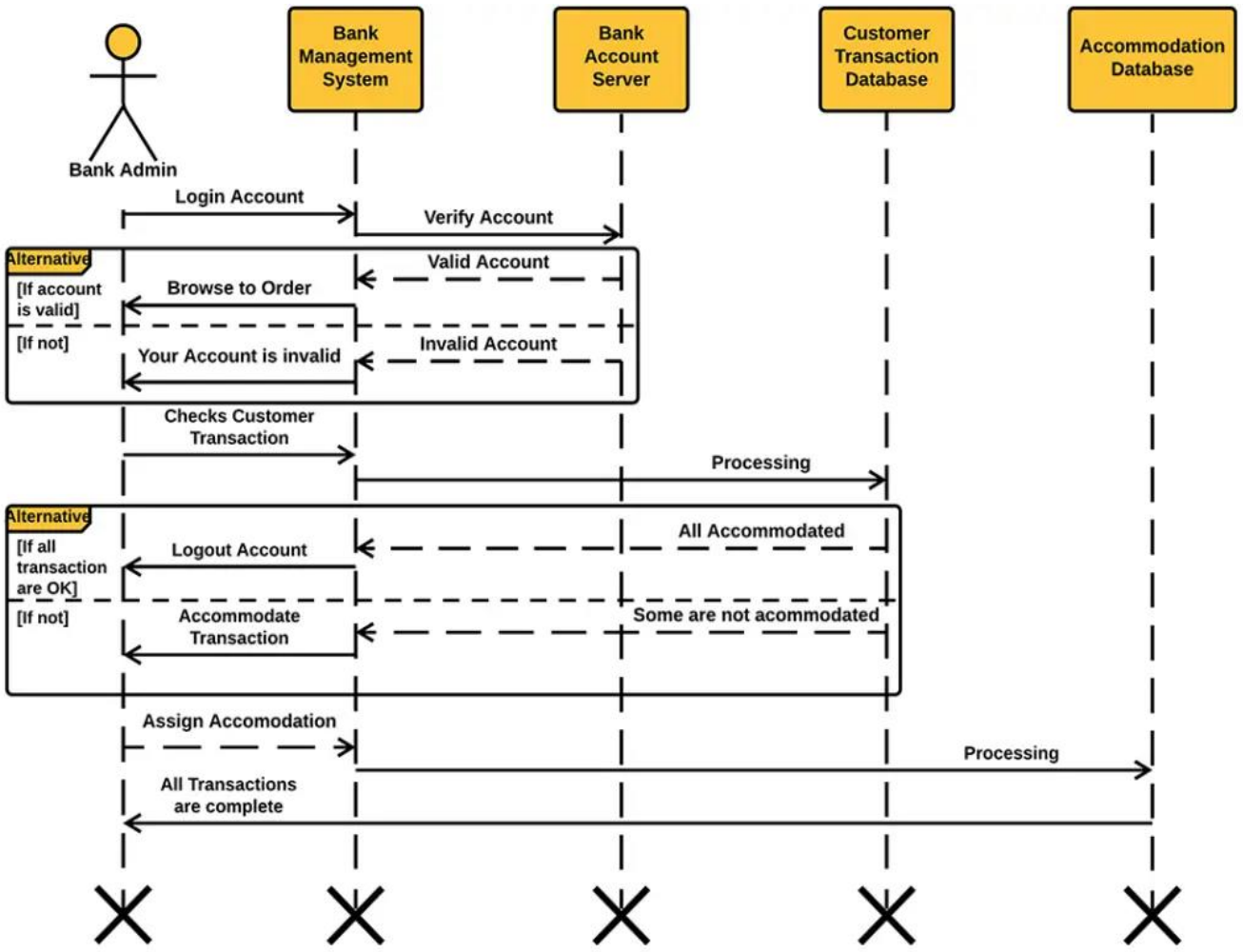
EXPERIMENT – 5

AIM:- Draw the sequence diagram for These two scenarios for a given problem

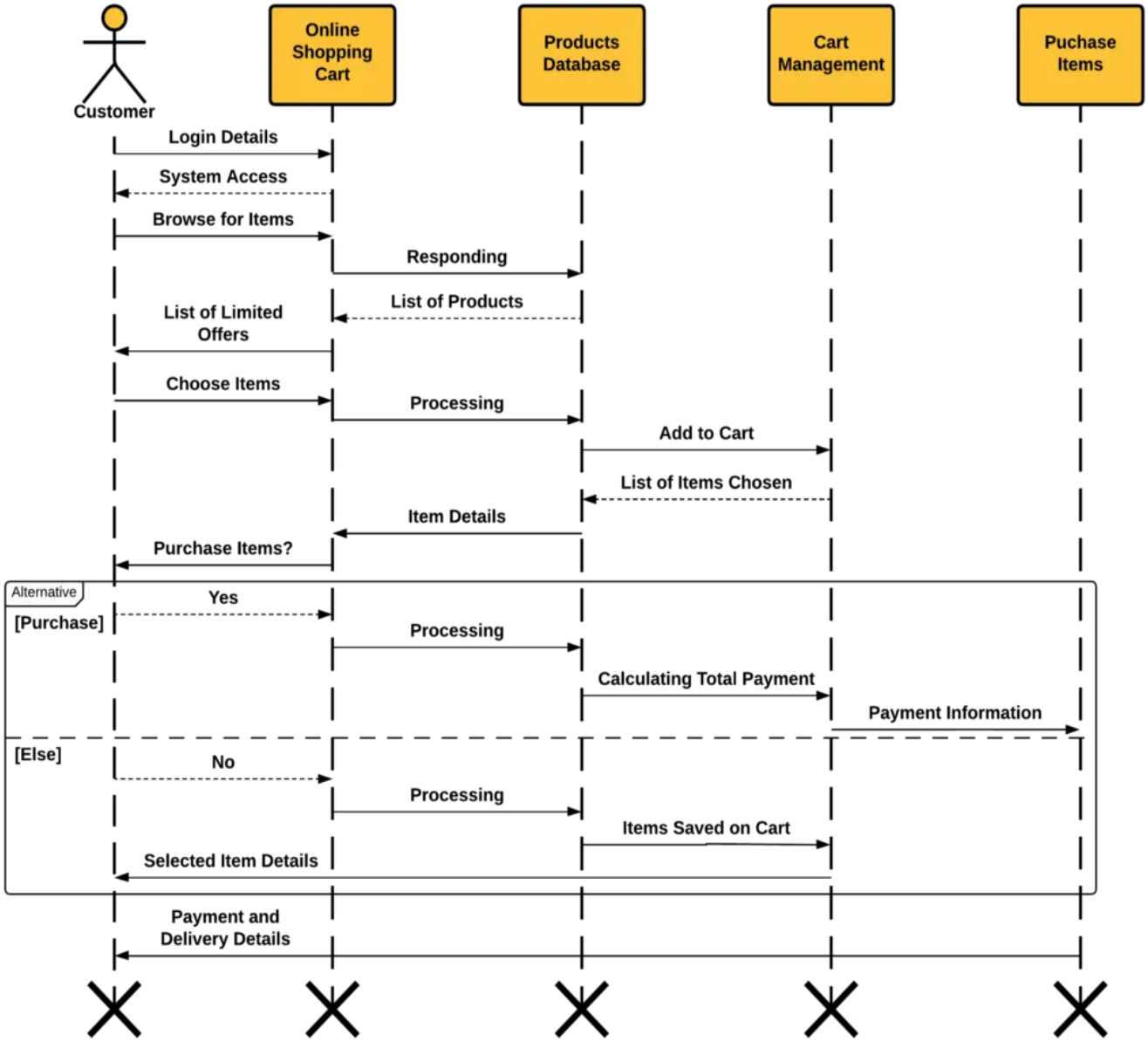
THEORY:- A sequence diagram is a type of UML (Unified Modeling Language) diagram that shows interactions between objects or components in a system over time. It is used in software engineering to visualize the flow of messages between objects, and is particularly useful for modeling complex systems with many interacting components.

