**SOFTWARE ENGINEERING**

**LAB (ETCS – 353)**

**Faculty Name : Student Name:**

Dr.Garima Gupta Udit Agarwal

12914802719

Sem 5

5C7



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

**INDEX**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.no** | **Experiments** | **Date of**  **performance** | **Date checked** | **Marks** | | | | | **Total Marks** | **Sign.** |
| **R1** | **R2** | **R3** | **R4** | **R5** |
| 1 | Write down the problem statement for a suggested system of relevance. | 23.9.21 |  |  |  |  |  |  |  |  |
| 2 | perform the function oriented diagram: Data Flow Diagram (DFD) , ER Diagram and Structured chart. | 30.9.21 |  |  |  |  |  |  |  |  |
| 3 | Perform the user’s view analysis for the suggested system: Use case diagram | 7.10.21 |  |  |  |  |  |  |  |  |
| 4 | To draw the structural view diagram for the system: Class diagram, object diagram. | 15.10.21 |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

**EXPERIMENT 1**

**Aim:** Write down the problem statement for a suggested system of relevance.

**Description:** The problem statement is the initial starting point for a project. It is basically a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem.

The input to requirement engineering is the problem statement prepared by customer.

It may give an overview of the existing system along with broad expectations from the new system.

The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So from here begins the preparation of problem statement.

So, basically a problem statement describes **what** needs to be done without describing **how**.

**Performance Instruction:**

1. Choose any one project from given list.
2. Collect all requirements
3. Identify functionalities
4. Write a one to three-page statement that everyone on the project agrees with that describes what will be done at a high level.

**Sample Output:**

**PROBLEM STATEMENT FOR B2C SYSTEM**

This would be a B2C System with 3 interfaces: an interface of an individual seller or small business an interface of a customer, and an efficient delivery system.

Current B2B systems don't support intermediary help for product delivery or transfer so we would provide aid for the same. This would also create revenue and employee opportunities

We aim at automating many operations performed at delivering the item to the customer. In many b2c services that already exist trade is carried out entirely at personal levels, with the risks of leaking of personal information and product not reaching the customer.

This can be solved if the seller uses one of the partner courier services. The seller can fill up an online request with all the details for their parcel,  get the parcel picked up at home get the cost of the delivery instantly when the pickup person arrives and a tracking ID is shared with both the seller and customer. A feedback system can also be implemented on both the sender and receiver end.

# **Features**

1. **E-commerce website (Interface for the customer):** where the seller could decide what they want to buy.
2. **Interface for Seller:** This module allows the seller to put up their product’s ad on the market.
3. **Personalized Delivery System:** This module allows the seller and customer to use the generated Tracking ID to track the current status of the product.
4. **Login  Process:**  This module allows valid customers to  access the functionalities.
5. **Product  Enquiry:**  This  module  maintains  the product details  of  a  particular product.
6. **Update  Products:**  This module  allows  the customer  to  update the product status of their  previous products they uploaded for sale.
7. **Product adds:**  This module  allows  the  customers  to  add  products they want to sell .
8. **Update profile:**  This module allows customers to change their password/ profile/ location/ details and verify contact details.
9. **Deals:**  This module allows customers to view the details of their products and browse to view someone else's.

**Viva - Questions:**

Q-1. What is problem statement?

Ans. A problem statement is a concise description of an issue to be addressed or a condition to be improved upon. It identifies the gap between the current (problem) state and desired (goal) state of a process or product. The first condition of solving a problem is understanding the problem, which can be done by way of a problem statement

Q-2. What are the benefits of writing problem statement?

Ans. A problem statement is basically a statement that illustrates a clear vision and the overall method that will be used to solve the problem at hand. Usually used when doing research, a problem statement discusses any foreseeable tangible or intangible problems that the researcher may face throughout the course of the project

Q-3. Writing a problem statement, is really a beneficial for you in proceeding project?

Ans. Yes, Writing a problem statement should help in careful decision-making for project approval. Often, the problem statement will serve as the basis for the introductory section of a final proposal, directing the reader's attention to the issues that your proposed project will address

Q-4. Explain 5W’s can be used to spark the problem?

Ans. The Five Ws  are questions whose answers are considered basic in problem solving and information gathering. They are :

* Who
* What
* When
* Where
* Why

The 5 'W's can be used to spark the discussion about the problem. A problem statement expresses the words that will be used to keep the effort focused and it should represent a solveable problem

Q-5. What are steps that need to follow while writing problem statement?

Ans. Steps that we need to follow while writing problem statement are:

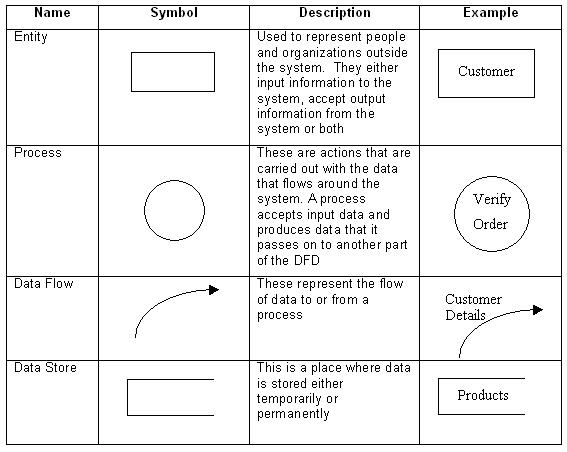
1. Define the problem.
2. Reason for the problem's occurrence.
3. When the problem began or was first noticed.
4. Place of the problem's first occurrence or sighting.
5. The person or thing that the problem affects.
6. The sequence of events that resulted in the problem.

