**DEPARTMENT OF INFORMATION TECHNOLOGY FACULTY OF  ENGINEERING & TECHNOLOGY**

**Mini Project**

**SUBJECT TITLE: OBJECT ORIENTED ANALYSIS AND DESIGN  SUBJECT CODE:**

**18CSC202J**

**Project Title:**

**Stock Management System Diagram**

**PALLAPOLU NIRANJAN REDDY**

**RA2111030010177**

**Department of NETWORKING & COMMUNICATIONS**

****

**SRM University, SRM Nagar, Kattankulathur-603203 Kancheepuram District,  Tamil Nadu**

# UML REPORT ON STOCK MAINTENANCE

**Date:**

**AIM:**

To create a system to perform the Stock maintenance

# (I) PROBLEM STATEMENT

The stock maintenance system must take care of sales information of the company and must analyze the potential of the trade. It maintains the number of items that are added or removed. The sales person initiates this Use case. The sales person is allowed to update information and view the database.

# (II) SOFTWARE REQUIREMENT SPECIFICATION

## 1.0 INTRODUCTION

Stock maintenance is an interface between the customer and the sales person. It aims at improving the efficiency in maintaining the stocks.

## 1.1 PURPOSE

The entire process of Stock maintenance is done in a manual manner Considering the fact that the number of customers for purchase is increasing every year, a maintenance system is essential to meet the demand. So this system uses several programming and database techniques to elucidate the work involved in this process.

## 1.2 SCOPE

* The System provides an interface to the customer where they can fill in orders for the item needed.
* The sales person is concerned with the issue of items and can use this system.
* Provide a communication platform between the customer and the sales person.

## 1.3 DEFINITIONS, ACRONYMS AND THE ABBREVIATIONS

* **Admin**: One who analyze the product and distribute the news.
* **Customer:** One who takes order of product
* **Supplier:** One who maintains the stock details
  1. **REFERENCES**

UML Diagram Formats.

* 1. **TECHNOLOGIES TO BE USED**
* Visual Studio Code

**1.6 TOOLS TO BE USED**

* Star UML Tool (for developing UML Patterns)

## 1.7 OVERVIEW

SRS includes two sections overall description and specific requirements

**Overall Description** will describe major role of the system components and inter-

Connections

**Specific Requirements** will describe roles & functions of the actors.

## 2.0 OVERALL DESCRIPTION 2.1 PRODUCT PERSPECTIVE

The Stock maintenance acts as an interface between the 'customer' and the 'sales person'. This system tries to make the interface as simple as possible and at the same time not risking the work of data stored in

## 2.2 SYSTEM FUNCTIONS

* Secure order of information by the customer
* Schedule the customer an appointment for manual delivery of the product.

## 2.3 USER CHARACTERISTICS

1. **Customer:** The person who orders for the item.
2. **Validate customer:** The items ordered by the customer are validated.
3. **Sales Detail:** Maintains the stock details after delivering the items to the customer.

## 2.4 CONSTRAINTS

1. The customer should wait until the trade contractor and other to analyze the product.
2. After the distribution of the news about the product. The customer can take order and request of sales person to fill it.
3. Finally the sales person delivers the order.

