SMART FARM MONITORING SYSTEM

PROJECT REPORT

18CSC202J/ 18AIC203J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

II Year/ III Semester

Academic Year: 2022 -2023

By

TUSHI MITTAL (RA211103010601)

KHUSHI ATREY (RA211103010610)

Under the guidance of

Dr. A. JEYASEKAR

Associate Professor

Department of Computing Technologies

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled "**ONLINE PAYMENT SYSTEM"** is the bonafide work of **TUSHI MITTAL (RA211103010601) KHUSHI ATREY (RA211103010610)** who undertook the task of completing the project within the allotted time.

Signature of the Guide

Dr. A. Jeyasekar

Associate Professor

Department of CTech,

SRM Institute of Science and Technology

Signature of the II Year Academic Advisor

Professor and Head

Department of CTech

SRM Institute of Science and Technology

Content Page

ABSTRACT

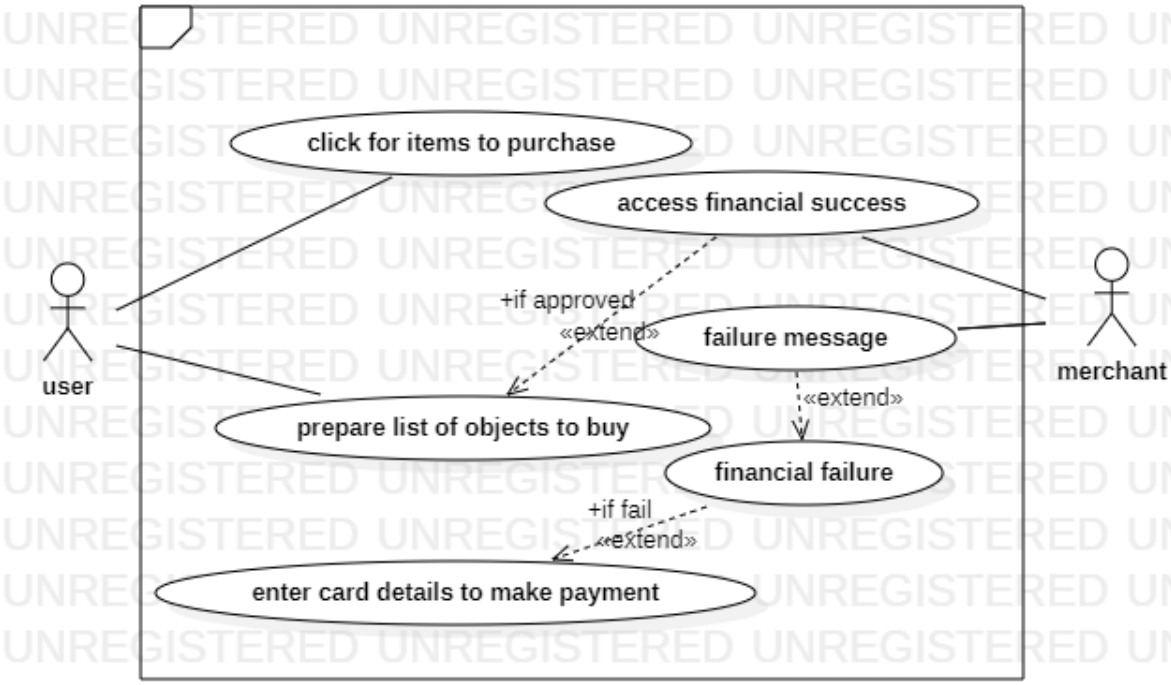
Online payment system deals with the bank, customers and the merchants basically. In the modern world where everything is possible online, payment system is made easy by presenting it in uml form. Online payment has several features you can use UPI, card, wallets, or net banking. In our system we have provided all the options user can choose the desired method and then do the payment by providing bank details, the system will verify the details and account balance with the bank and then continue with the payment process after once the bank approves.

Cash is very hefty to carry and online payment system makes user life easier the payments are directly from one bank account to another. So it is very conveniently presented in this project report how do online payment system works using several uml diagrams.

