ELECTRONIC PAYMENT SYSTEM

PROJECT REPORT

21CSC101T- OBJECT ORIENTED DESIGN AND PROGRAMMING

(2021 Regulation)

I YEAR / II SEMESTER

Academic Year: 2023 - 2024

By:

SIMRAN (RA2212704010034)
TANMAYA (RA2212704010032)
RISHIKA CHOUDHARY (RA2212704010039)
KUDZAISHE (RA2211033010206)

Under the guidance of

Dr.P.C. Karthik

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Chennai

April 2023

BONAFIDE

This is to certify that **21CSC101T - OBJECT ORIENTED DESIGN AND PROGRAMMING** project report titled **“ELECTRONIC PAYMENT SYSTEM”** is the bonafide work of

SIMRAN (RA2212704010034)
TANMAYA (RA2212704010032)
RISHIKA CHOUDHARY (RA2212704010039)
KUDZAISHE (RA2211033010206)

who undertook the task of completing the project within the allotted time.

ABSTRACT

An e-payment or Electronic Payment system allows customers to pay for the services via electronic methods They are also known as online payment systems. Normally e-payment is done via debit, credit cards, direct bank deposits, and e-checks, other alternative e-payment methods like e-wallets, bitcoin, cryptocurrencies, bank transfers are also gaining popularity.

MODULE DESCRIPTION

The development of the Internet and the arrival of e-commerce fostered digitalization in the payment processes by providing a variety of electronic payment options including payment cards (credit and debit), digital and mobile wallets, electronic cash, contactless payment methods etc. Mobile payment services with their increasing popularity are presently under the phase of transition, heading towards a promising future of tentative possibilities along with the innovation in technology.

In our system we have provided all the options user can choose the desired method Types of

e-payment system

- Internet banking
- Card payments
- Smart card
- Stored value card
- Direct debit
- E-wallet
- QR payments
- UPI payments

How e-payment system works?

Entities involved in an online payment system

- The merchant
- The customer / the cardholder
- The issuing bank
- The acquirer
- Payment Processor
- Payment Gateway

Benefits of electronic payments

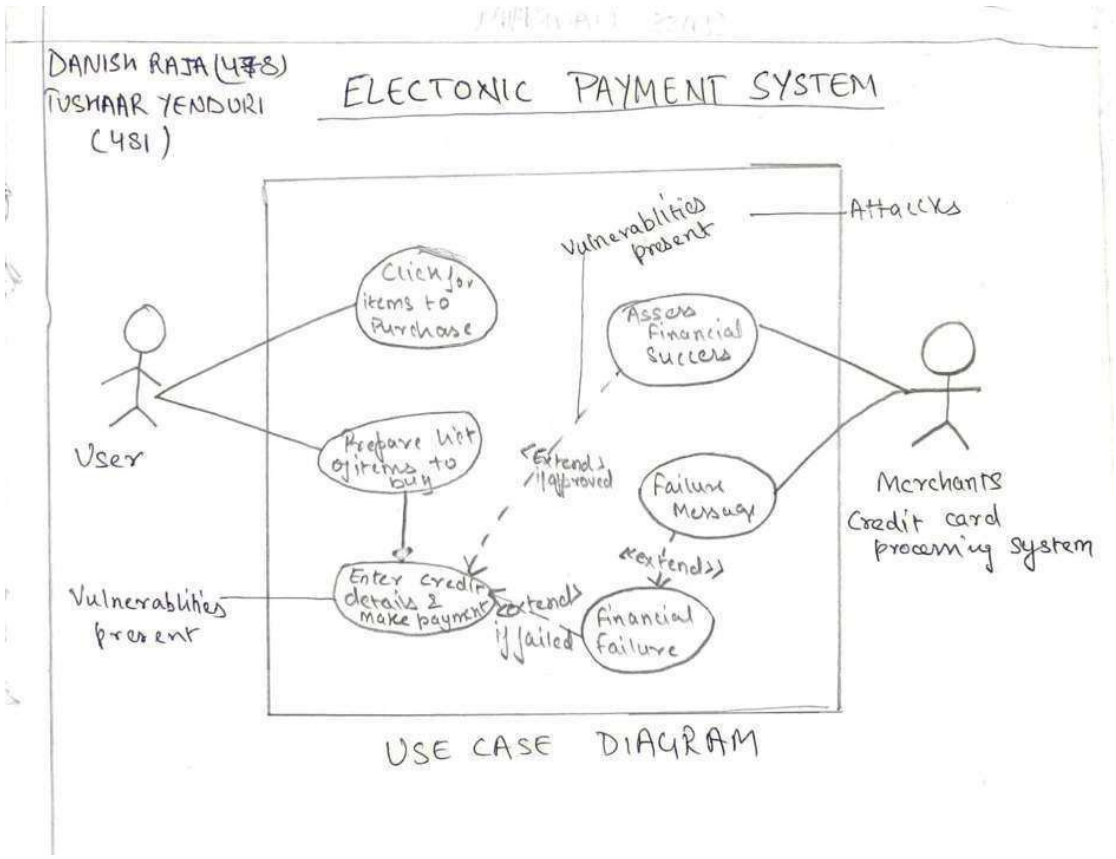
With the rapid advancements in the payment technology, electronic payment systems are becoming the substitute for the traditional payment methods. An e-payment solution supports online transactions that use e-tokens, checks, and digital cash. Electronic payment systems offer multiple benefits to the businesses and customers