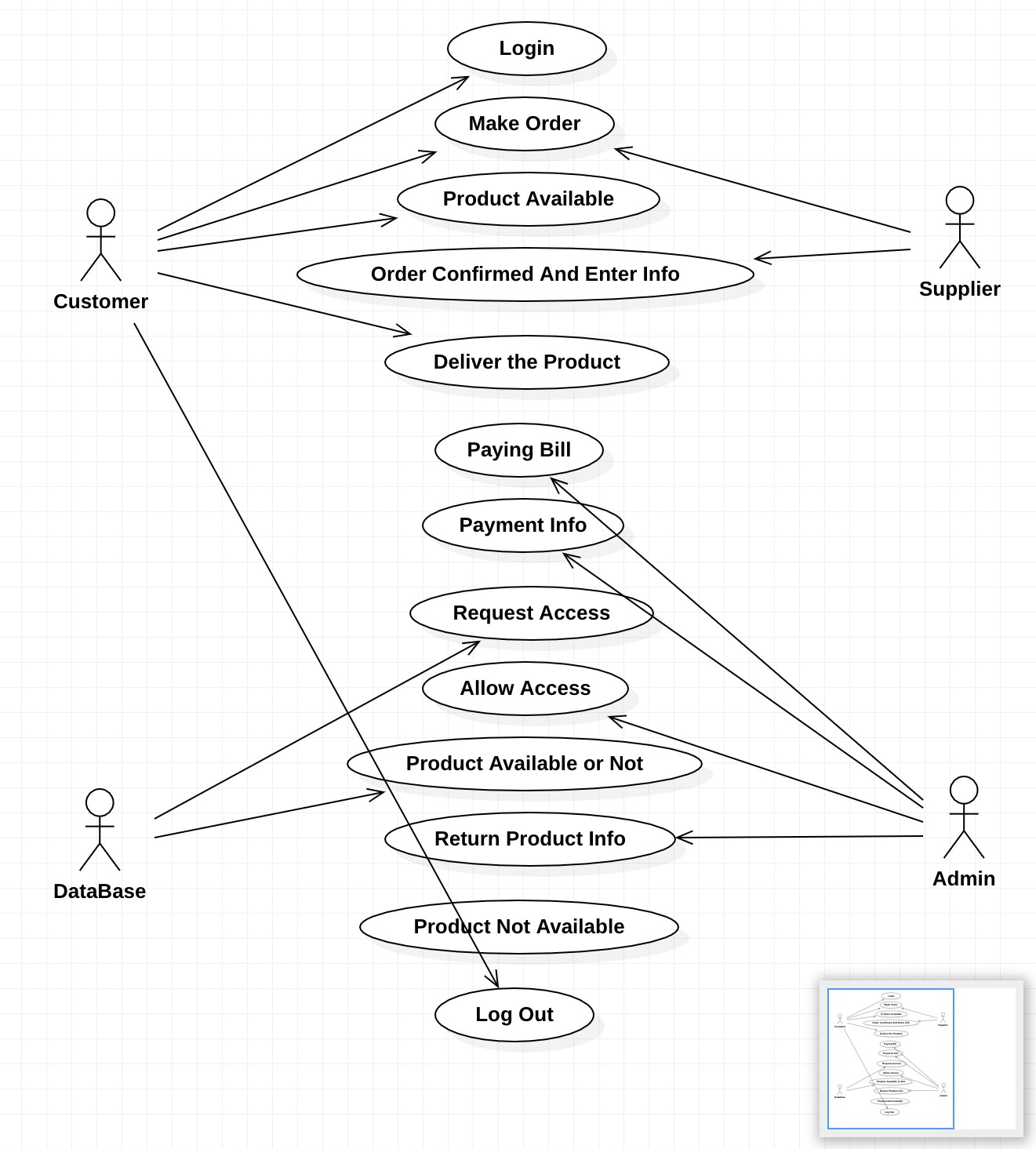
# (III) USE CASE DIAGRAM

The functionality of a system can be described in a number of different use-cases, each of which represents a specific flow of events in a system. It is a graph of actors, a set of use-cases enclosed in a boundary, communication, associations between the actors and the use-cases, and generalization among the use-cases.

# ACTORS

The actors used in this system are

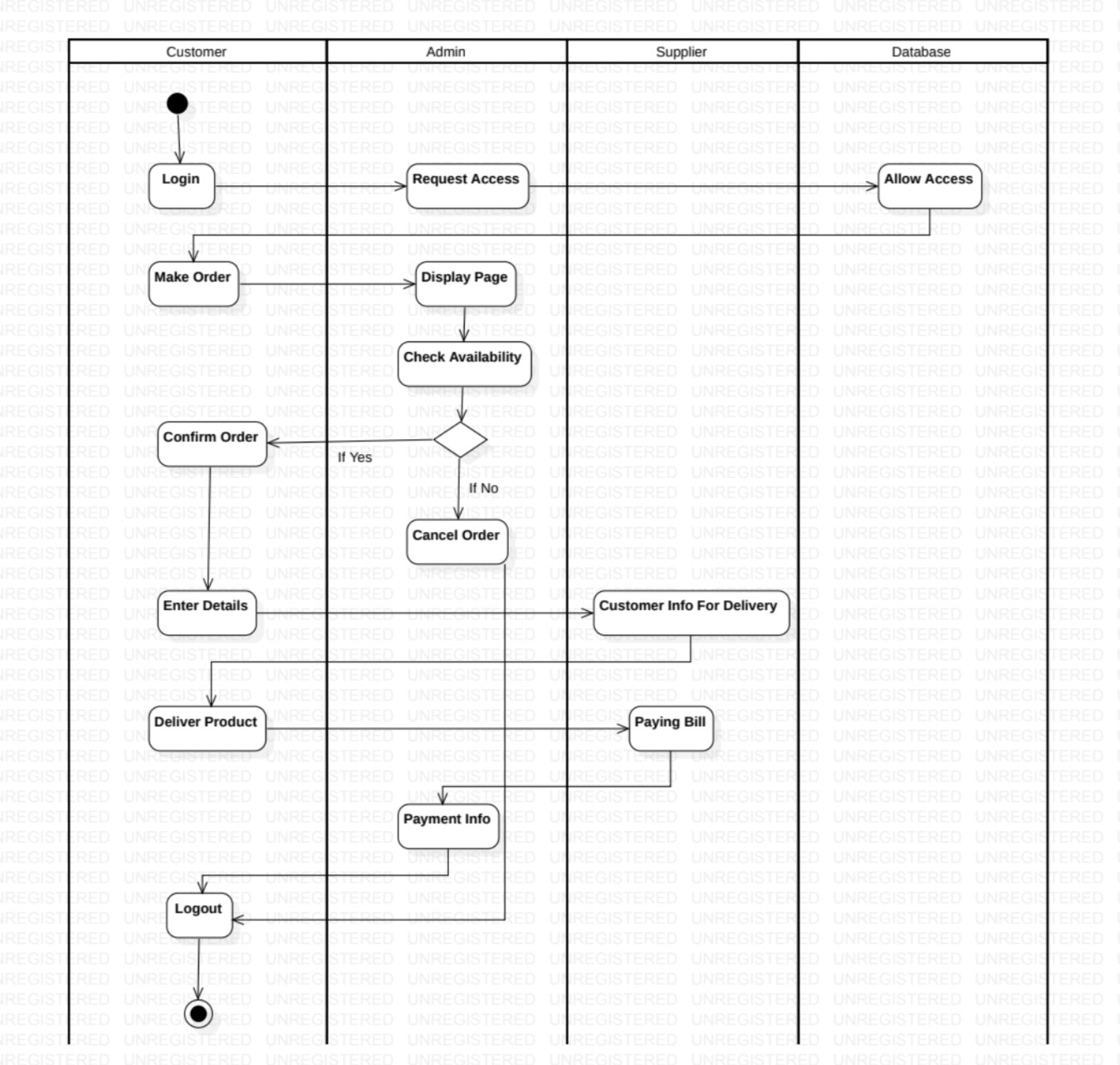
1. **Customer:** The person who orders for the item.
2. **Supplier:** The items ordered by the customer are validated.
3. **Admin:** Maintains the stock details after delivering the items to the customer.
4. **Database:** Maintains all the Records.



