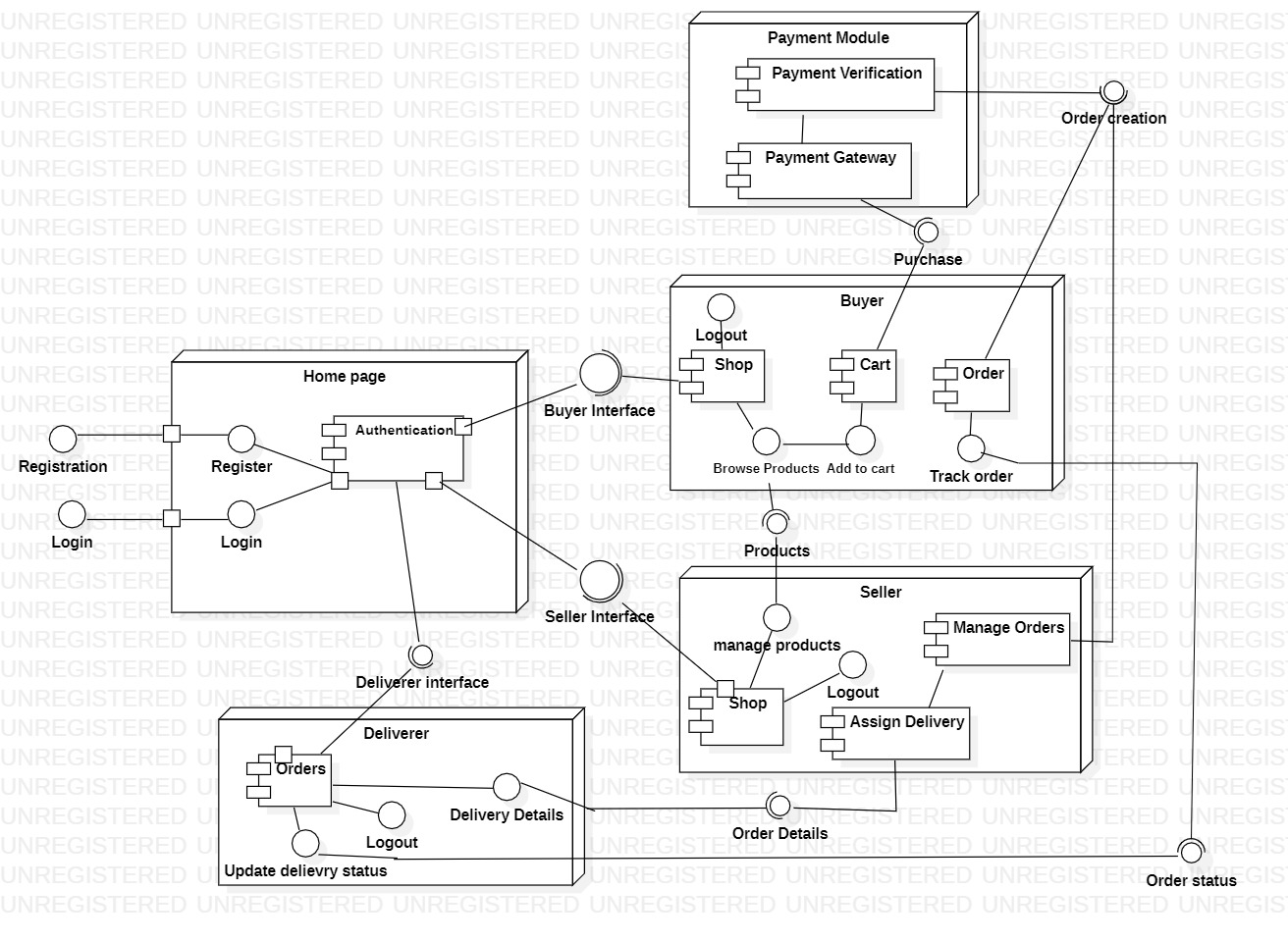


**Performance Instruction:**

**To Draw a Component Diagram**

1. 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.
2. Create a visual for each of the components.
3. Describe the organization and relationships between components using interfaces, ports, and dependencies.

**Sample Output:**



**Component Diagram**

**Conclusion:** Component diagram was made successfully by following above steps

**Viva - Questions:**

**Q-1. Explain term component diagram?**

**Ans.** A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development.

**Q-2. Component diagram explains which view of system?**

**Ans.** The component diagram also describes the static view of a system, which includes the organization of components at a particular instant.