**EXPERIMENT 2**

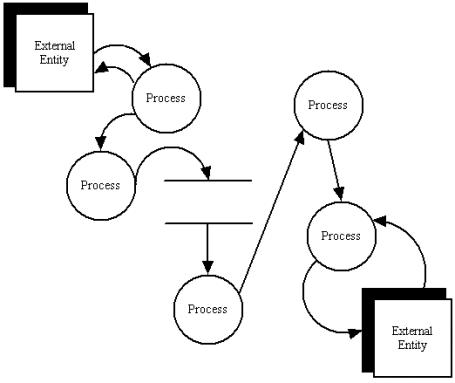
**Aim:** To perform the function oriented diagram: Data Flow Diagram (DFD), ER Diagram and Structured chart.

**Description:** Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical and logical information systems. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

**Symbols of DFD:**



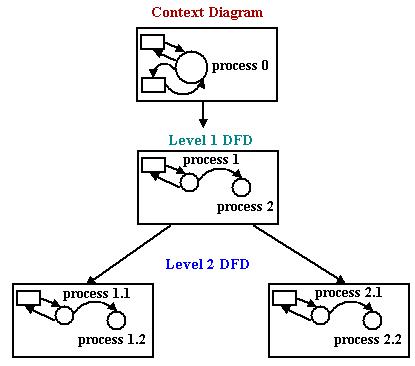
Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.



**Data Flow Diagram Layers**

Draw data flow diagrams in several nested layers. A single process node on a high level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.



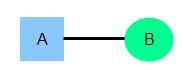


**ER Diagram :** An entity relationship model*, also called an* entity-relationship (ER) diagram*, is a* graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

***Relationships Between Entities***

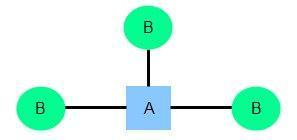
A relationship is how the data is shared between entities. There are three types of relationships between entities:

***1. One-to-One***

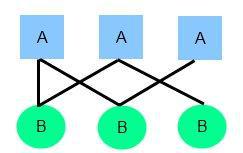
One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).

***2. One-to-Many***

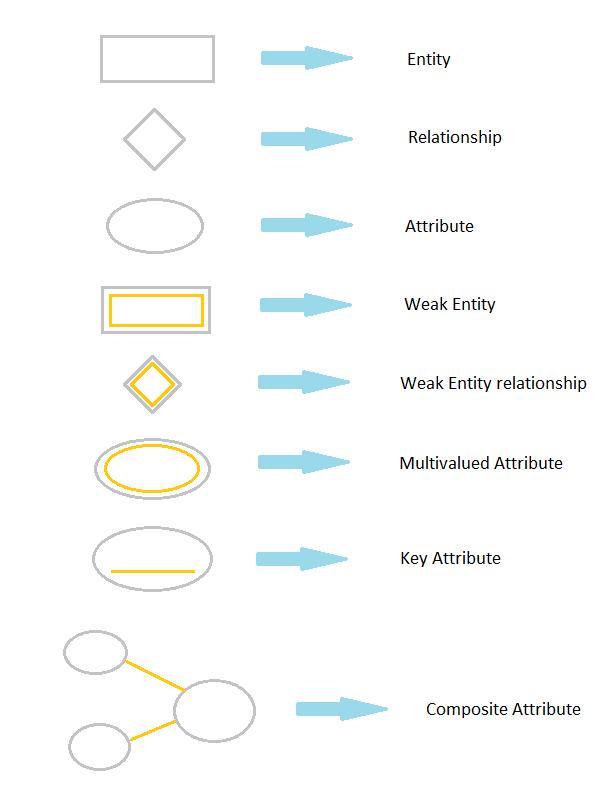
One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.



***3. Many-to-Many***

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.

**Symbols used to draw ER diagram:**



A **Structure Chart** (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name. The lines represent the connection and or ownership between activities and sub activities as they are used in organization charts. The tree structure visualizes the relationships between modules.

**Performance Instruction:**

**To draw DFD**

1. Identify various processes, data store, input, output etc. of the system and analyse it.
2. Use processes at various levels to draw the DFDs.

**To draw ER Diagram**

1)Identify all entities and their attributes.

2)Identify weak entities, attributes and their identifying relationship.

1. Design ER diagram according to norms.
2. Imply cardinalities on diagram.

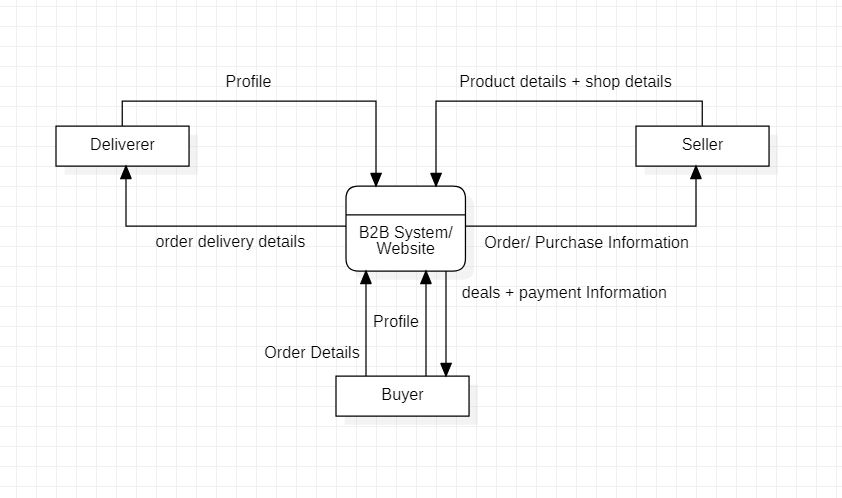
**To draw structured Chart Diagram**

1. Identify various modules, input, output etc. of the system.
2. Draw structured chart diagram describing it in form of levels.