- Secure transfer across internet
- High reliability: no single failure point
- Atomic transactions
- Anonymity of buyer
- Economic and computational efficiency: allow micropayments
- Flexiblility: across different methods
- Scalability in number of servers and users

Use case diagram with explanation

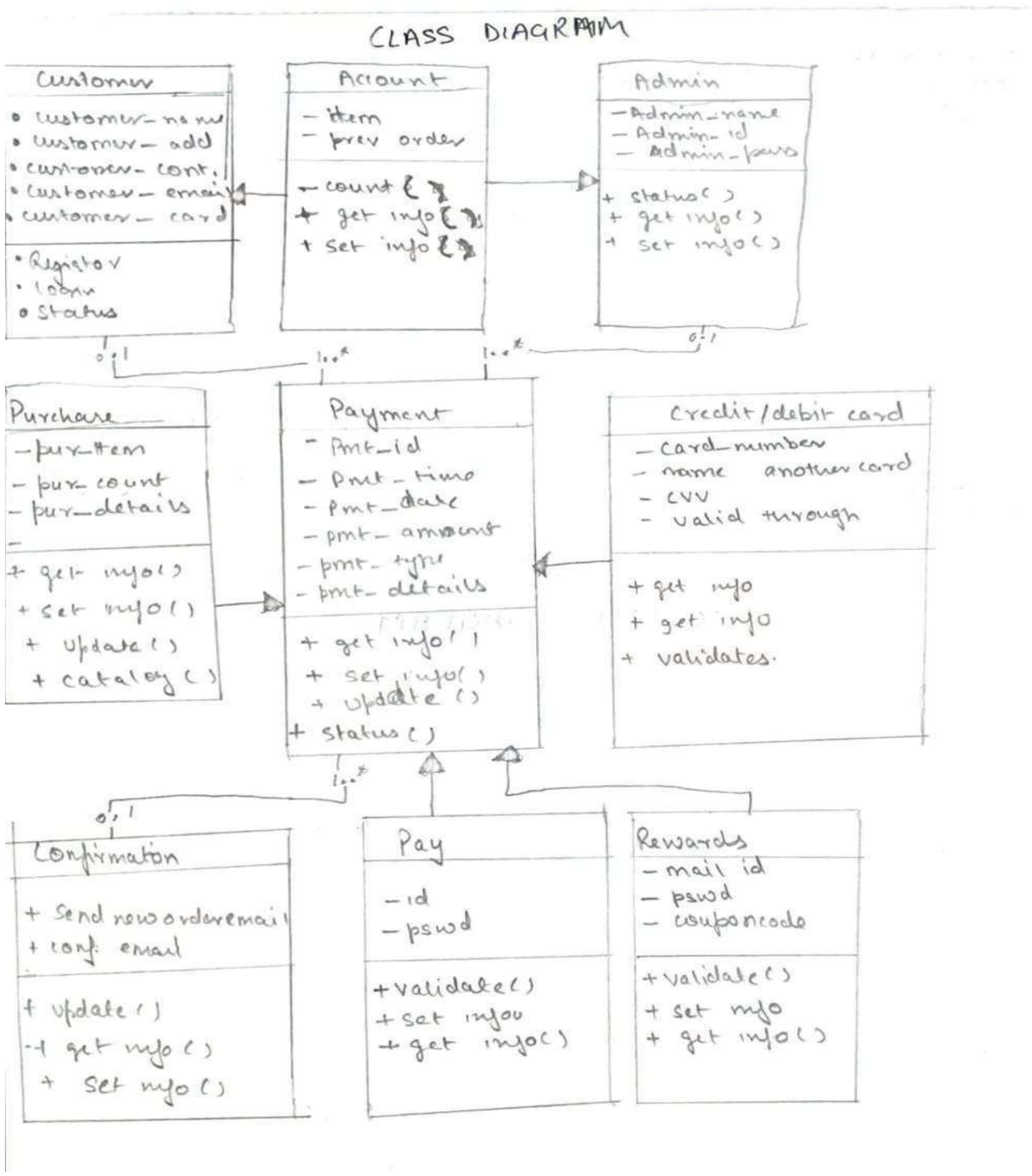
A cornerstone part of the system is the functional requirements that the system fulfills. Use Case diagrams are used to analyze the system's high-level requirements. These requirements are expressed through different use cases. We notice three main components of this UML diagram:

- 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



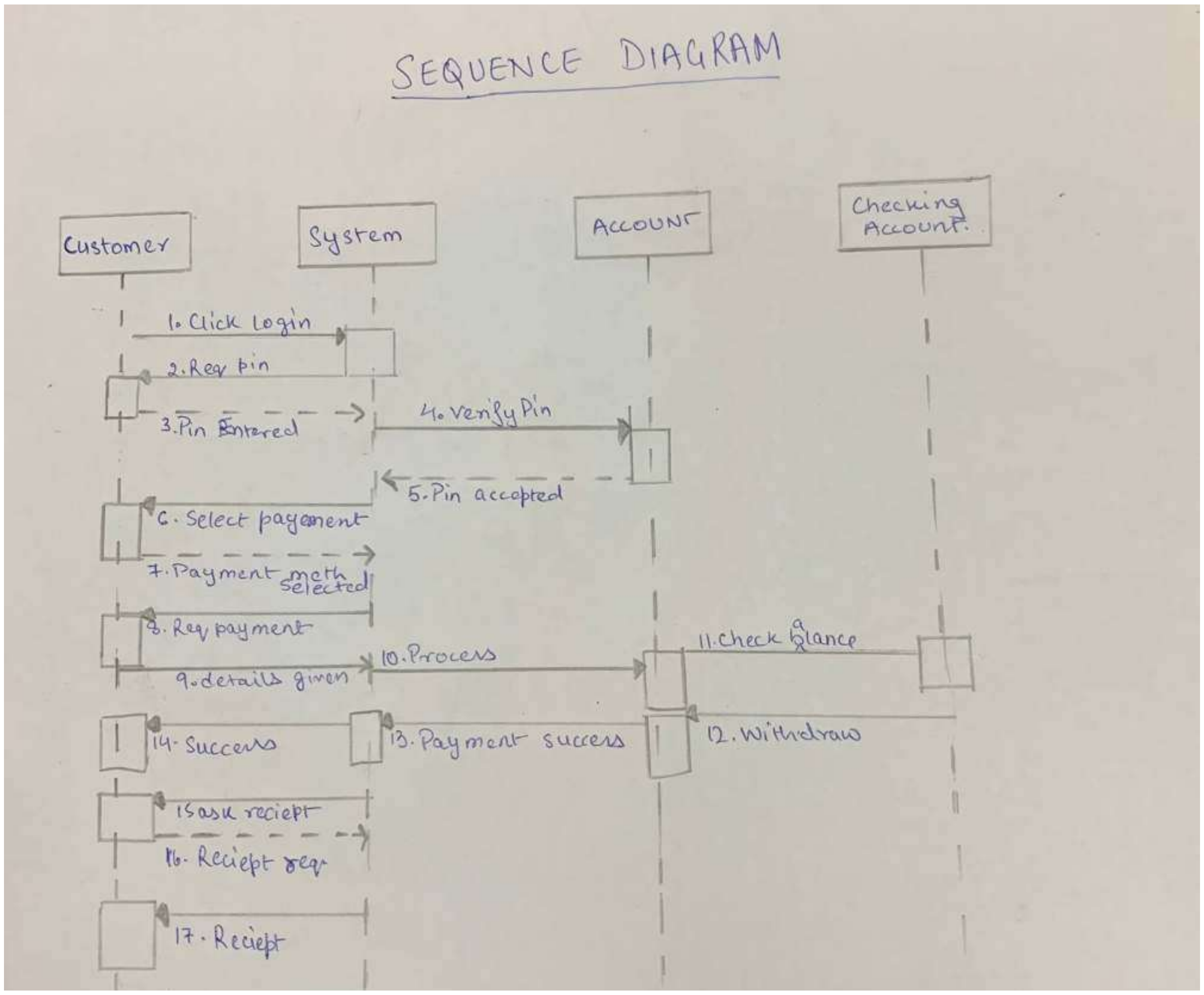
Class diagram with explanation

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.