DAIGRAM:-

1). BANK MANAGEMENT SYSTEM :-



2). ONLINE SHOPPING SYSTEM :-



Conclusion :-

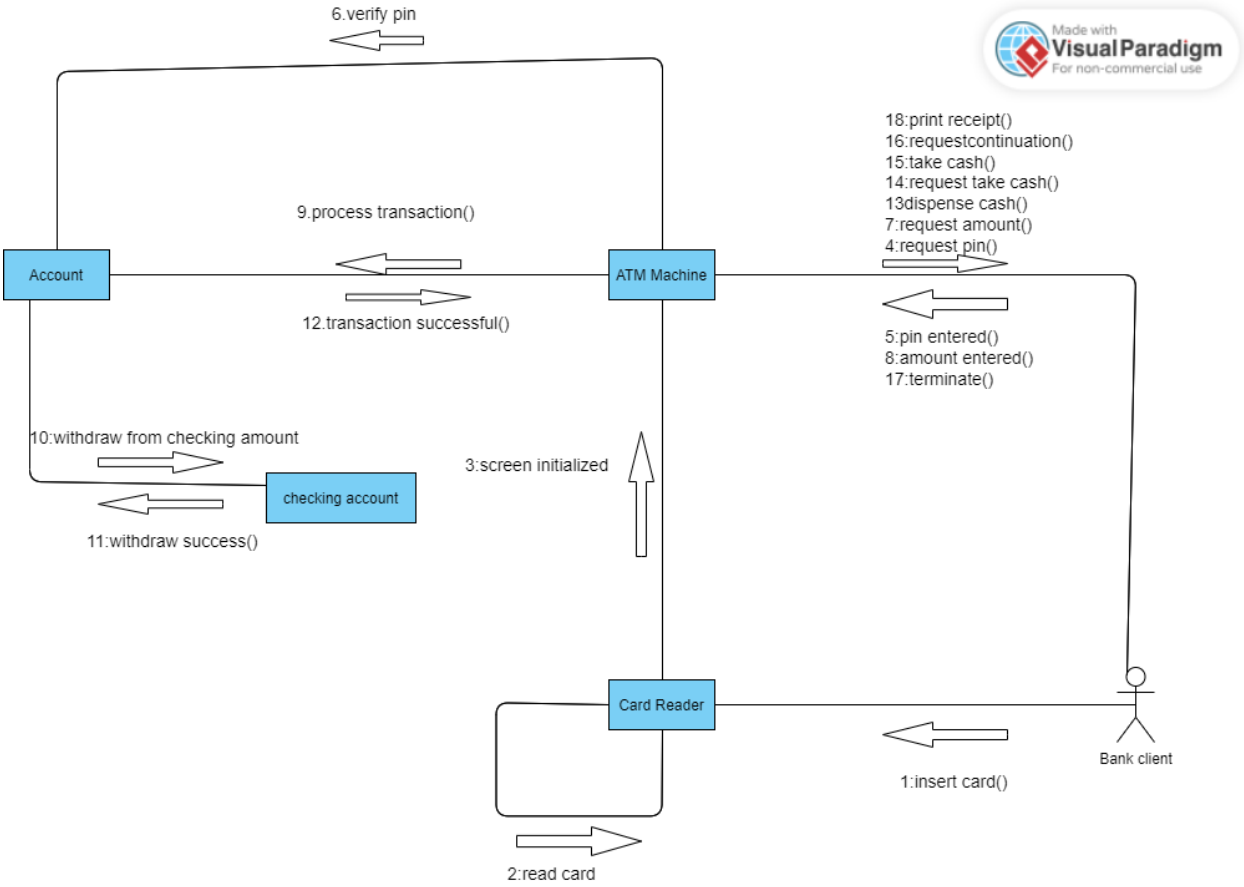
- The sequence diagram is drawn for the given problem.

EXPERIMENT – 6

AIM:- Draw the collaboration diagram for online shopping system.

THEORY:- A collaboration diagram, also known as a communication diagram, is a type of interaction diagram that shows the interactions and communications between objects or roles in a system. It is a visual representation of how objects or roles collaborate with each other to achieve a specific task or goal.