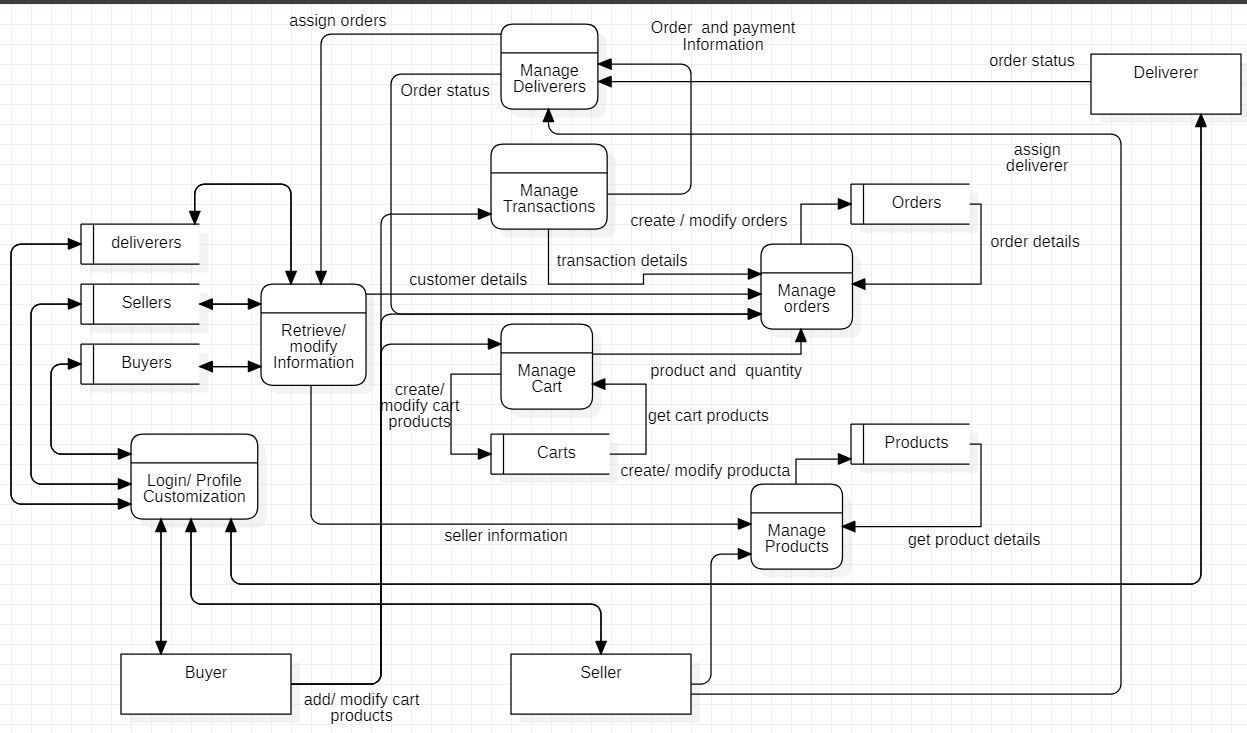
**Sample Output:**

DFD Diagram :

Level 0 :

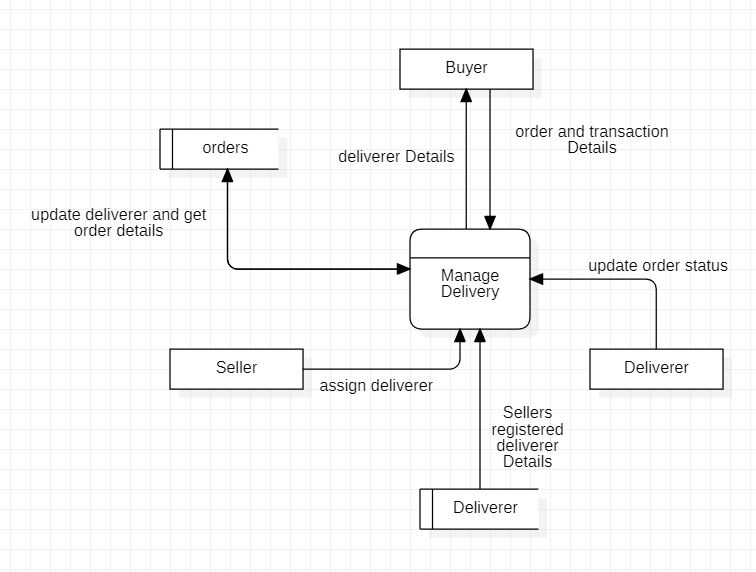


Level 1 :