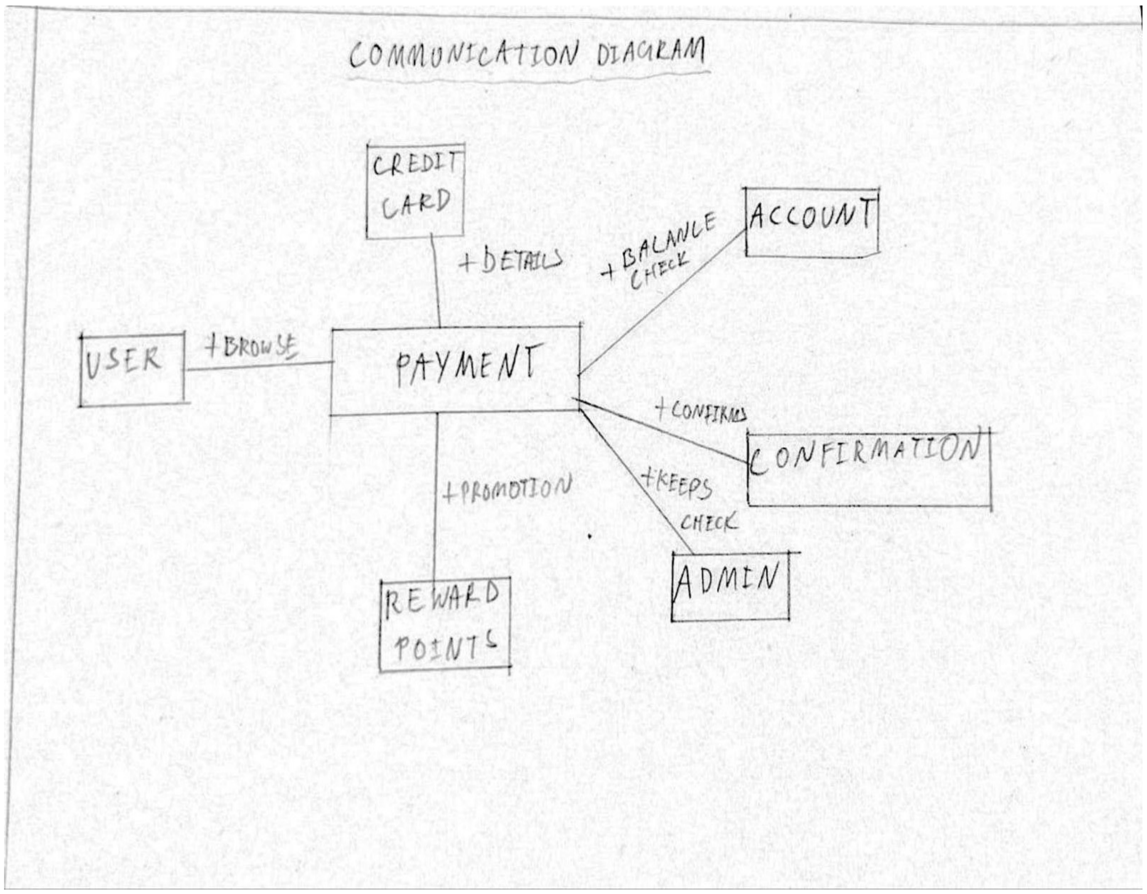
Sequence diagram with explanation

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.