USE CASE DIAGRAM

A cornerstone part of the system is the functional requirements that the system fulfills. Use Case diagrams are used to analyze the system's high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram:

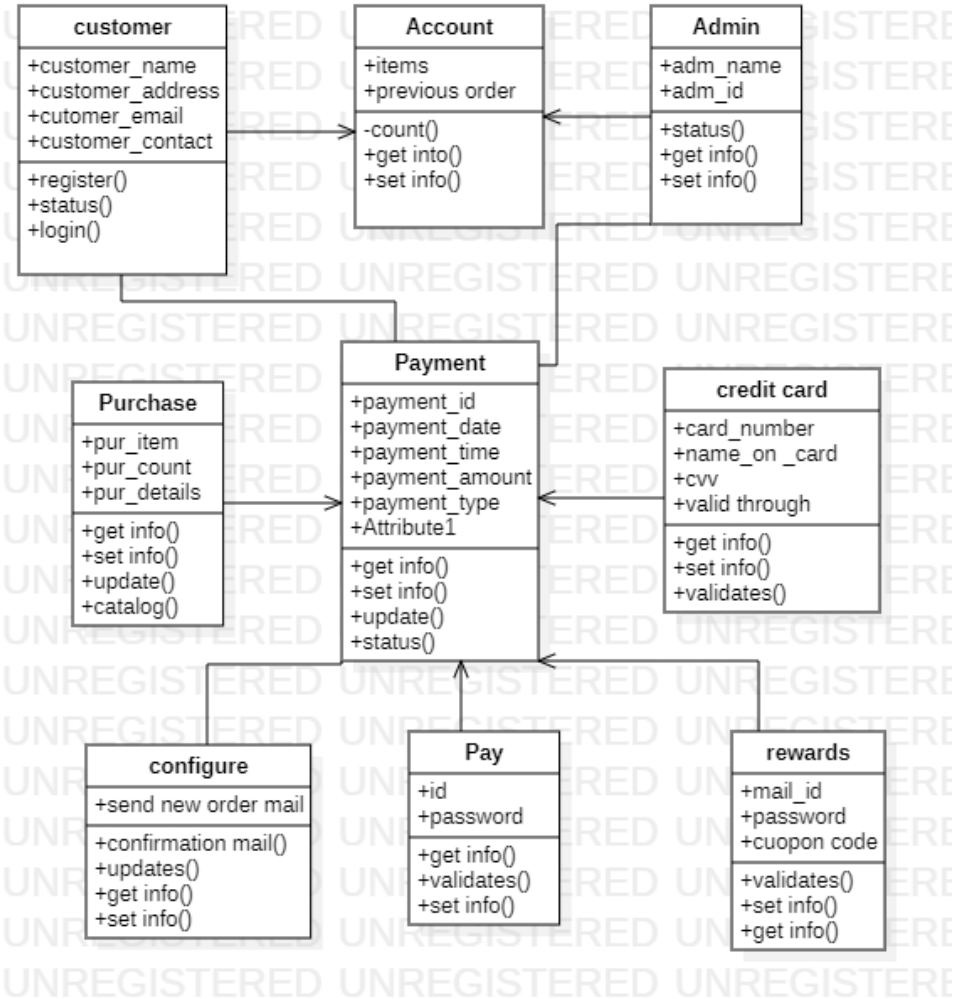
- **Functional requirements** – represented as use cases; a verb describing an action
- **Actors** – they interact with the system; an actor can be a human being, an organization or an internal or external application
- **Relationships** between actors and use cases – represented using straight arrows



CLASS DIAGRAM

Class UML diagram is the most common diagram type for software documentation. Since most software being created nowadays is still based on the Object-Oriented Programming paradigm, using class diagrams to document the software turns out to be a common-sense solution. This happens because OOP is based on classes and the relations between them.