**Q-3. Explain steps to draw component diagram?**

**Ans.**

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

**Q-4. What is benefit of drawing component diagram?**

**Ans.** Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities.

**Q-5. Explain symbols used to draw component diagram?**

**Ans.**

***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. ​

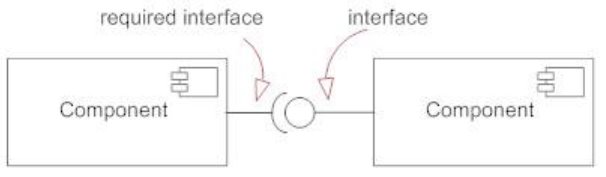
​

****

***Interface***​

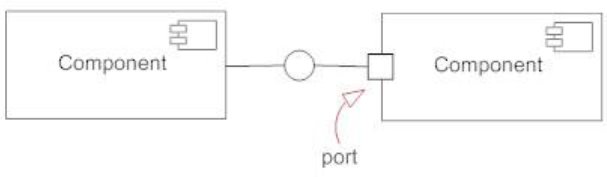
An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input. ​

​

****

***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 .​



**Experiment 9**

**Aim:** To perform the environmental view diagram: Deployment diagram for the system.

**Description:** The deployment diagram shows how a system will be physically deployed in the hardware environment. Its purpose is to show where the different components of the system will physically run and how they will communicate with each other. Since the diagram models the physical runtime, a system's production staff will make considerable use of this diagram.

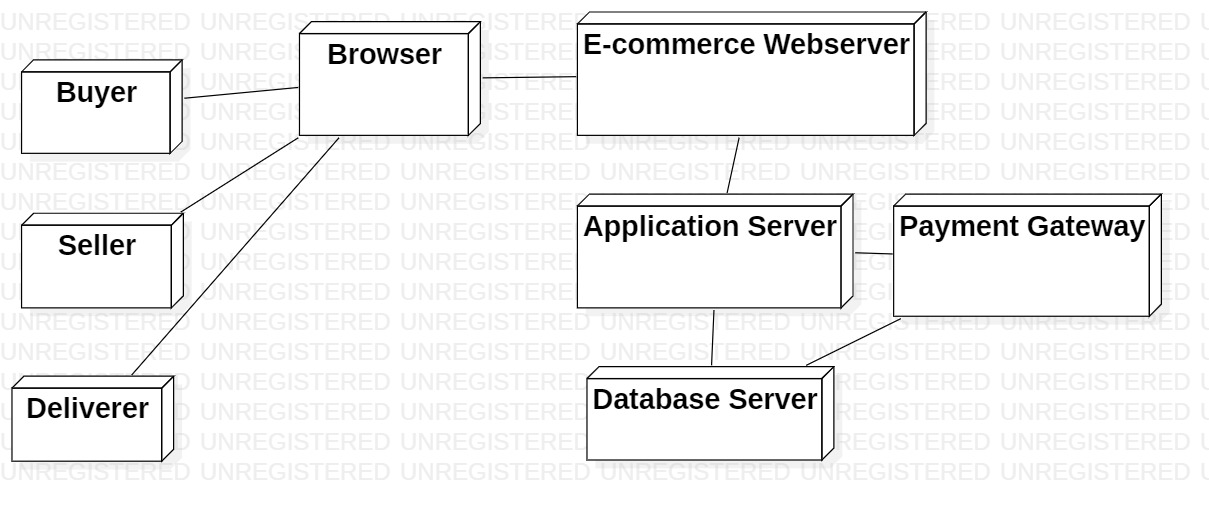
The notation in a deployment diagram includes the notation elements used in a component diagram, with a couple of additions, including the concept of a node. A node represents either a physical machine or a virtual machine node (e.g., a mainframe node). To model a node, simply draw a three-dimensional cube with the name of the node at the top of the cube.

**Performance Instruction:**

To create a deployment diagram

1. Identify component
2. Add shapes
3. Connect Nodes
4. Format arrows

**Sample Output:**



**Deployment Diagram**

**Conclusion:** Deployment diagram was made successfully by following above steps.

**Viva - Questions:**

**Q-1. What is deployment diagram?**

**Ans.** 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.

**Q-2. Deployment diagram explains which view of system?**

**Ans.** Deployment diagrams are used to describe the static deployment view of a system

**Q-3. Explain steps to draw deployment diagram?**

**Ans.** To create a deployment diagram

1. Identify component
2. Add shapes
3. Connect Nodes
4. Format arrows

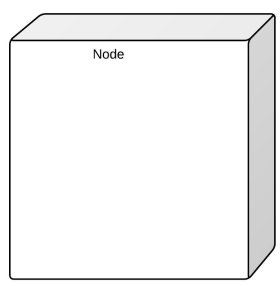
**Q-4. Explain symbols used to draw deployment diagram?**

**Ans. Communication path:** A straight line that represents communication between two device nodes.

**Artifacts:** A box with the header ">" and then the name of the file.

**Package:** A file-shaped box that groups together all the device nodes to encapsulate the entire deployment.

**Nodes**



There are two types of nodes in a deployment diagram: device nodes and execution environment nodes. Device nodes are computing resources with processing capabilities and the ability to execute programs. Some examples of device nodes include PCs, laptops, and mobile phones.

**Q-5. What is benefit of drawing deployment diagram?**

**Ans.** System engineers mainly consume deployment diagrams. These diagrams help us to describe the physical components like hardware involved, participant nodes, their distribution and how they are inter-connected Deployment diagrams could be assumed as the hardware components where the software components reside.

​