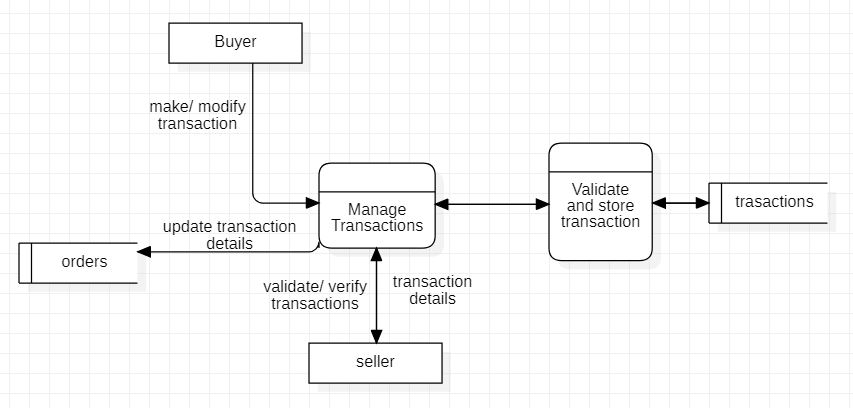


Level 2 :

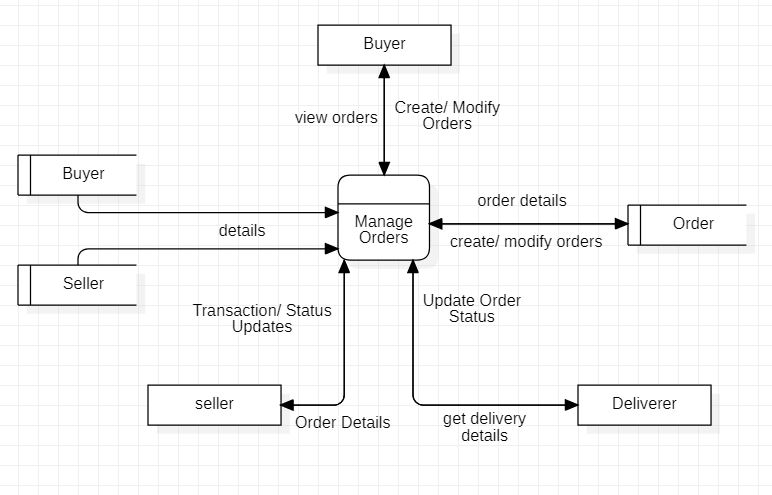
Level 2.1 :



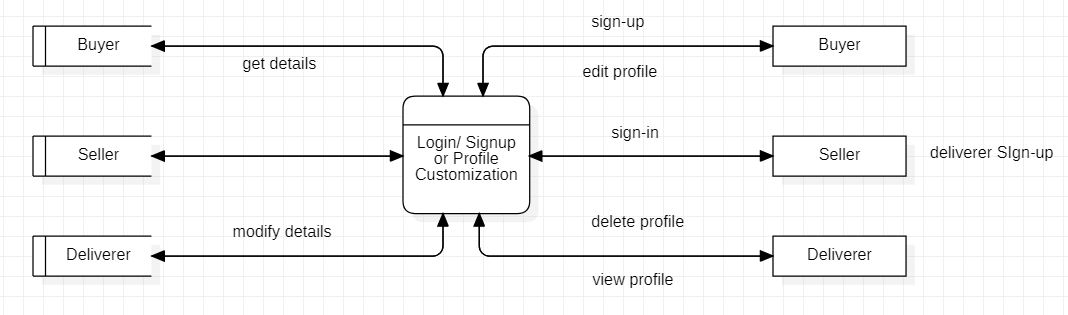
Level 2.2 :