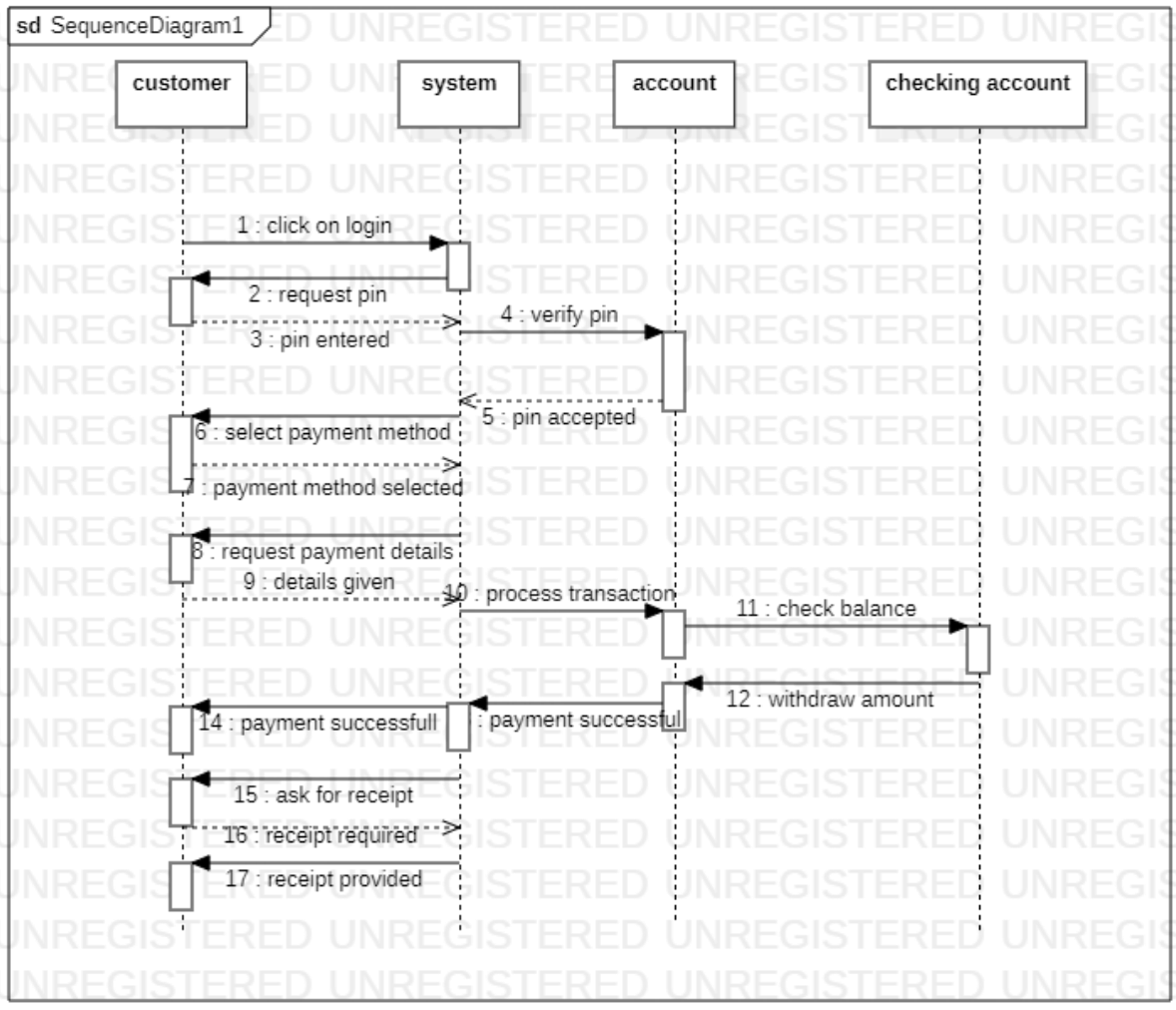
In a nutshell, class diagrams contain classes, alongside with their attributes (also referred to as data fields) and their behaviors (also referred to as member functions). More specifically, each class has 3 fields: the class name at the top, the class attributes right below the name, the class operations/behaviors at the bottom. The relation between different classes (represented by a connecting line), makes up a class diagram.



SEQUENCE DIAGRAM

Sequence diagrams are probably the most important UML diagrams among not only the computer science community but also as design-level models for business application development. Lately, they have become popular in depicting business processes, because of their visually self-explanatory nature.