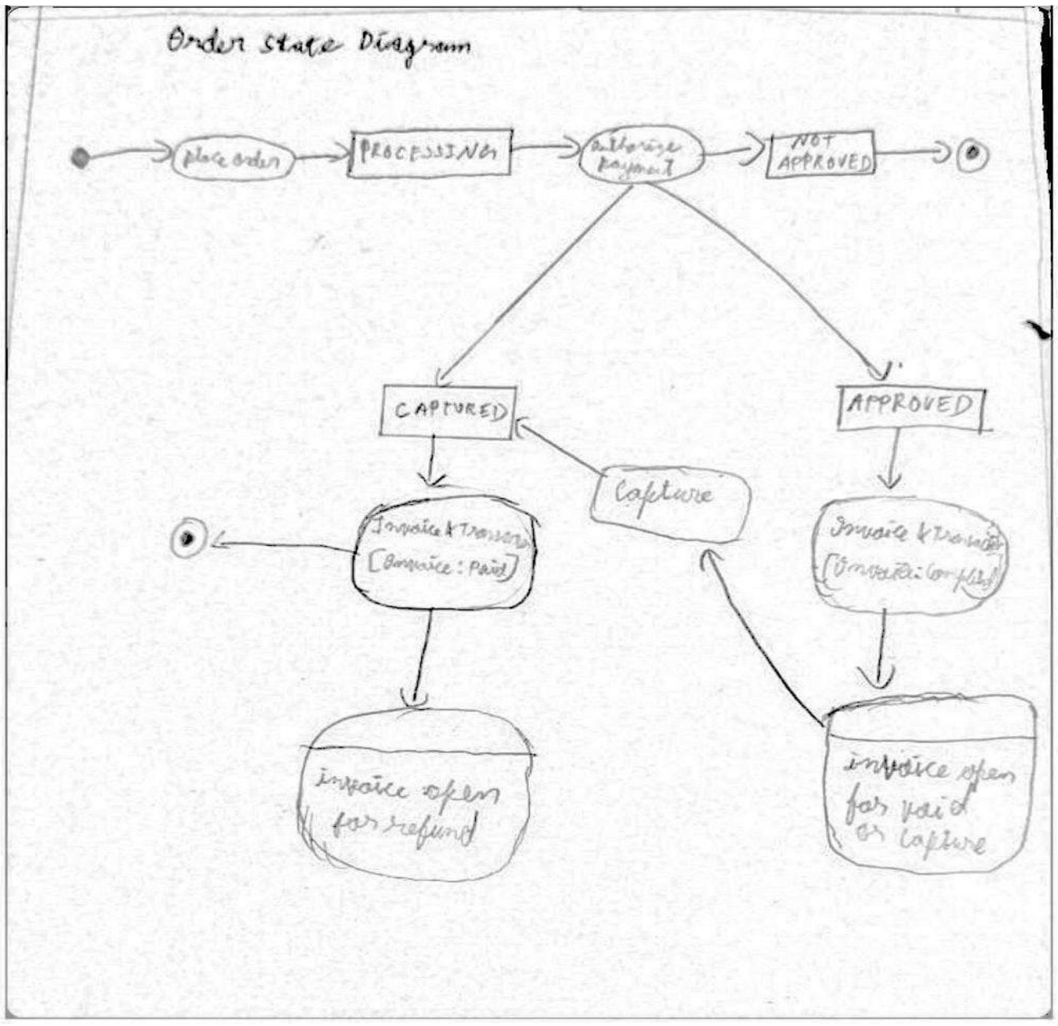
Communication diagram with explanation

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.



State chart diagram with explanation

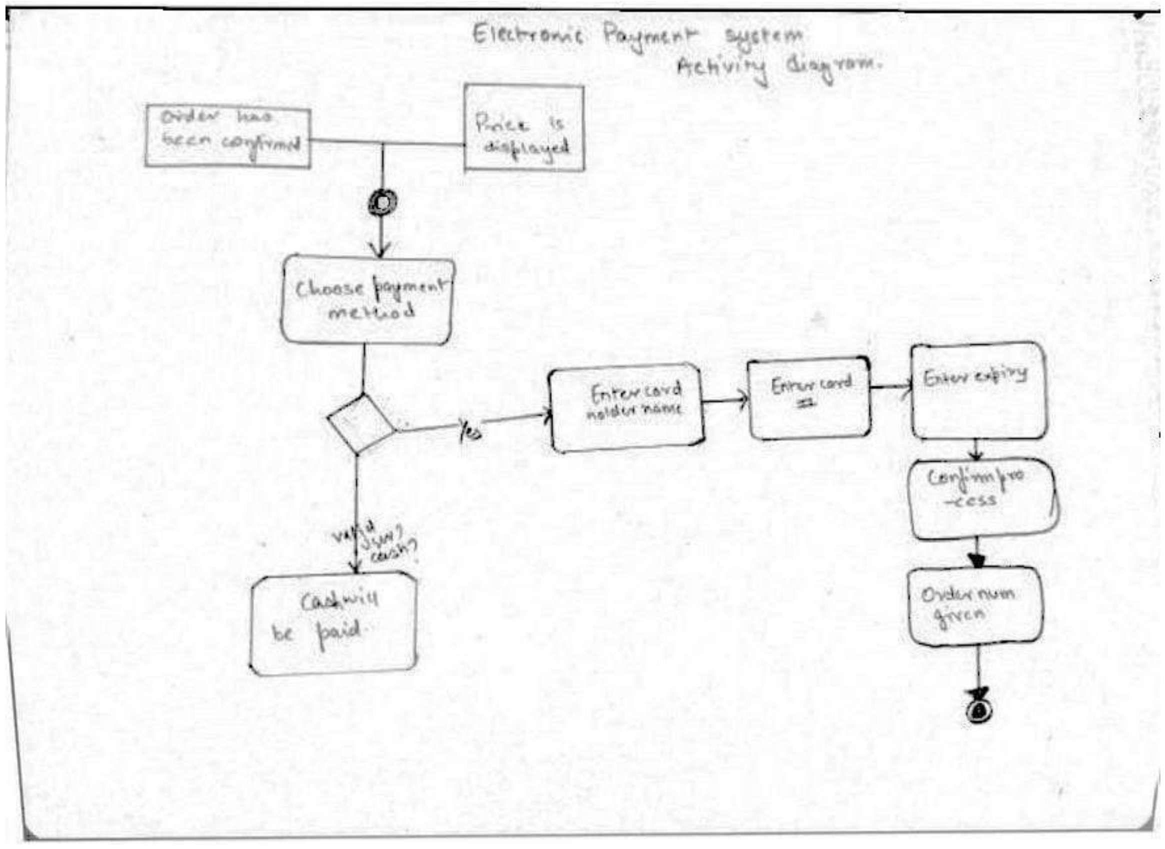
It takes the name state machine because the diagram is essentially a machine that describes the several states of an object and how it changes based on internal and external events.
A very simple state machine diagram would be that of a chess game. A typical chess game consists of moves made by White 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 with explanation

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.