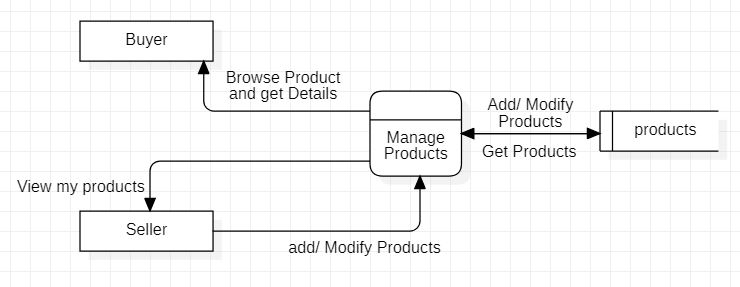
Level 2.3



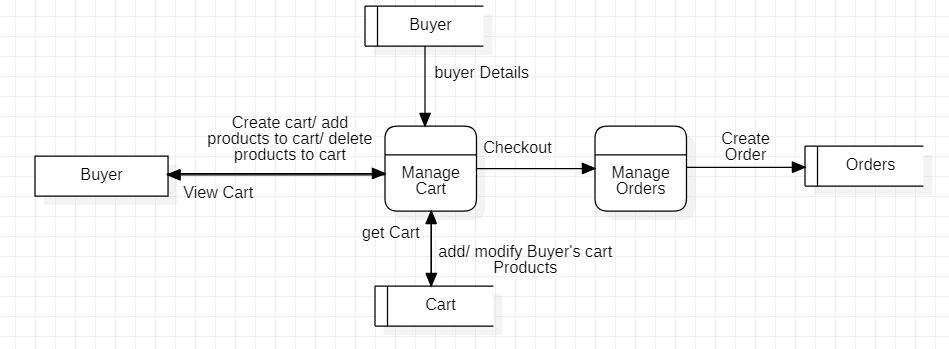
Level 2.4



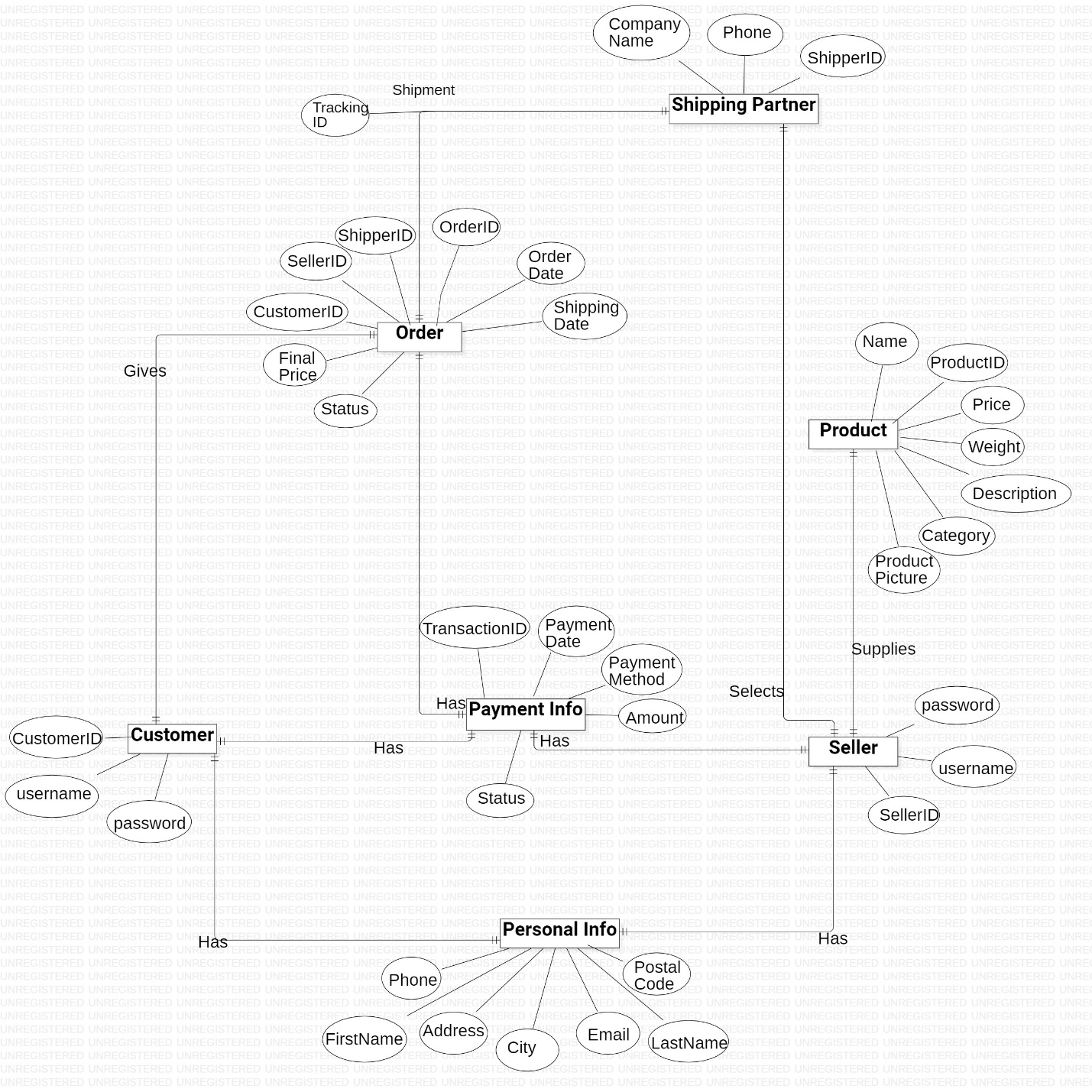
Level 2.5



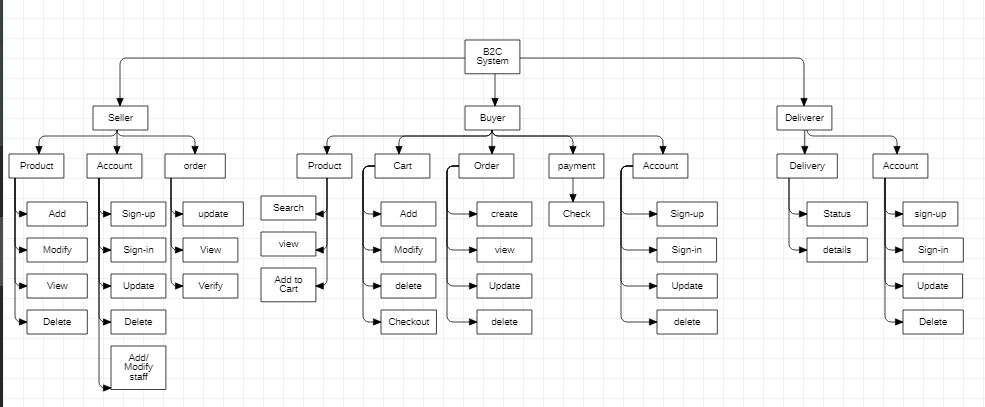
Level 2.6



ER Diagram :



Structured Diagram :



**Viva - Questions:**

Q-1. Define DFD? What are different levels of DFD?