DAIGRAM:-



Conclusion :-

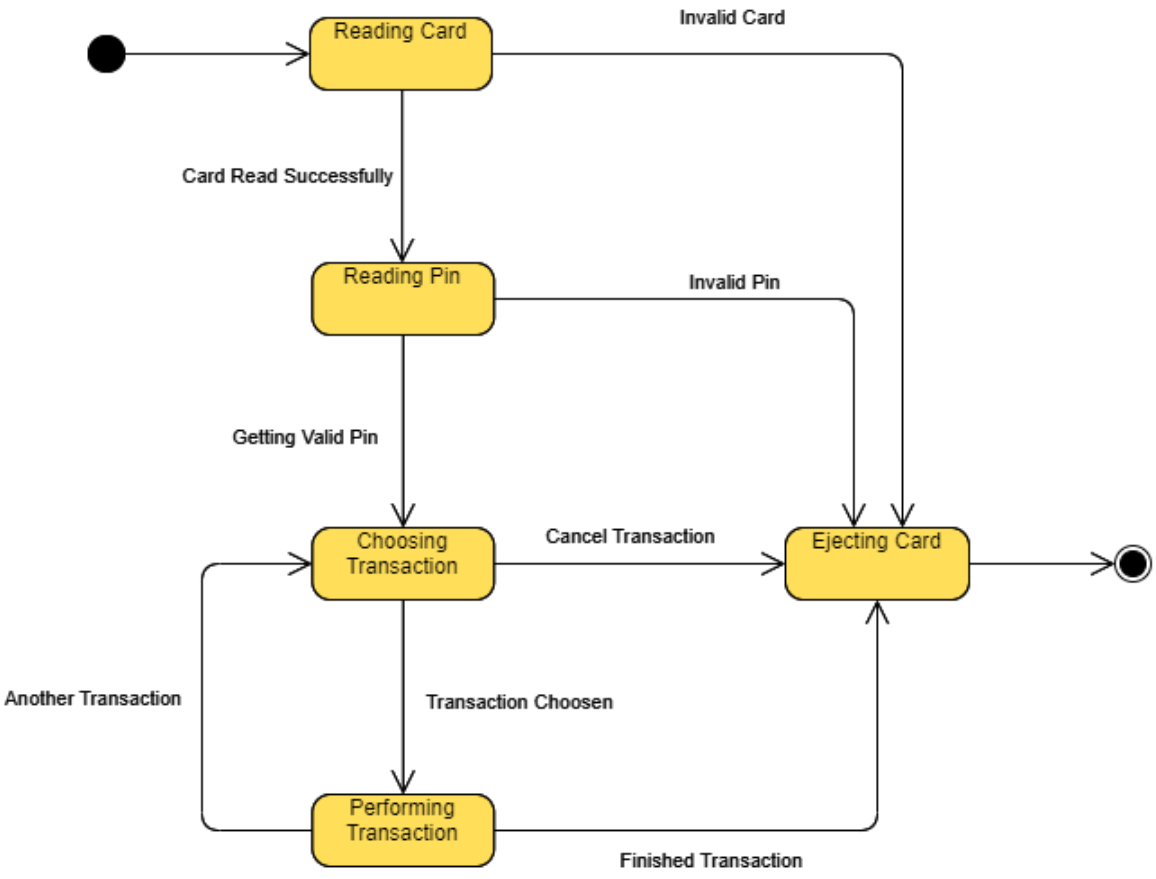
- The collaboration diagram is drawn for the given problem.

EXPERIMENT – 7

AIM:- Draw the state chart diagram for ATM Machine Transaction Processing.

THEORY:- A chart diagram is a graphical representation of a software system or a specific aspect of it. It is a visual representation of software components, their relationships, and interactions. The purpose of a chart diagram is to help software developers understand the structure of a software system, its components, and their interdependencies.