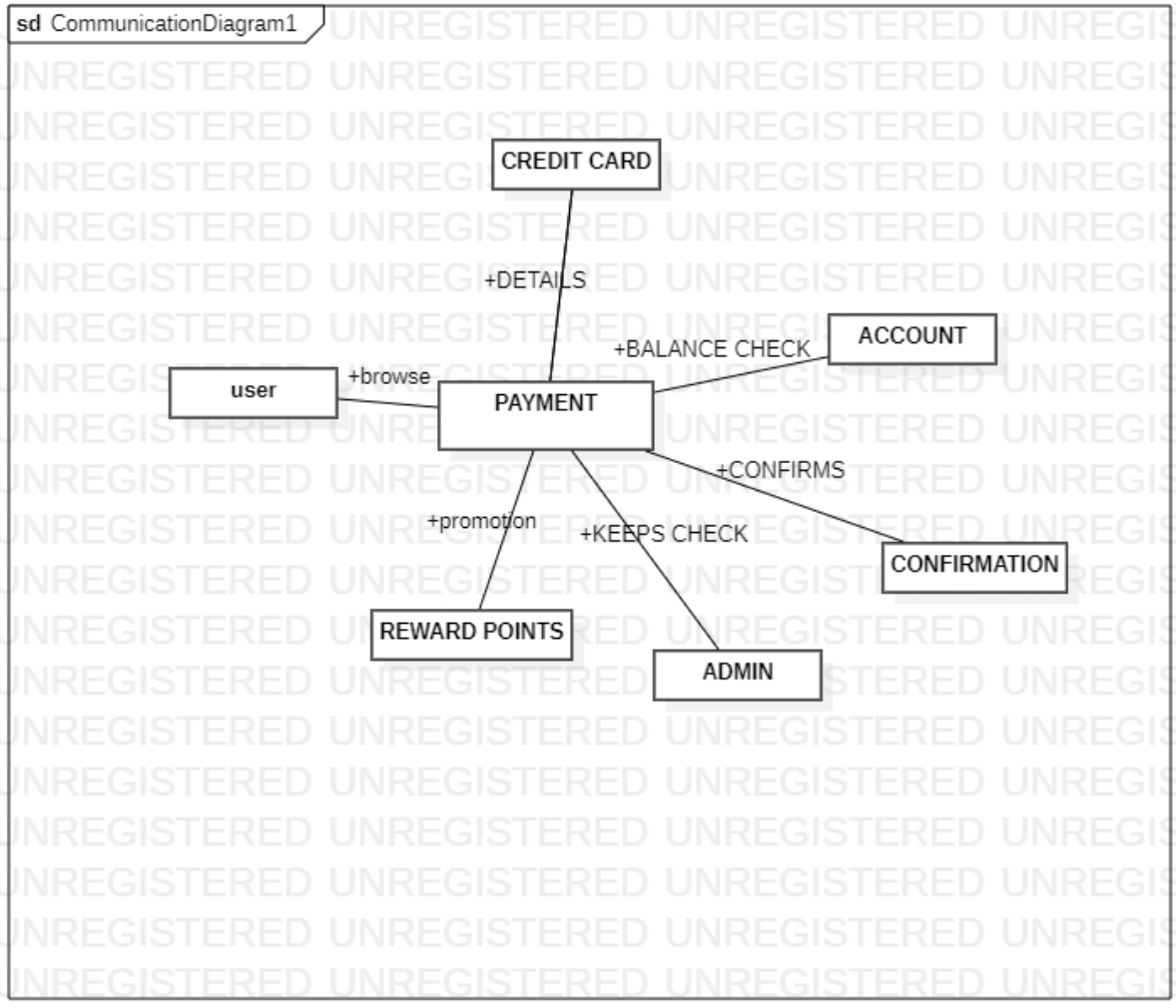
As the name suggests, sequence diagrams describe the sequence of messages and interactions that happen between actors and objects. Actors or objects can be active only when needed or when another object wants to communicate with them. All communication is represented in a chronological manner.



COMMUNICATION DIAGRAM

Since the core components are the messages that are exchanged between objects, we can build communication diagrams the same way we would make a sequence diagram. The only difference between the two is that objects in communication diagrams are shown with association connections.