Ans. Data flow diagrams are versatile diagramming tools. With only four symbols, data flow diagrams can represent both physical and logical information systems. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

0-level DFD:   
It is also known as a context diagram. It’s designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble

1-level DFD:   
In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system

2-level DFD:   
2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system’s functioning.

Q-2. Describe symbols used for constructing DFDs?

Ans. The four symbols used in DFD representation are data flows, data stores, processes, and sources / sinks (or external entities).

Q-3. Distinguish between a data flow diagram and a flow chart with example?

Ans. The main difference between DFD and Flowchart is that DFD is a graphical diagram that represents the data flow of a system while a flowchart is a graphical diagram that represents the sequence of steps to solve a problem.

Q-4. Explain structured chart diagram?

Ans. A Structure Chart (SC) in software engineering is a chart which shows the breakdown of a system to its lowest manageable levels.

Q-5. Describe symbols used for constructing structured chart diagram?

Ans. **Module** - It represents process or subroutine or task. A control module branches to more than one sub-module. Library Modules are re-usable and invokable from any module.

**Condition** - It is represented by small diamond at the base of module. It depicts that control module can select any of sub-routine based on some condition.

**Jump** - An arrow is shown pointing inside the module to depict that the control will jump in the middle of the sub-module.

**Loop** - A curved arrow represents loop in the module. All sub-modules covered by loop repeat execution of module.

**Data flow** - A directed arrow with empty circle at the end represents data flow.

**Control flow** - A directed arrow with filled circle at the end represents control flow.

Q-6. Explain ER diagram?

Ans. An entity relationship model*, also called an* entity-relationship (ER) diagram*, is a* graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems

Q-7. What is entity? Explain strong and weak entity?

Ans. Entity : An entity is an object or component of data. An entity is represented as rectangle in an ER diagram. The entity set which does not have sufficient attributes to form a primary key is called as Weak entity set. An entity set that has a primary key is called as Strong entity set

Q-8. What are attributes? Explain different types of attributes?

Ans. Attributes are the properties which define the entity type. They are characteristics of the entity that help users to better understand the database. Attributes are included to include details of the various entities that are highlighted in a conceptual ER diagram.

**EXPERIMENT 3**

**Aim:** To perform the user’s view analysis for the suggested system: Use case diagram.

**Description:** The use-case diagram can provide the user’s view for designing of the software product. And it can also be tested by matching up the requirements with the use-cases.

**When to Use:** Use Cases Diagrams

Use cases are used in almost every project. They are helpful in exposing requirements and Planning the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible.

**Actors---**Are NOT part of the system – they represent anyone or anything that must interact with the system.

Only input information to the system.

Only receive information from the system.

Both input to and receive information from the system.

Represented in UML as a stickman.

**Use Case**

A sequence of transactions performed by a system that yields a measurable result of values for a particular actor

A use case typically represents a major piece of functionality that is complete from beginning to end. A use case must deliver something of value to an actor

**Use Case Relationships Between actor and use case.**

Association / Communication.

Arrow can be in either or both directions; arrow indicates who initiates communication.

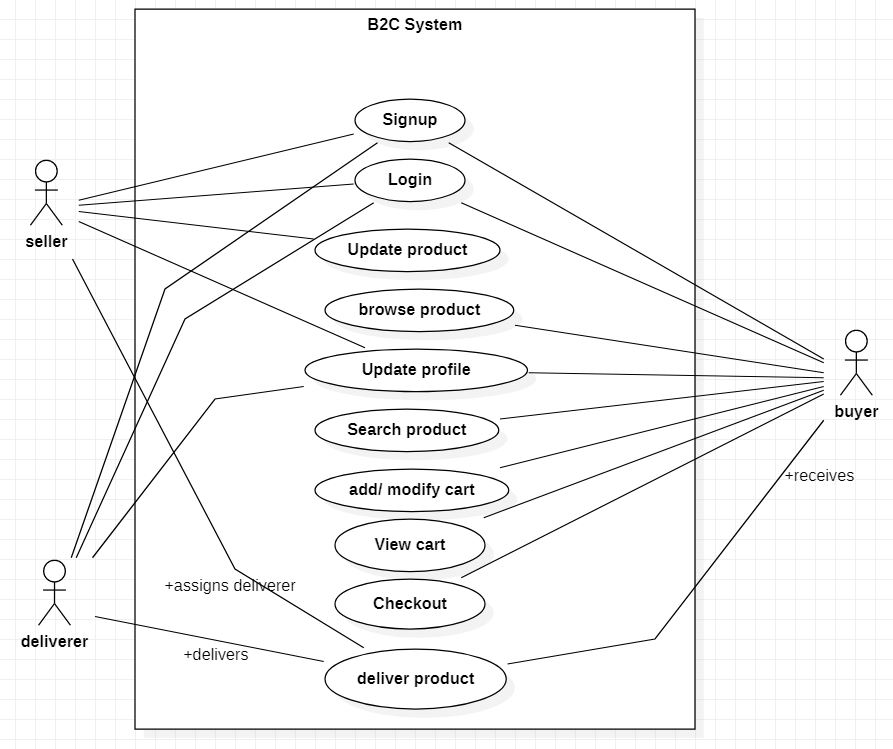
**Between use cases (generalization):**

Uses: Where multiple use cases share pieces of same functionality.

**Performance Instruction**:

1. Identify various processes, use-cases, actors etc. of the system and analyze it.
2. Use processes at various levels and draw use case diagram.

**Sample Output:**



**Viva - Questions:**

Q-1. Explain use case approach of requirement elicitation?