DAIGRAM:-



Conclusion :-

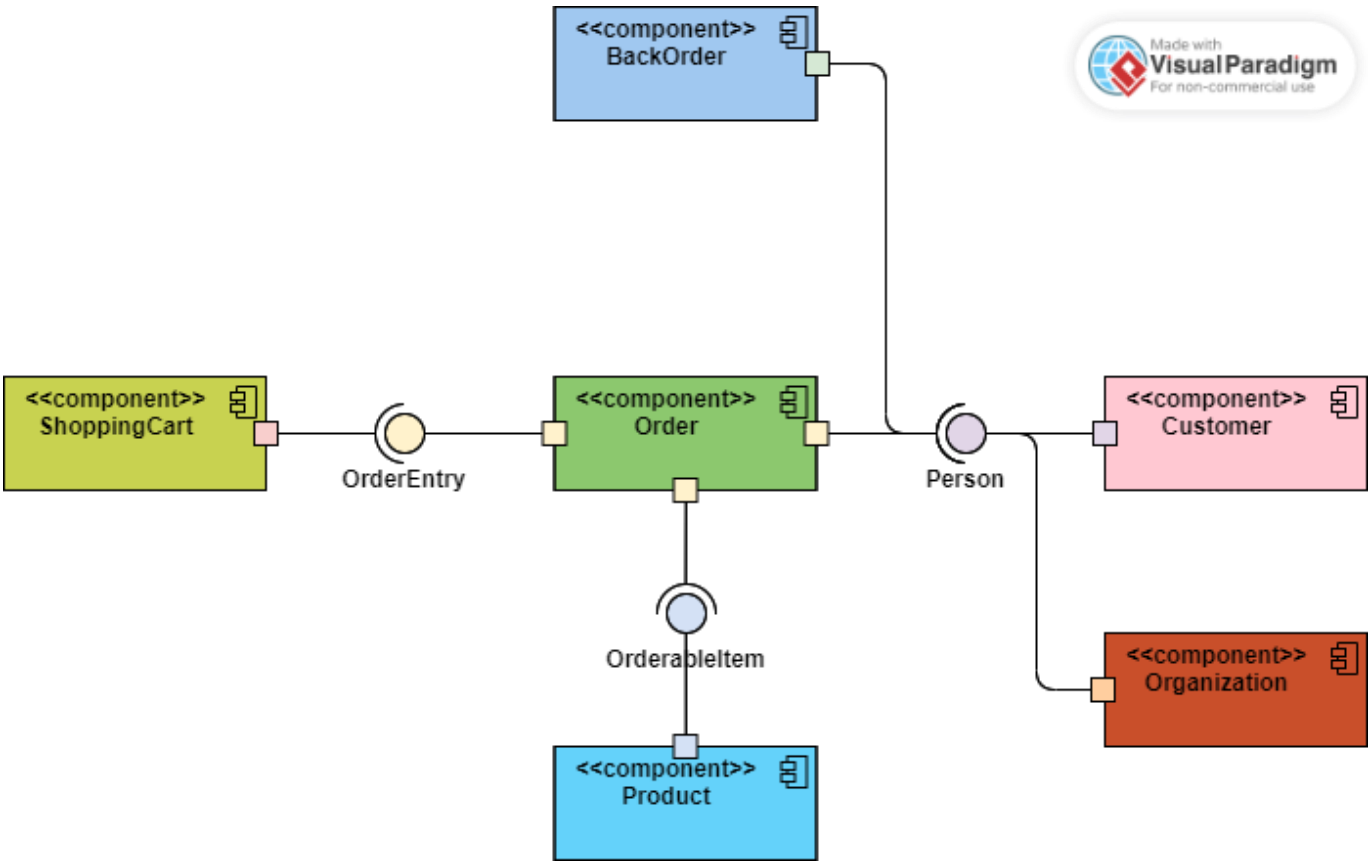
- The state chart diagram for drawn ATM Machine Transaction Processing.

EXPERIMENT – 8

AIM:- Draw the component diagram for online shopping.

THEORY:- A component diagram can be used to represent the overall architecture of a software system, including the different components and how they interact with each other. It can also be used to illustrate the relationships between components and their interfaces, dependencies, and associations.