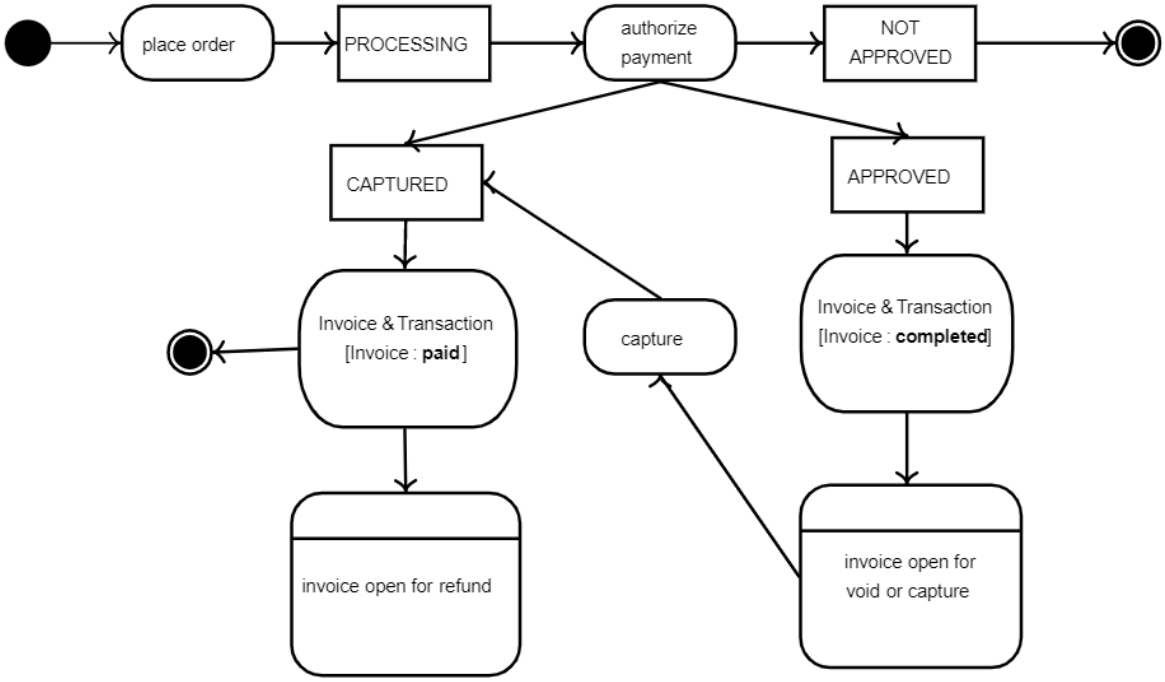
Visually, the two differ in that sequence diagrams are well-structured vertically and the message flow follows a top-down chronological approach. Communication UML diagrams on the other hand use number schemes and pointing arrows in order to depict the message flow



STATE CHART DIAGRAM

It takes the name state machine because the diagram is essentially a machine that describes the several states of an object and how it changes based on internal and external events.