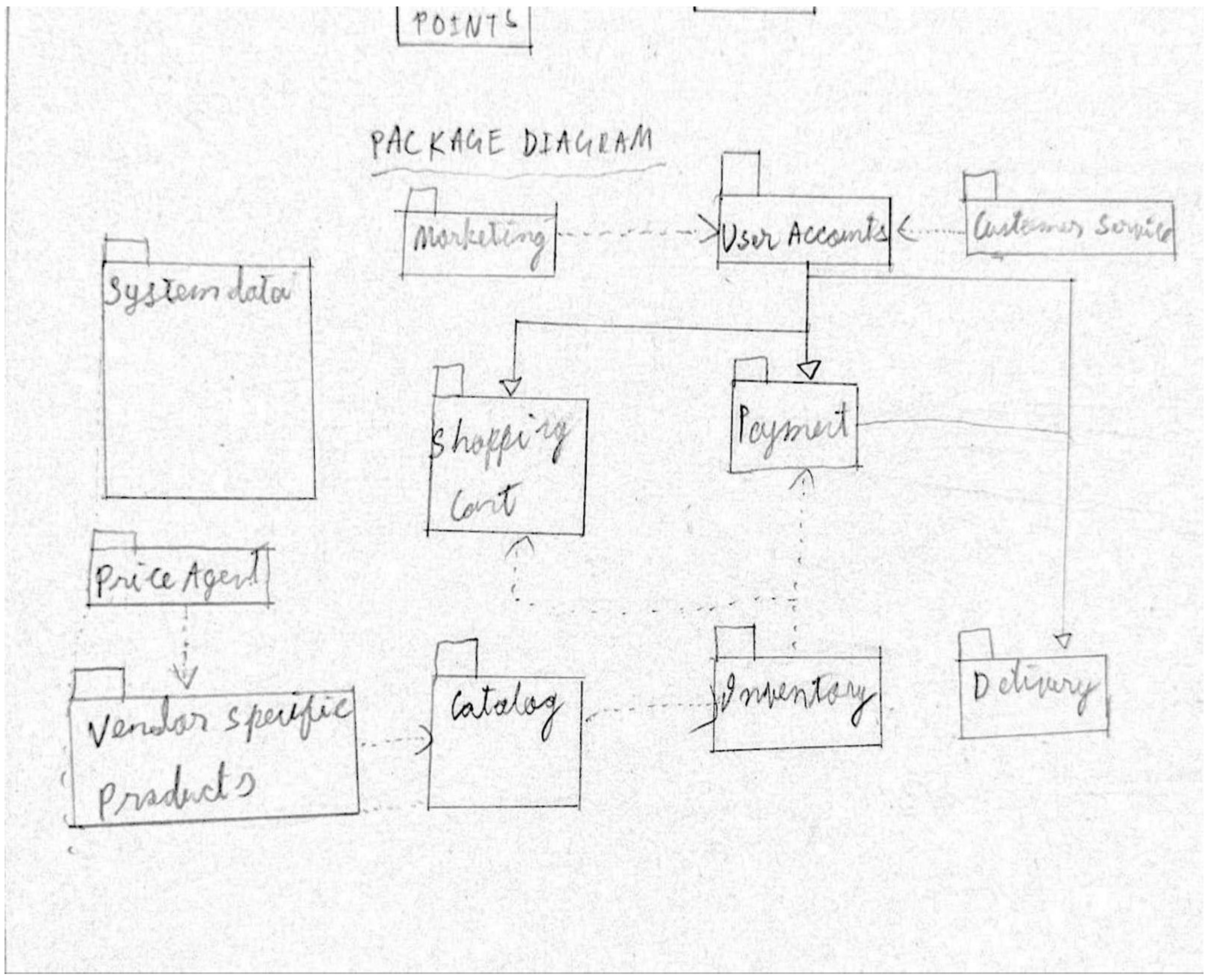
A process is not focused on what is being produced but rather on the set of activities that lead to one the other and how they are interconnected, with a clear beginning and end. The example above depicts the set of activities that take place in a content publishing process. In a business environment,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