DAIGRAM:-



Conclusion :-

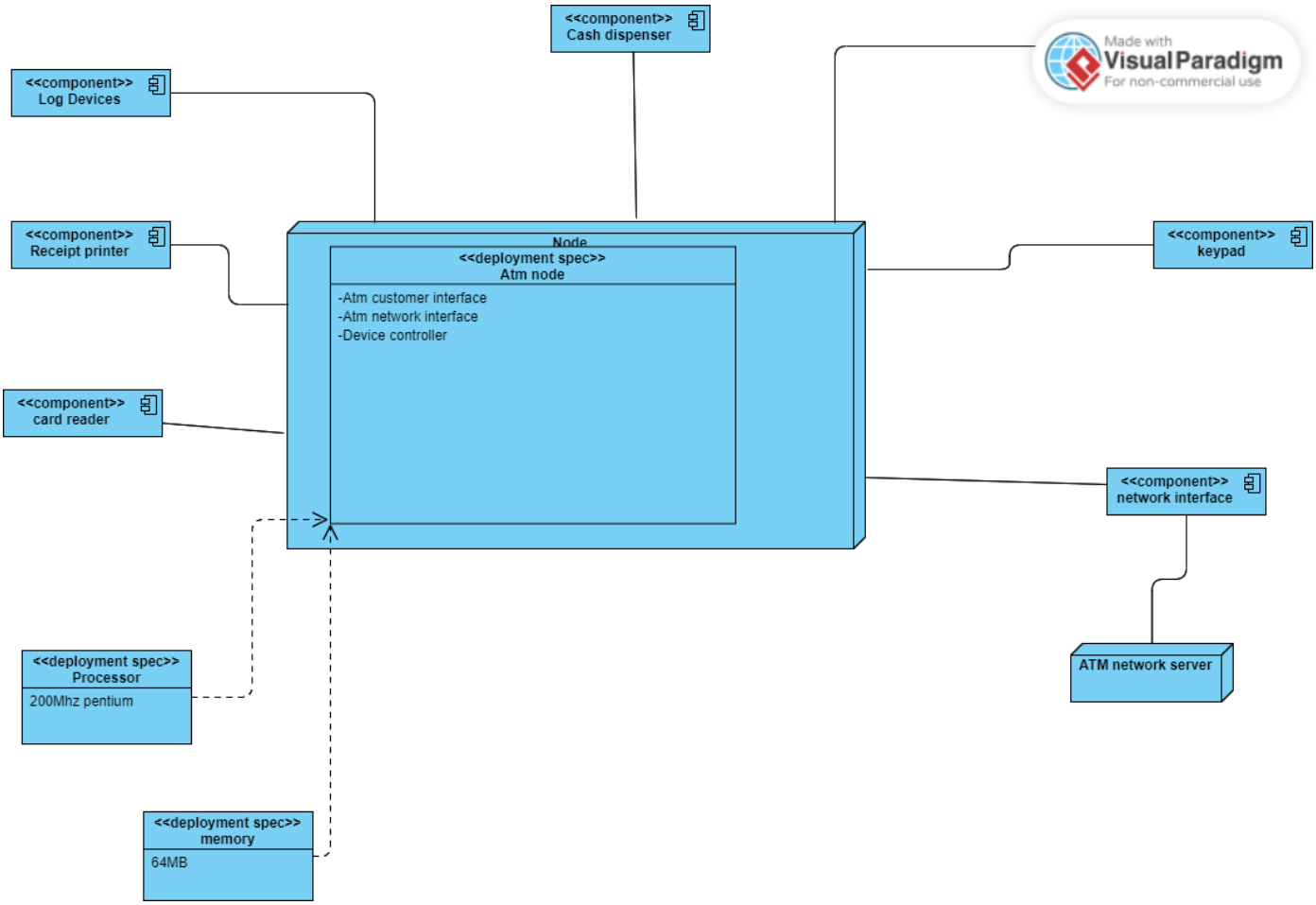
- The component diagram drawn for online shopping.

EXPERIMENT – 9

AIM:- Draw the deployment diagram for Hospital Management System.