Ans.This technique combines text and pictures to provide a better understanding of the requirements.  The use cases describe the ‘what’, of a system and not ‘how’. Hence, they only give a functional view of the system.  The components of the use case design includes three major things – Actor, Use cases, use case diagram.

Q-2. Explain term: use-case, use-case scenarios, use-case diagrams?

Ans. A **use case** is a written description of how users will perform tasks on your website.  It outlines, from a user’s point of view, a system’s behaviour as it responds to a request.

A use case scenario is a single path through the use case. unlike a use case which is a step-by-step enumeration of the tasks carried out during a process (with the associated actors), a scenario is much more free-form.

A use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system.

Q-3. What are actors and use cases?

Ans. An **actor** in use case modelling specifies a role played by a user or any other system that interacts with the subject. An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject.

A **use case** is a written description of how users will perform tasks on your website.  It outlines, from a user’s point of view, a system’s behaviour as it responds to a request.

Q-4. Explain guidelines that should be kept in mind while creating use cases?

Ans. Consider the following:

* Single statement per line
* Always have a subject – “User” or “System”
* Be concise – remember, use cases are not end requirements – you should be demonstrating the interaction between the system and user, but not detailed specifications
* Use an active voice

Q-5. Name the person who invented use case approach?

Ans. Ivar Jacobson first formulated textual and visual modelling techniques for specifying use cases.

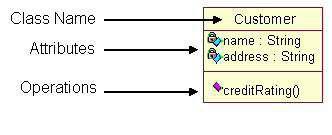
**EXPERIMENT 4**

**Aim:** To draw the structural view diagram for the system: Class diagram, object diagram.

**Description: Class diagrams** are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as

classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation.

Classes are composed of three things: a name, attributes, and operations. Below is an example of a class:



**Object diagrams** are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

**Performance Instruction:**

**To draw class diagram**

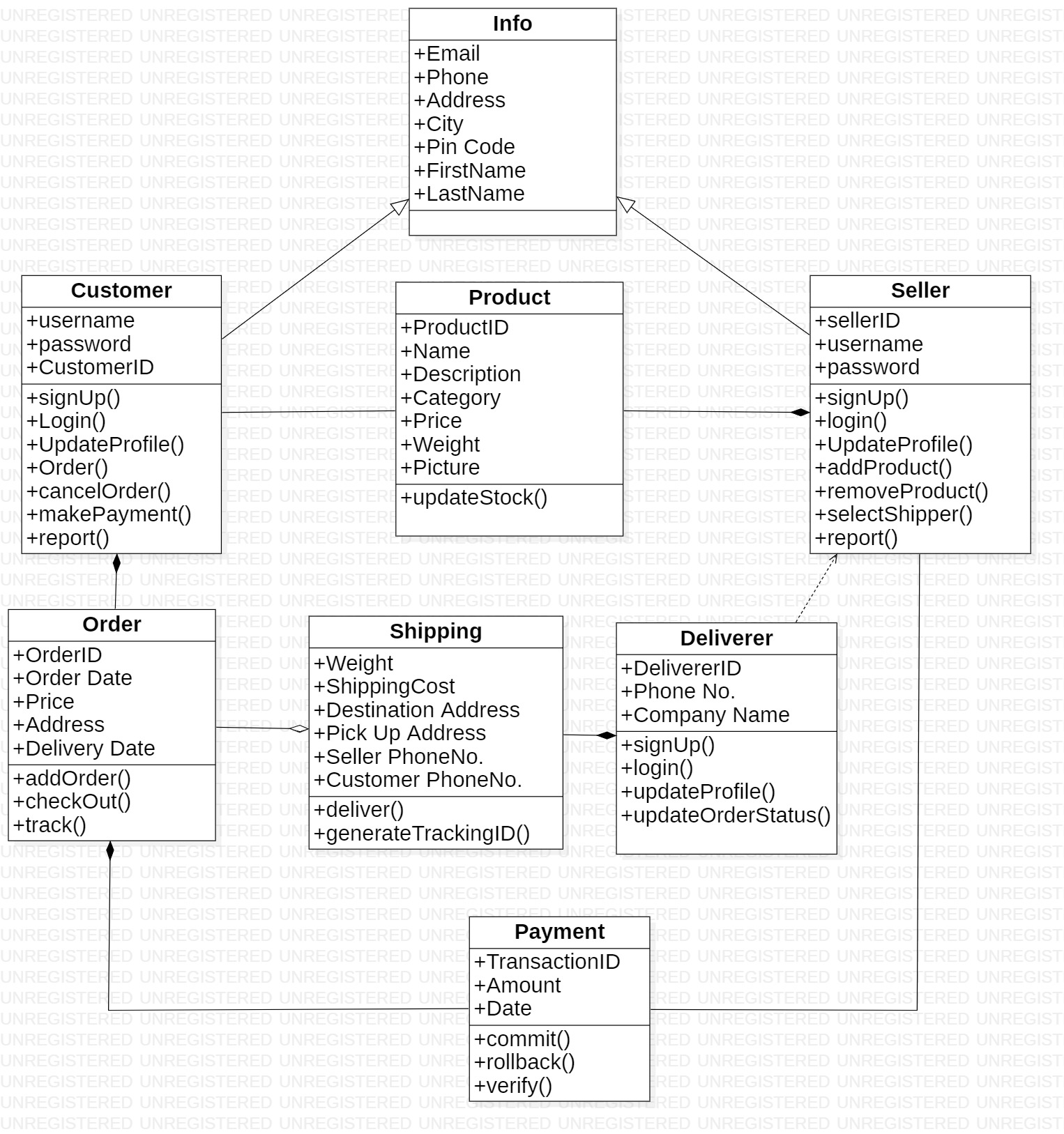
1. Identify various elements such as classes, member variables, member functions etc. of the class diagram
2. Draw the class diagram as per the norms