**USECASE DIAGRAM**

# (IV) ACTIVITY DIAGRAM

It shows organization and their dependence among the set of components. These diagrams are particularly useful in connection with workflow and in describing behavior that has a lot of parallel processing. An activity is a state of doing something: either a real-world process, or the execution of a software routine.



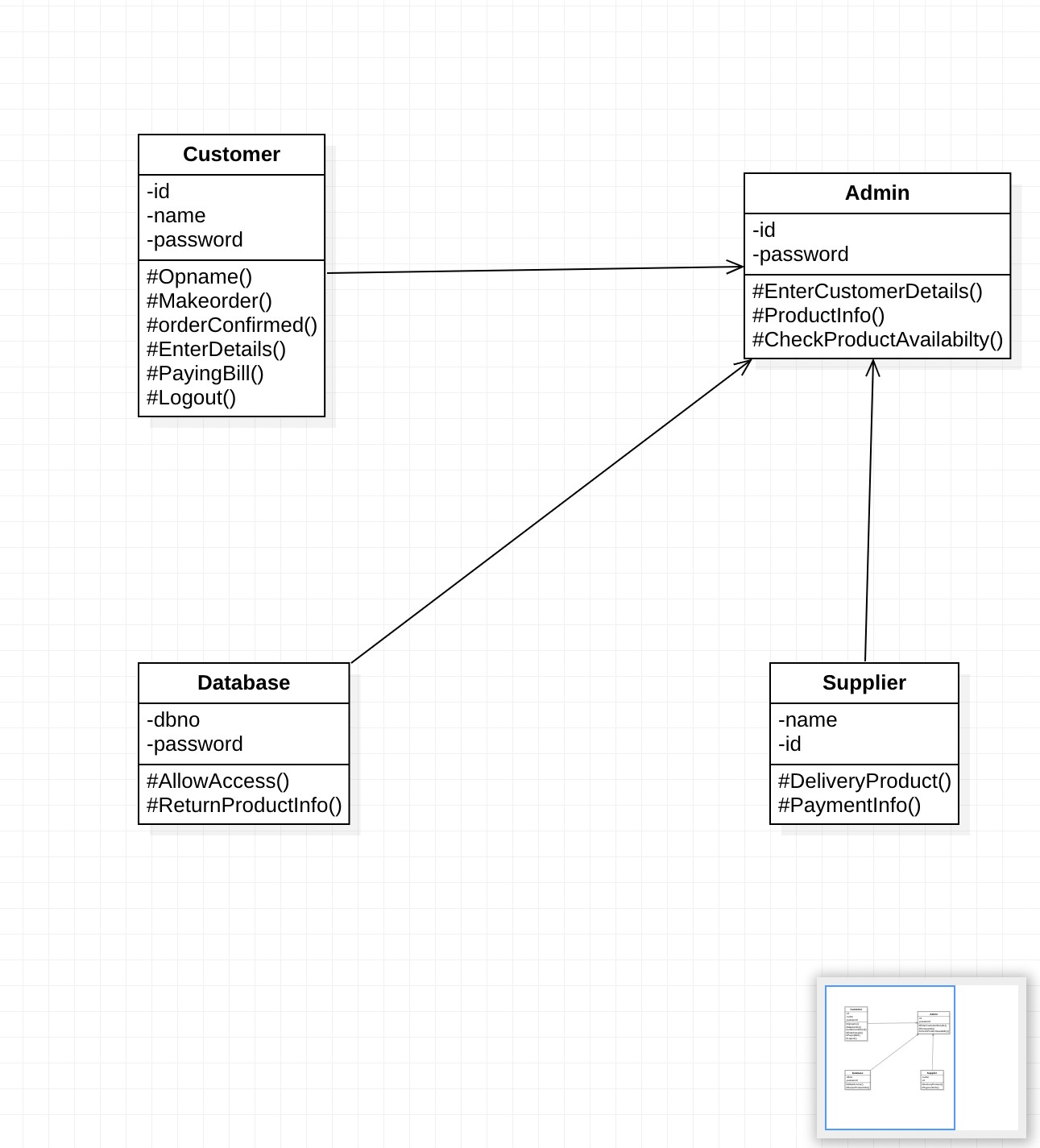
# ACTIVITY DIAGRAM

**(V) CLASS DIAGRAM**

**Description:**

* + A class diagram describes the type of objects in system and various kinds of relationships that exists among them.
  + Class diagrams and collaboration diagrams are alternate representations of object models.

1. **PurchaseDetails:** One who takes orders for the product?