THEORY:- A deployment diagram consists of nodes, which are physical entities, and components, which are software entities. Nodes can represent devices such as servers, computers, or mobile devices, while components can represent software applications, modules, or libraries. The connections between nodes and components represent communication channels or network protocols used for inter-component communication.

DAIGRAM:-



Conclusion :-

- The deployment diagram drawn for Hospital Management System.

EXPERIMENT – 10

AIM:- Draw Data Flow Diagram for Bank Account.

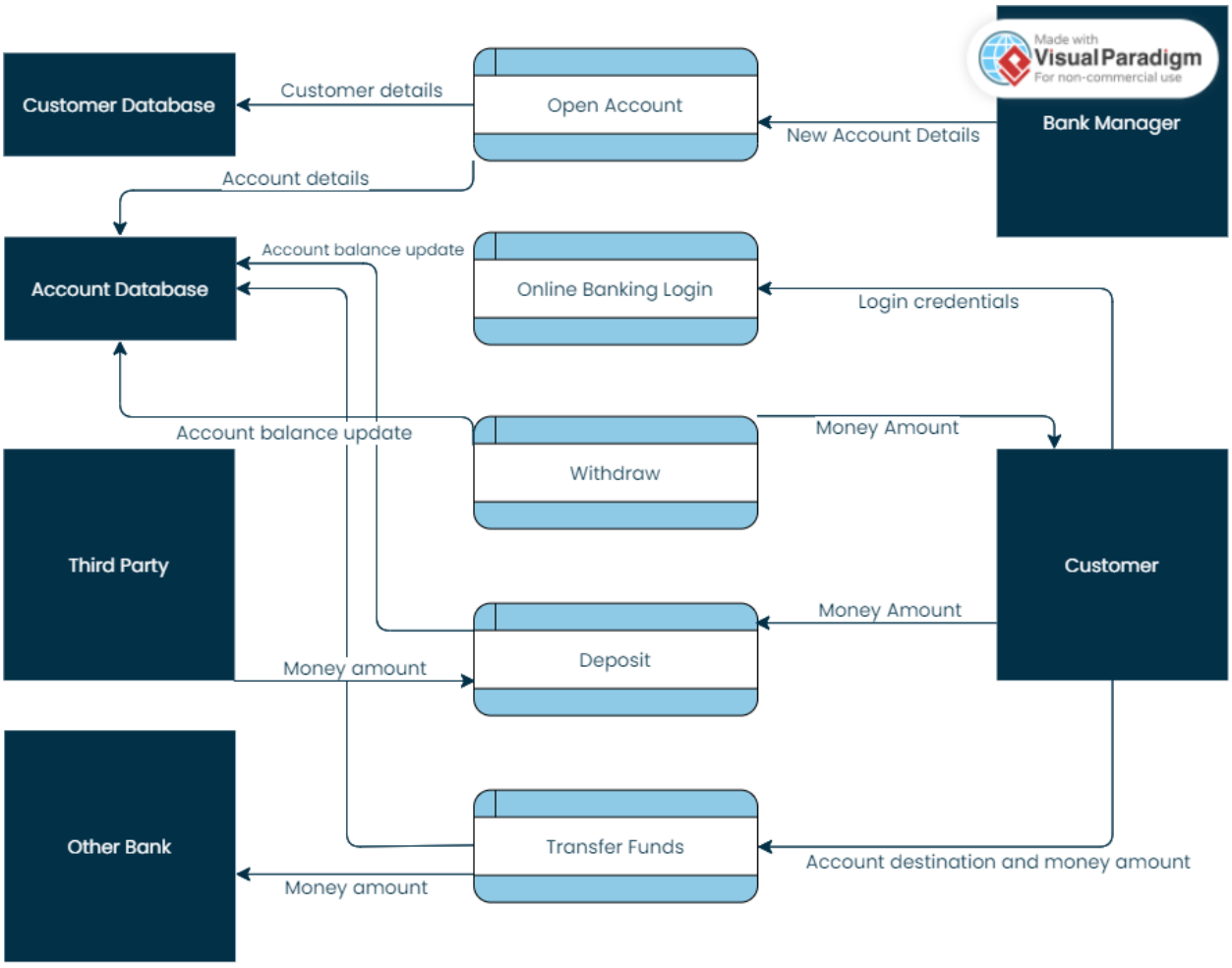
THEORY:- A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It is used to show how data moves between different processes, entities, and data stores within a system.