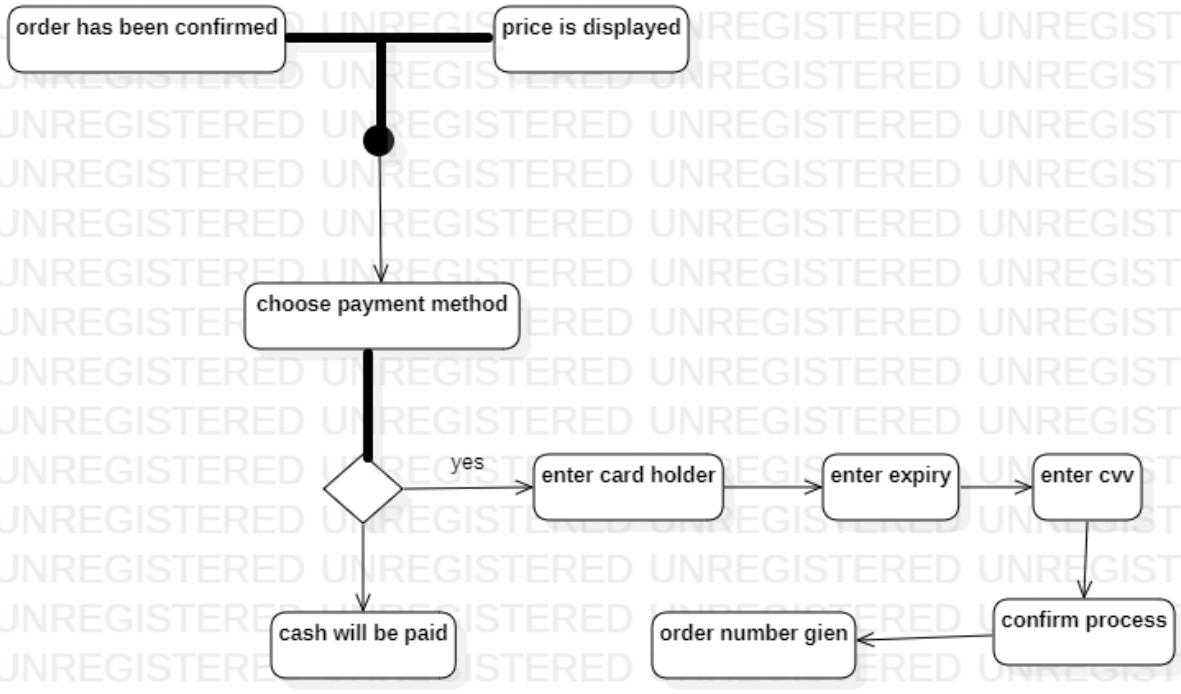
A very simple state machine diagram would be that of a chess game. A typical chess game consists of moves made by White and moves made by Black. White gets to have the first move and thus initiates the game. The conclusion of the game can occur regardless of whether it is the White's turn or the Black's. The game can end with a checkmate, resignation or in a draw (different states of the machine).



ACTIVITY DIAGRAM

Activity diagrams are probably the most important UML diagrams for doing business process modeling. In software development, it is generally used to describe the flow of different activities and actions. These can be both sequential and in parallel. They describe the objects used, consumed or produced by an activity and the relationship between the different activities. All the above are essential in business process modeling.

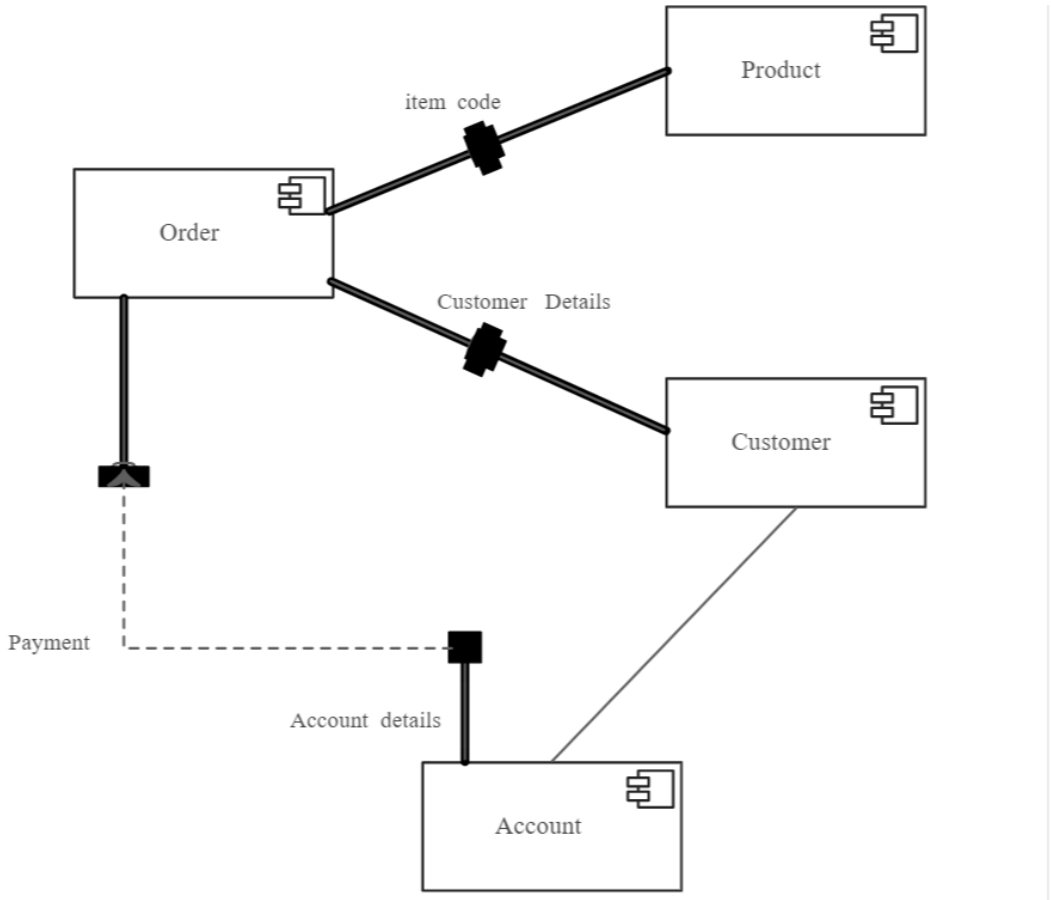
A process is not focused on what is being produced but rather on the set of activities that lead to one the other and how they are interconnected, with a clear beginning and end. The example above depicts the set of activities that take place in a content publishing process. In a business environment, this is also referred to as business process mapping or business process modeling.