2. **SalesDetails:** The customer make an order for the required products.
3. **Product Details:** The items that are stored as stock.



**CLASS DIAGRAM**

# (VI) UML INTERACTION DIAGRAMS

It is the combination of sequence and collaboration diagram. It is used to depict the flow of events in the system over a timeline. The interaction diagram is a dynamic model which shows how the system behaves during dynamic execution.

# SEQUENCE DIAGRAM

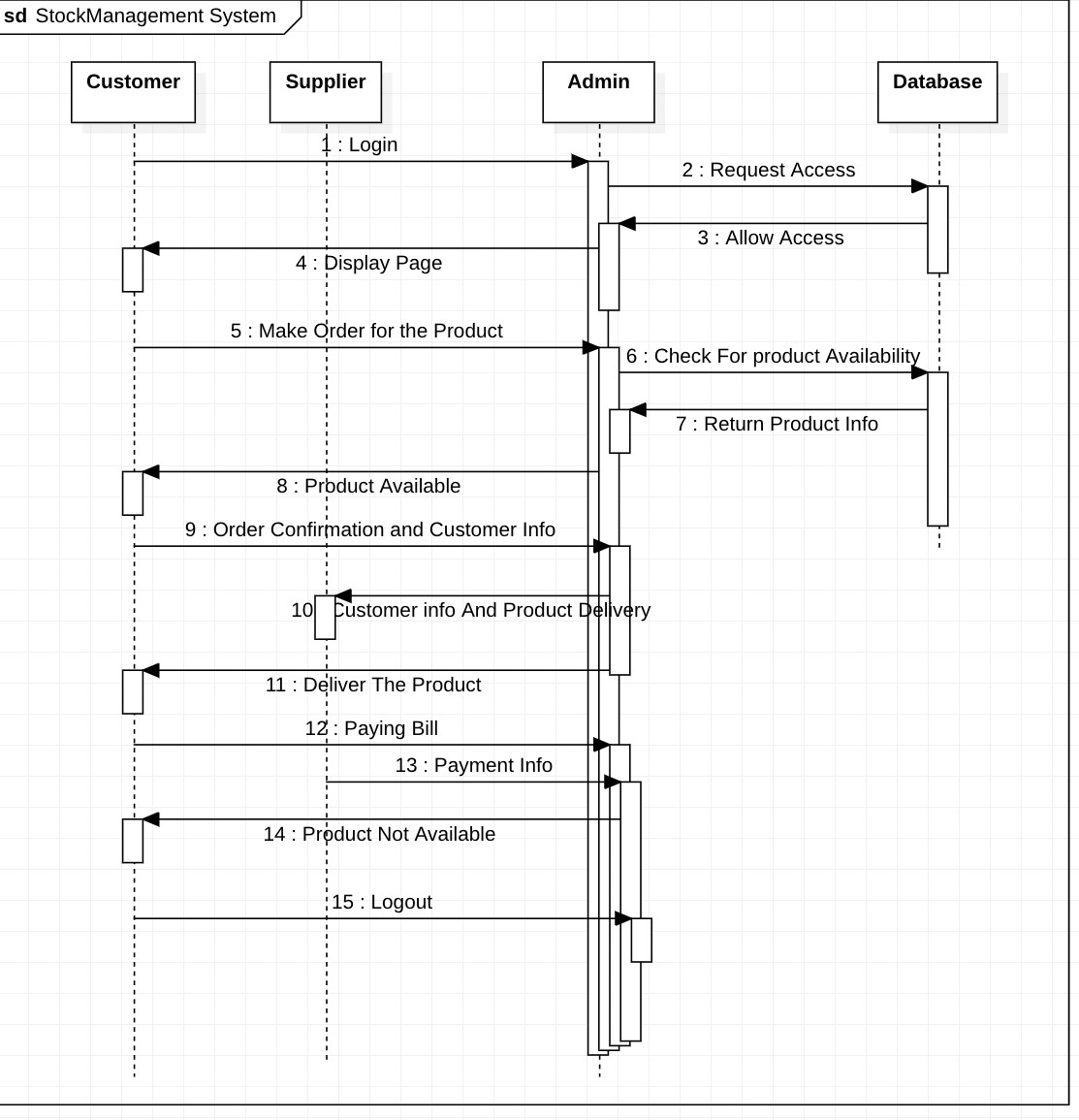
• A sequence diagram represents the sequence and interactions of a given USECASE or scenario. Sequence diagrams can capture most of the information about the system. Most object to object interactions and operations are