Package diagram with explanation

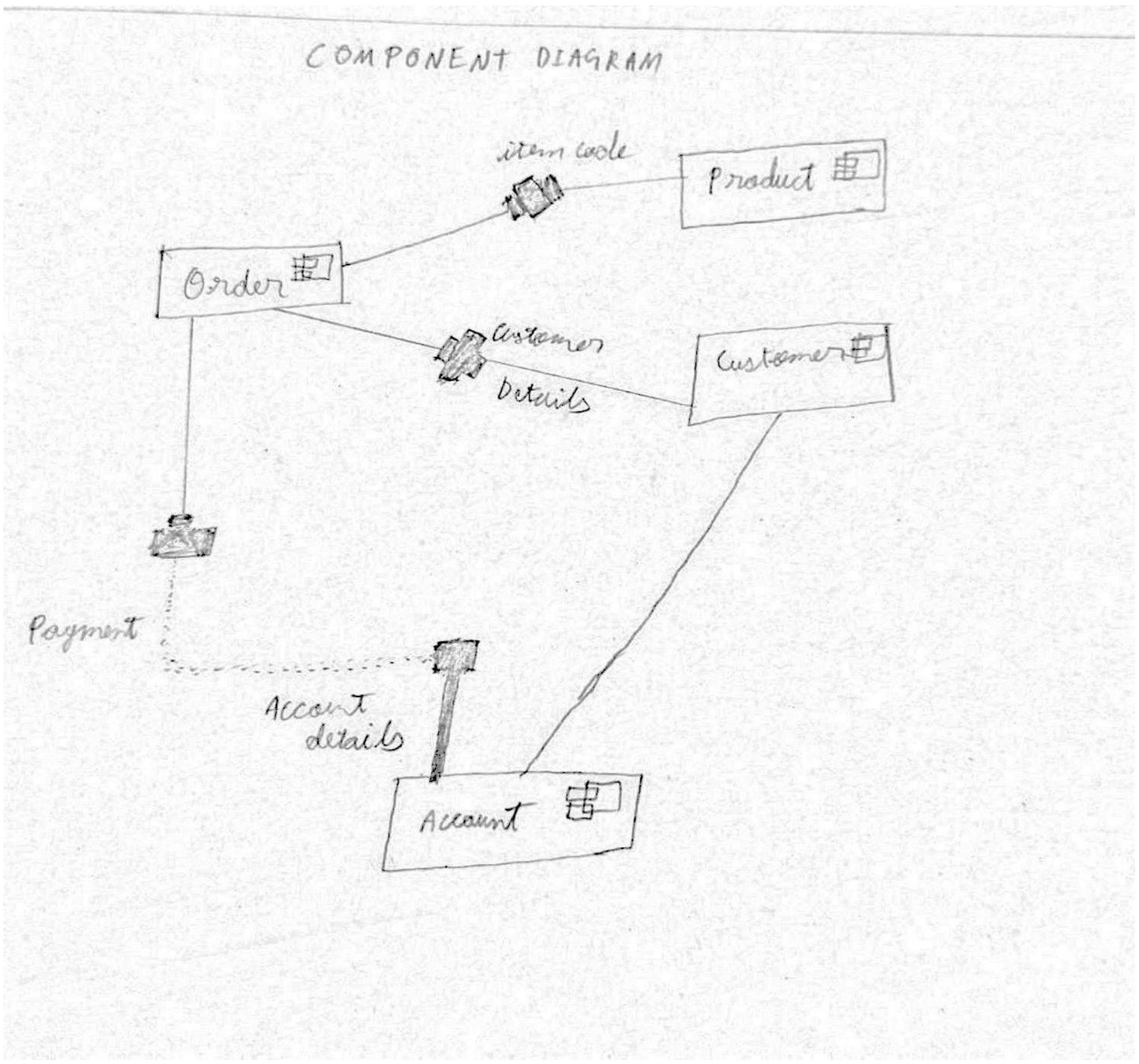
The package diagram is like a macro container for deployment UML diagrams that we explained above. Different packages contain nodes and artifacts. They organize the model diagrams and components into groups, the same way a namespace encapsulates different names that are somewhat interrelated.

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.



Component diagram with explanation

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 dataprocessing 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.