COMPONENT DIAGRAM

When dealing with documentation of complex systems, component UML diagrams can help break down the system into smaller components. Sometimes it is hard to depict the architecture of a system because it might encompass several departments or it might employ different technologies.

For example, Lambda architecture is the typical example of a complex architecture that can be represented using a component UML diagram. Lambda architecture is a data-processing architecture employed by several companies for storing and processing data in a distributed system. It is made up of three different layers: the speed layer, the batch layer and the serving one.



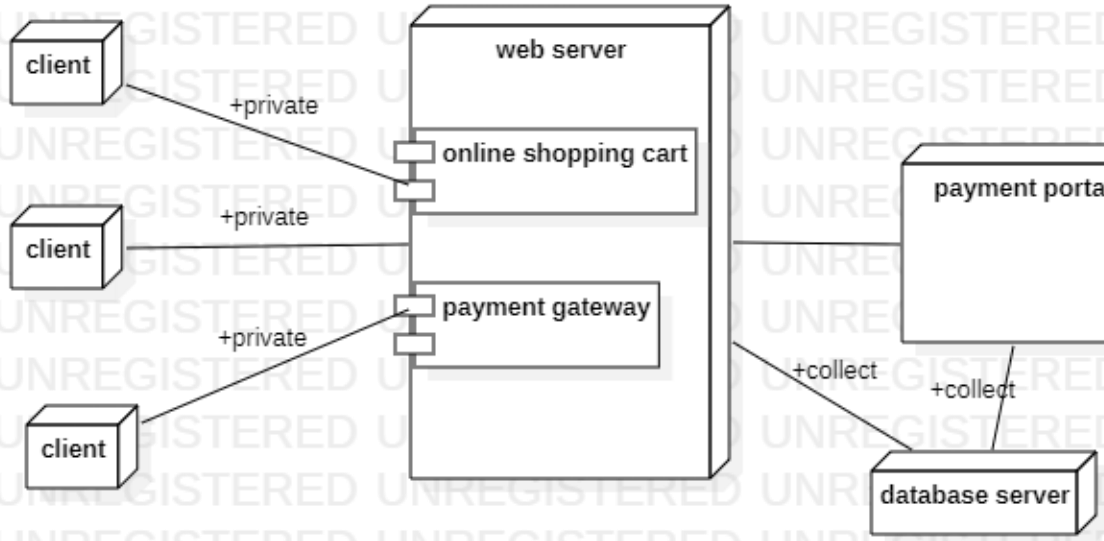
DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the relation between software and hardware. To be more specific, with deployment diagrams we can construct a physical model of how software components (artifacts) are deployed on hardware components, known as nodes.