considered events and events include signals, inputs, decisions, interrupts, transitions and actions to or from users or external devices.

o. An event also is considered to be any action by an object that sends information.

The event line represents a message from one object to another, in which the “from” object is requesting an operation be performed by the “to” object. The “to” object performs the operation using a method that the class contains.

o. It is also represented by the order in which things occur and how the objects in the system send message to one another.



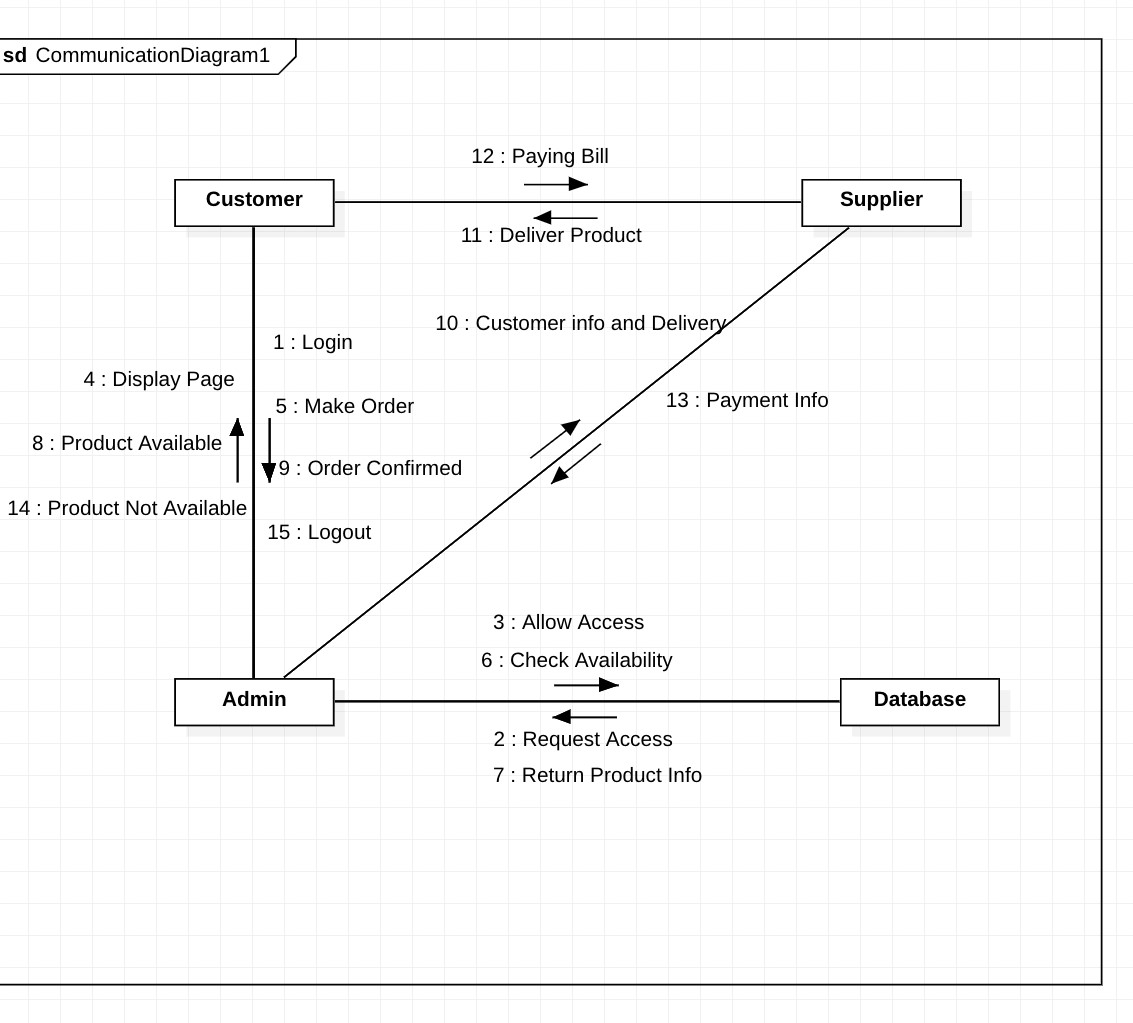
**SEQUENCE DIAGRAM**

# COLLABORATION DIAGRAM

Collaboration diagram and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. Collaboration diagram shows object s, their links and their messages. They can also contain simple class instances and class utility instances.