**To draw object diagram**

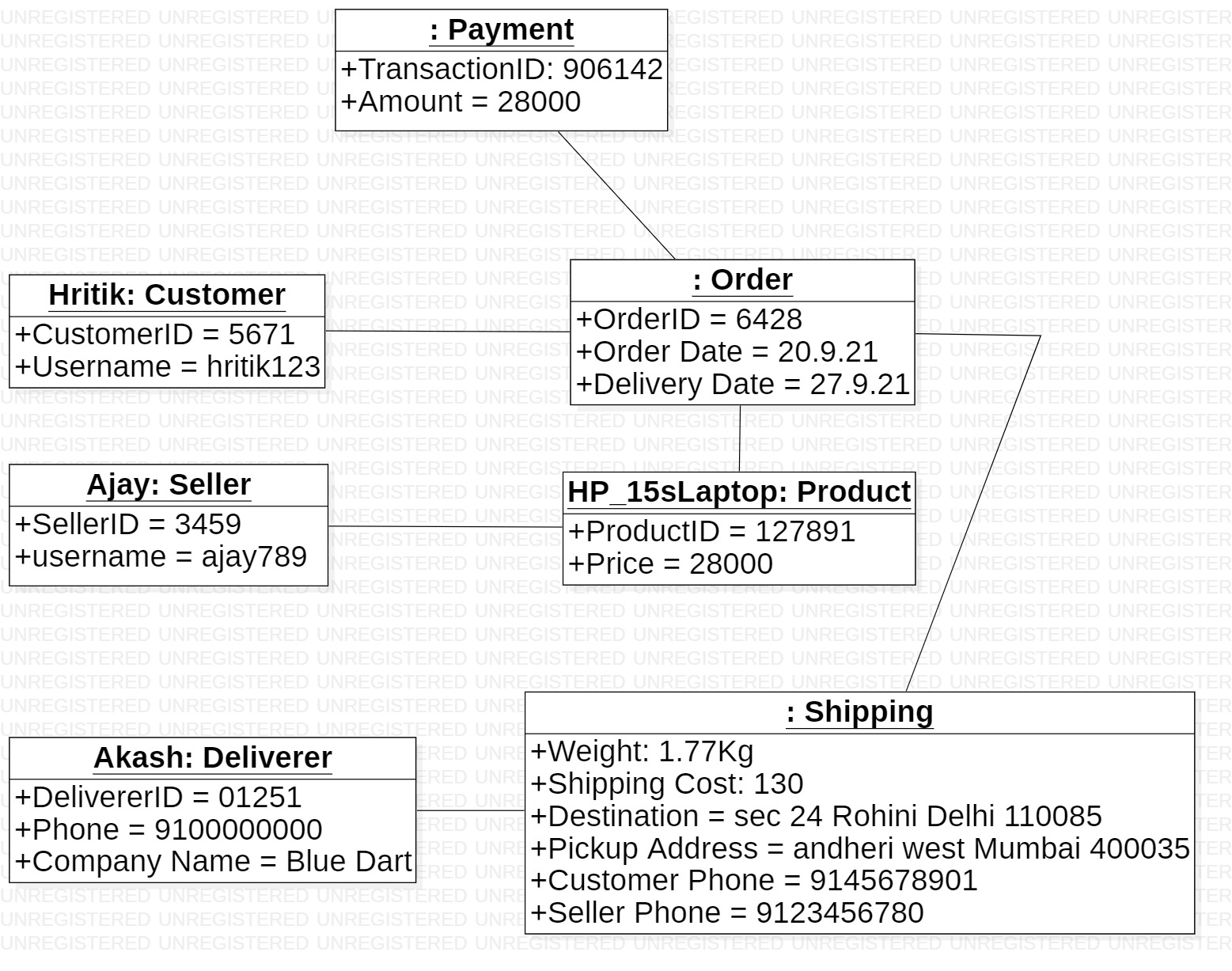
1. First, analyze the system and decide which instances have important data and association.
2. Second, consider only those instances, which will cover the functionality.
3. Third, make some optimization as the number of instances are unlimited.

**Sample Output:**

**Class Diagram :**



**Object Diagram :**

****

**Viva - Questions:**

Q-1. Explain class diagram?

Ans. A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Q-2. Explain four types of relationship used in class diagram?

**Ans.** **Inheritance (or Generalization):**

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier.

**Association**

Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes. Associations are typically named using a verb or verb phrase which reflects the real world problem domain.

**Realization**

Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class.

**Dependency**

An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modelled as a dependency relationship.

Q-3. Explain terms classes, interfaces, collaborations and dependency?

Ans.

Classes

A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations

Interfaces

Interfaces are model elements that define sets of operations that other model elements, such as classes, or components must implement. An implementing model element realizes an interface by overriding each of the operations that the interface declares.

**Collaborations**

It 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. An object consists of several features.

**Dependency**

An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modeled as a dependency relationship.

Q-4. Explain object diagram?

Ans. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.