A typical simplified deployment diagram for a web application would include:

- **Nodes** (application server and database server)
- **Artifacts** (application client and database schema

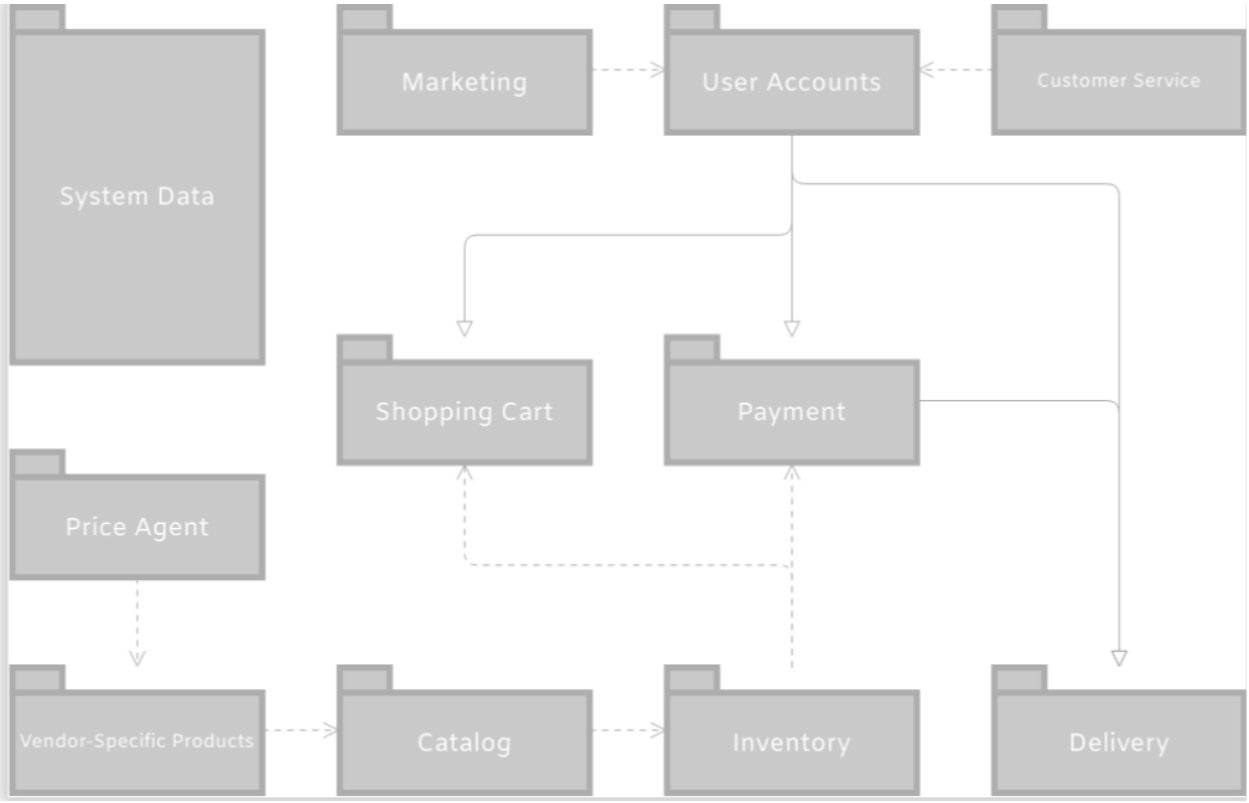
The nodes host the artifacts. The database schema runs on the database server and the application client runs on the application server.



PACKAGE DIAGRAM

The package diagram is like a macro container for deployment UML diagrams that we explained above. Different packages contain nodes and artifacts. They organize the model diagrams and components into groups, the same way a namespace encapsulates different names that are somewhat interrelated.

Ultimately a package can also be constructed by several other packages in order to depict more complex systems and behaviors. The main purpose of a package diagram is to show the relations between the different large components that make up a complex system. Programmers find this abstraction opportunity a good advantage for using package diagrams, especially when some details can be left out of the big picture.



CONCLUSION

In the modern world, the use of computers is becoming a rampant.the recent development in technology have revolutionized and consequently brought a paradigm shift in the way activities are accomplished. UML helps organizing, planning. Visualizing a program. It is used to present the idea of program in a very easy way because visuals help human brain understand anything better than text. It is based on diagrammatic representations of software components. As the old proverb says: “a picture is worth a thousand words”. By using visual representations, we are able to better understand possible flaws or errors in software or business processes.

REFERENCE:

Star UML

Oodp presentations