During, analysis indicates the semantics of the primary and secondary interactions.

Design, shows the semantics of mechanisms in the logical design of system.



# COLLABORATION DIAGRAM

3**. STATE CHART DIAGRAM:**

Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. Statechart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc. State chart diagram is used to visualize the reaction of a system by internal/external factors.

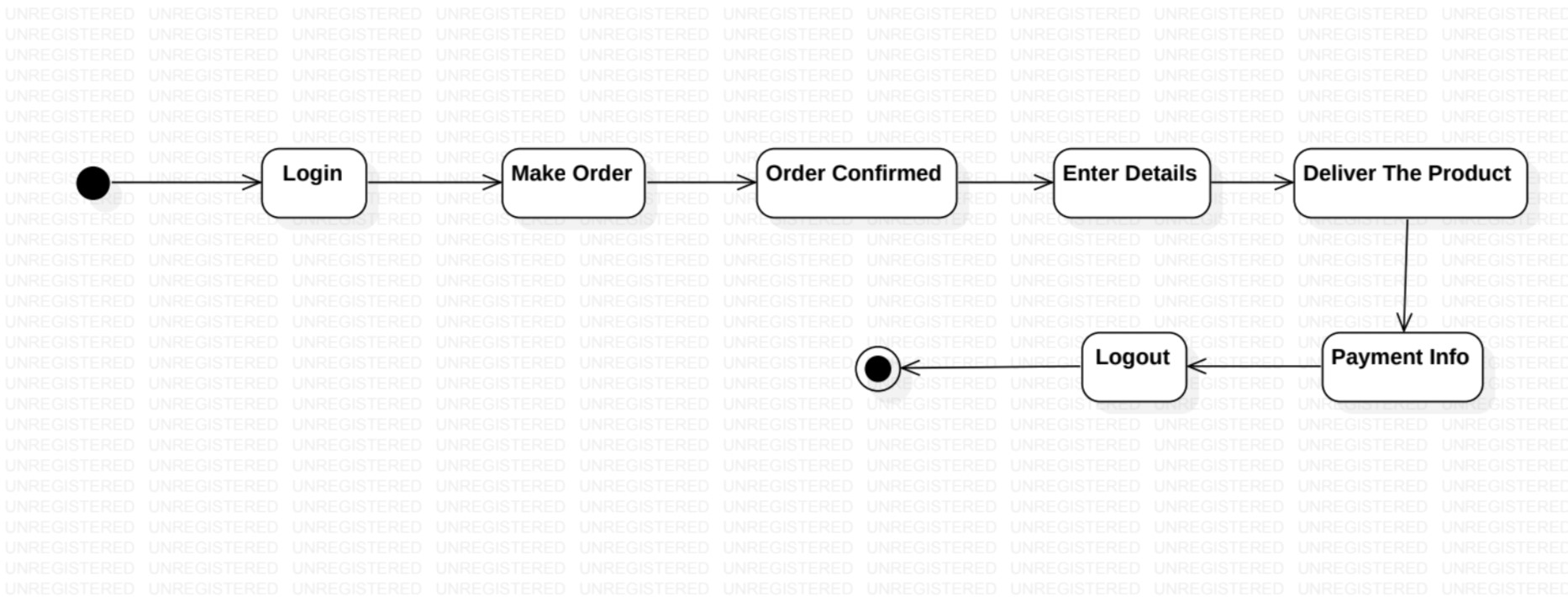
Following are the main purposes of using Statechart diagrams:

To model dynamic aspect of a system.