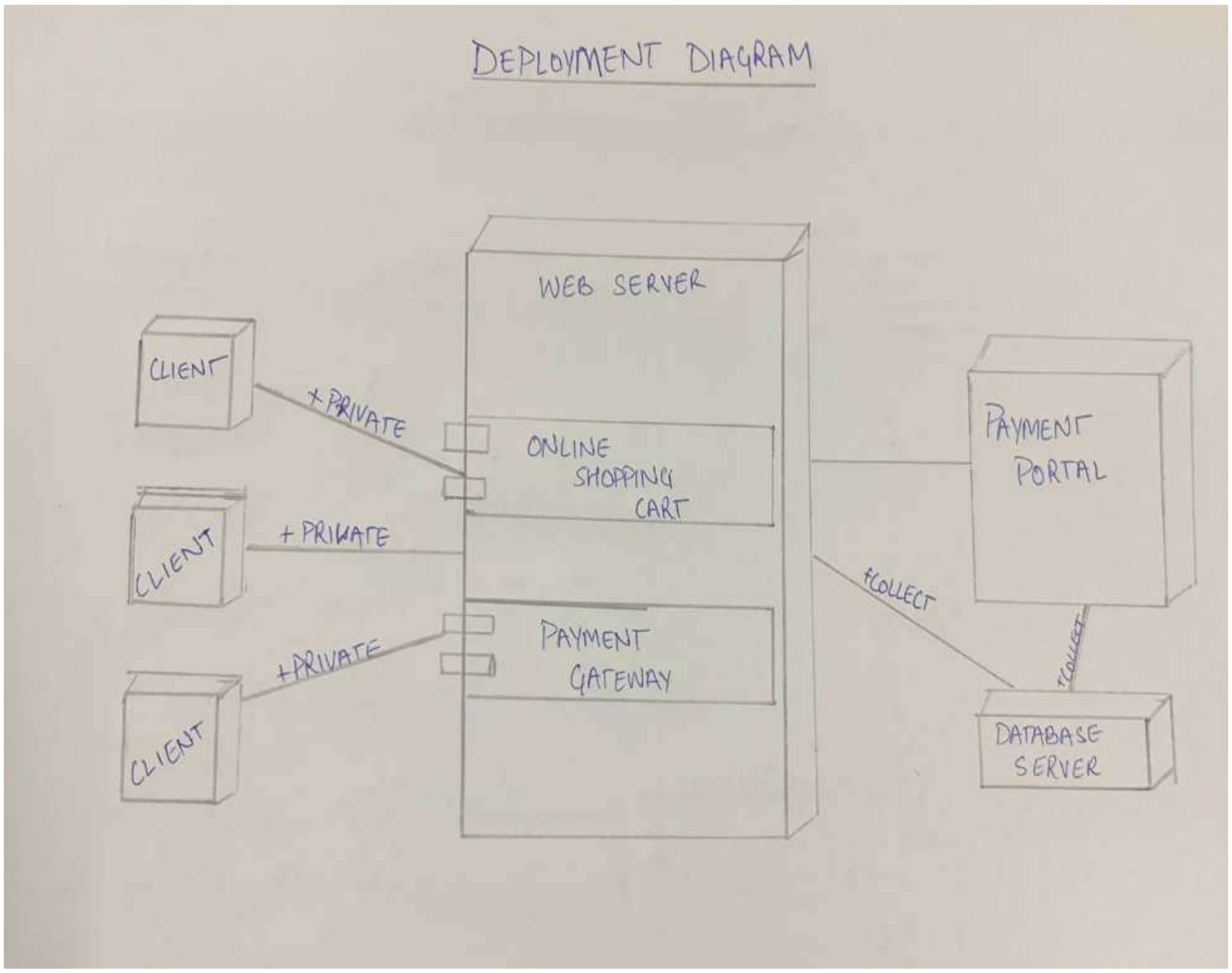
Deployment diagram with explanation

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



CODE FOR ELECTRONIC PAYMENT SYSTEM

```
#include <iostream>
#include <string>

using namespace std;

// Function to display main menu
void displayMainMenu() {
   cout << "Welcome to Electronic Payment System!" << endl;
   cout << "1. Deposit Money" << endl;
   cout << "2. Withdraw Money" << endl;
   cout << "3. Check Balance" << endl;
   cout << "4. Transfer Money" << endl;
   cout << "5. Exit" << endl;
   cout << "Enter your choice: ";
}

// Function to deposit money
void depositMoney(double& balance) {
   double amount;
   cout << "Enter amount to deposit: ";
   cin >> amount;
   balance += amount;
   cout << "Deposit successful. New balance is: " << balance << endl;
}

// Function to withdraw money
void withdrawMoney(double& balance) {
   double amount;
   cout << "Enter amount to withdraw: ";
   cin >> amount;
   if (amount > balance) {
      cout << "Insufficient balance. Withdrawal failed." << endl;
   } else {
      balance -= amount;
      cout << "Withdrawal successful. New balance is: " << balance << endl;
   }
}

// Function to check balance
void checkBalance(double balance) {
   cout << "Your current balance is: " << balance << endl;
}

// Function to transfer money
void transferMoney(double& balance) {
   double amount;
   string recipient;
   cout << "Enter recipient name: ";
   cin >> recipient;
   cout << "Enter amount to transfer: ";
   cin >> amount;
   if (amount > balance) {
      cout << "Insufficient balance. Transfer failed." << endl;
```

```
    } else {
        balance -= amount;
        cout << "Transfer successful. " << amount << " has been transferred to " << recipient << endl;
        cout << "New balance is: " << balance << endl;
    }
}

int main() {
    double balance = 0;
    int choice = 0;
    do {
        displayMainMenu();
        cin >> choice;

        switch (choice) {
            case 1:
                depositMoney(balance);
                break;

            case 2:
                withdrawMoney(balance);
                break;

            case 3:
                checkBalance(balance);
                break;

            case 4:
                transferMoney(balance);
                break;

            case 5:
                cout << "Thank you for using Electronic Payment System!" << endl;
                break;

            default:
                cout << "Invalid choice. Please try again." << endl;
        }
    } while (choice != 5);

    return 0;
}
```

Conclusion

E-Payment has taken over the world, all it takes is a single click on your mobile phone to make your transaction, from a pani puri bhaiya to a 5 star hotel everyone accepts e-payment nowadays. In offline mode there are more chances of scams and in online transaction It is much safer and easier to keep track as we can keep record with us instead of going to bank again and again, E-payment easier and highly liquid