To draw a data flow diagram, you first need to identify the inputs, outputs, processes, and data stores in the system. You can then use symbols to represent these components and show how data flows between them.

There are four main components in a DFD:

1. Entities: These are the external entities that send or receive data to or from the system.
2. Processes: These are the operations or transformations that are performed on the data. Data
3. Flows: These represent the movement of data between the entities, processes, and data
4. stores. Data Stores: These are the places where data is stored within the system.

DAIGRAM:-



EXPERIMENT – 11

AIM:- Write a program to compute cyclomatic complexity.

THEORY:- Cyclomatic complexity is a software metric that measures the complexity of a program based on the number of independent paths through its source code. The higher the cyclomatic complexity, the more complex the program is considered to be, which can make it more difficult to understand, test, and maintain.

CODE:-

```
#include <stdio.h>
 int main() {
    int nodes, edges, connected_components;
    printf("Enter the number of nodes: ");
    scanf("%d", &nodes);
    printf("Enter the number of edges: ");
    scanf("%d", &edges);
    printf("Enter the number of connected components: ");
    scanf("%d", &connected_components);
    int cyclomatic_complexity = edges - nodes + 2 * connected_components;
    printf("Cyclomatic Complexity: %d\n", cyclomatic_complexity);
    return 0;
 }
```

Output:-

```
Output
/tmp/GP6yekxP7T.o
Enter the number of nodes: 5
Enter the number of edges: 5
Enter the number of connected components: 3
Cyclomatic Complexity: 6
```

Conclusion :-

- The cyclomatic complexity is calculated for the given problem.

EXPERIMENT – 12

AIM:- Write a program to compute Halstead's Complexity.

THEORY:- Halstead complexity measures are software metrics introduced by Maurice Howard Halstead in 1977 as part of his treatise on establishing an empirical science of software development. Halstead makes the observation that metrics of the software should reflect the implementation or expression of algorithms in different languages, but be independent of their execution on a specific platform. These metrics are therefore computed statically from the code.

CODE:-

```
#include <stdio.h>
#include <math.h>
int main()
{
 float op_distinct, opd_distinct, op_total, opd_total;
 float length, vocab_size, volume, level, difficulty, effort;
 float time;
 float k = 18;
 printf("\n Enter the value of number of distinct operators: ");
 scanf("%f", &op_distinct);
 printf("\n Enter the value of number of distinct operands: ");
 scanf("%f", &opd_distinct);
 printf("\n Enter the value of total number of operators: ");
 scanf("%f", &op_total);
 printf("\n Enter the value of total number of operands: ");
 scanf("%f", &opd_total);
 vocab_size = op_distinct + opd_distinct;
 length = op_total + opd_total;
 volume = length * log(vocab_size);
 level = (op_distinct / 2) * (opd_total / opd_distinct);
 difficulty = 1 / level;
 effort = volume * difficulty;
 time = effort / k;
 printf("\n Program Vocabulary: %f", vocab_size);
```

```
 printf("\n Program Length: %f", length);
 printf("\n Calculated Program Length: %f", volume);
 printf("\n Volume: %f", volume);
 printf("\n Difficulty: %f", difficulty);
 printf("\n Effort: %f", effort);
 printf("\n Time to implement: %f", time);
 return 0;
}
```

CODE:-

Output

```
/tmp/6qV0bu8FWN.o
Enter the value of number of distinct operators: 3
Enter the value of number of distinct operands: 4
Enter the value of total number of operators: 8
Enter the value of total number of operands: 12
Program Vocabulary: 7.000000
 Program Length: 20.000000
 Calculated Program Length: 38.918201
 Volume: 38.918201
 Difficulty: 0.222222
 Effort: 8.648489
 Time to implement: 0.480472
```

Conclusion :-

- The Halstead's Complexity is calculated for the given problem.