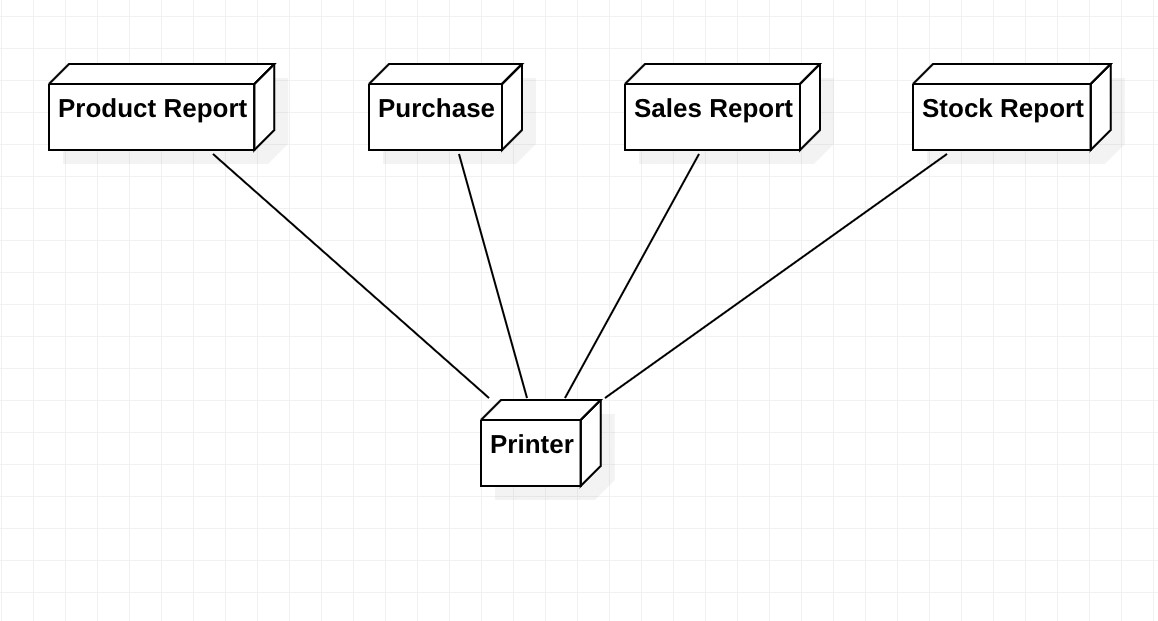
To model life time of a reactive system.

To describe different states of an object during its life time.

Define a state machine to model states of an object.



# (VII) DEPLOYMENT DIAGRAM



# DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed. Deployment diagrams are used for visualizing deployment view of a system. This is generally used by the deployment team. Note: If the above descriptions and usages are observed carefully then it is very clear that all the diagrams are having some relationship with one another. Component diagrams are dependent upon the classes, interfaces etc which are part of class/object diagram. Again the deployment diagram is dependent upon the components which are used to make a component diagrams.

The purpose of deployment diagrams can be described as:

Visualize hardware topology of a system.

Describe the hardware components used to deploy software components.

Describe runtime processing nodes.

**(VIII) COMPONENT DIAGRAM:**

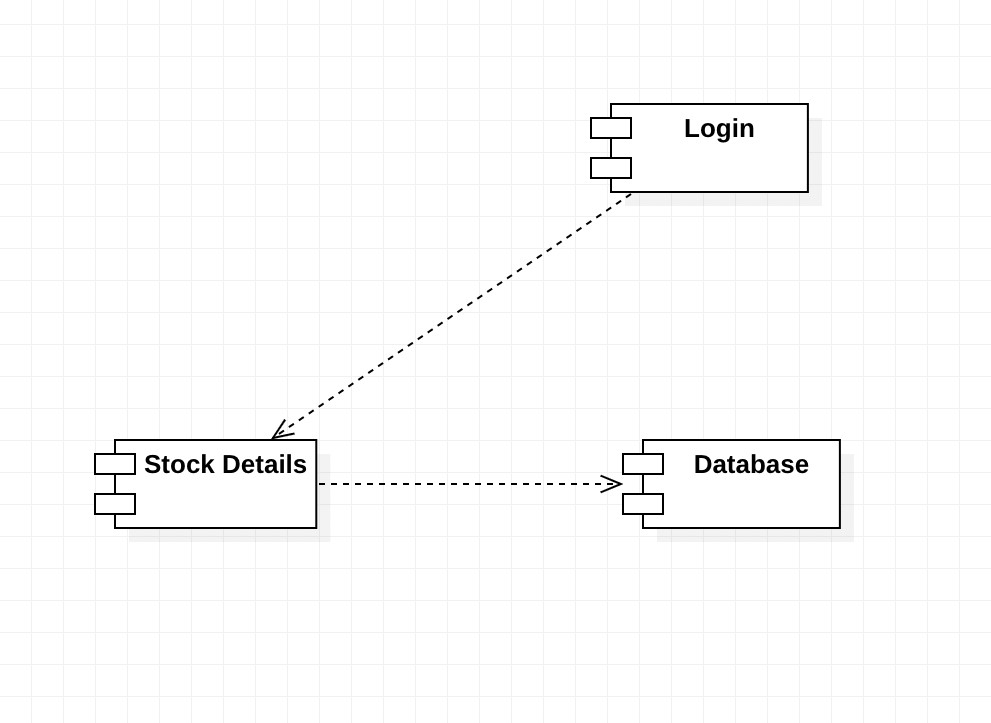
Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces or collaborations. So Component diagrams represent the implementation view of a system. During design phase software artifacts (classes, interfaces etc) of a system are arranged in different groups depending upon their relationship. Now these groups are known as components. Finally, component diagrams are used to visualize the implementation.

The purpose of the component diagram can be summarized as:

Visualize the components of a system.

Construct executables by using forward and reverse engineering.

Describe the organization and relationships of the components.



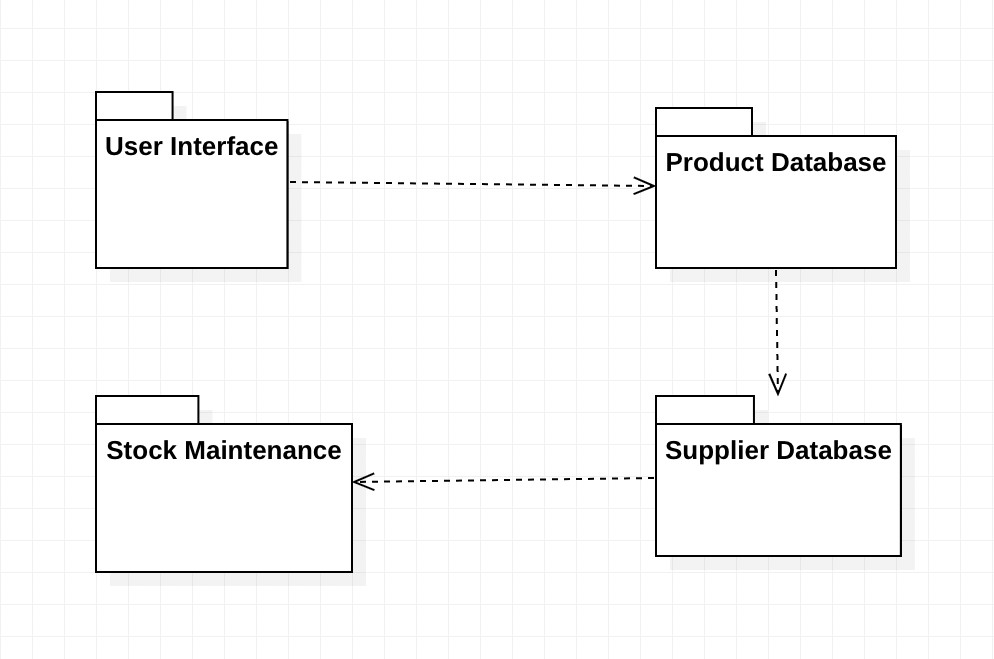
# COMPONENT DIAGRAM

**(IX) PACKAGE DIAGRAM:**

Package diagrams are structural diagrams used to show the organization and arrangement of various model elements in the form of packages. A package is a grouping of relate UML Diagrams, such as diagrams, documents, classes, or even other packages. Each element is nested within the package, which is depicted as a file folder within the diagram, then arranged hierarchically within the diagram. Package diagrams are most commonly used to provide a visual organization of the layered architecture within any UML classifier, such as a software system.

A well-designed package diagram provides numerous benefits to those looking to create a visualization of their UML system or project.

* They provide a clear view of the hierarchical structure of the various UML elements within a given system.
* These diagrams can simplify complex class diagrams into well-ordered visuals.
* They offer valuable high-level visibility into large-scale projects and systems.
* Package diagrams can be used to visually clarify a wide variety of projects and systems.
* These visuals can be easily updated assystems and projects evolve.



# PACKAGE DIAGRAM

**(VIII) C++ CODE GENERATION FOR STOCK MAINTENANCE SYSTEM:**

/\*\*

* *STOCK MAINTENANCE SYSTEM*

\*/

#*include* "Admin.h"

/\*\*

* *Admin implementation*

## \*/

void Admin::*EnterCustomerDetails*() {

}

void Admin::*ProductInfo*() {

}

void Admin::*CheckProductAvailabilty*() {

}

/\*\*

* *STOCK MAINTENANCE SYSTEM*

\*/

#*include* "Customer.h"

/\*\*

* *Customer implementation*

## \*/

void Customer::*Opname*() {

}

void Customer::*Makeorder*() {

}

void Customer::*orderConfirmed*() {

}

void Customer::*EnterDetails*() {

}

void Customer::*PayingBill*() {

}

void Customer::*Logout*() {

}

|  |  |
| --- | --- |
| /\*\* |  |
| *\* STOCK MAINTENANCE SYSTEM* | |

\*/

#*include* "Database.h"

/\*\*

*\* Database implementation*

## \*/

void Database::*AllowAccess*() {} void Database::*ReturnProductInfo*() {

}

|  |  |
| --- | --- |
| /\*\* |  |
| *\* STOCK MAINTENANCE SYSTEM* | |

\*/ #*include* "Supplier.h" /\*\*

*\* Supplier implementation*

## \*/

void Supplier::*DeliveryProduct*() {

}

void Supplier::*PaymentInfo*() { }

**RESULT:**

Thus the mini project for stock maintenance system has been successfully executed and codes are generated.