MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY

OBJECT ORIENTED SOFTWARE ENGINEERING

LAB FILE

Submitted to:
Ms. Sonica Upadhyay

Submitted by:
Chander Shekhar
11515002719
CSE-3

INDEX

S.No.	Date	Experiment	Signature
1.	14/03/2023	Introduction to unified modelling language (UML).	
2.	21/03/2023	To make Use Case diagram for Automated Teller Machine (ATM)	
3.	11/04/2023	To make Sequence diagram for Automated Teller Machine (ATM)	
4.	25/04/2023	To make Collaboration diagram for Automated Teller Machine (ATM)	
5.	01/05/2023	To make the Class diagram for Automated Teller Machine (ATM)	
6.	02/05/2023	To make the State Chart diagram for Automated Teller Machine (ATM)	
7.	23/05/2023	To make the Activity diagram for Automated Teller Machine (ATM)	
8.	30/05/2023	To make the Component diagram for Automated Teller Machine (ATM)	
9.	30/05/2023	To make the Deployment diagram for Automated Teller Machine (ATM)	

EXPERIMENT – 1

Aim:

Introduction to unified modelling language (UML).

Theory:

A picture is worth a thousand words. That's why Unified Modelling Language (UML) diagramming was created: to forge a common visual language in the complex world of software development that would also be understandable for business users and anyone who wants to understand a system. Learn the essentials of UML diagrams along with their origins, uses, concepts, types and guidelines on how to draw them using our UML diagram tool.

The Unified Modeling Language (UML) was created to forge a common, semantically, and syntactically rich visual modeling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally. UML has applications beyond software development, such as process flow in manufacturing.

It is analogous to the blueprints used in other fields and consists of different types of diagrams. In the aggregate, UML diagrams describe the boundary, structure, and the behavior of the system and the objects within it.

UML is not a programming language but there are tools that can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design. There are many problem-solving paradigms or models in Computer Science, which is the study of algorithms and data.

There are four problem-solving model categories: imperative, functional, declarative and object-oriented languages (OOP). In object-oriented languages, algorithms are expressed by defining 'objects' and having the objects interact with each other. Those objects are things to be manipulated and they exist in the real world. They can be buildings, widgets on a desktop, or human beings.

Object-oriented languages dominate the programming world because they model real-world objects. UML is a combination of several object-oriented notations: Object-Oriented Design, Object Modeling Technique, and Object-Oriented Software Engineering.

UML uses the strengths of these three approaches to present a more consistent methodology that's easier to use. UML represents best practices for building and documenting different aspects of software and business system modeling.

The UML is popular among programmers but isn't generally used by database developers. One reason is simply that the UML creators did not focus on databases. Despite this, the UML is effective for high-level conceptual data modelling, and it can be used in different types of UML diagrams. You can find information about layering of an object-oriented class model onto a relational database in this article about Database Modelling in UML.

UML meets the following requirements:

- Setting a formal definition of a common Meta-Object Facility (MOF)-based meta-model that specifies the abstract syntax of the UML. The abstract syntax defines the 2 set of UML modeling concepts, their attributes and their relationships, as well as the rules for combining these concepts to construct partial or complete UML models.
- Providing a detailed explanation of the semantics of each UML modeling concept. The semantics define, in a technology independent manner, how the UML concepts are to be realized by computers.
- Specifying the human-readable notation elements for representing the individual UML modeling concepts as well as rules for combining them into a variety of different diagram types corresponding to different aspects of modeled systems.
- Defining ways in which UML tools can be made compliant with this specification. This is supported (in a separate specification) with an XML-based specification of corresponding model interchange formats (XMI) that must be realized by compliant tools.

EXPERIMENT – 2

Aim:

To make Use Case diagram for Automated Teller Machine (ATM)

Theory:

The automated teller machine is simply a data terminal with two input and four output devices. These devices are interfaced to the processor. The processor is heart of the ATM machine. All the ATM machines working around the world are based on centralized database system. The Automated Teller Machine involve following component: -

Card reader - The card reader captures the account information stored on the magnetic stripe on the back of an ATM/debit or credit card.

Keypad - The keypad lets the cardholder tell the bank what kind of transaction is required (cash withdrawal, balance inquiry, etc.) and for what.

Display screen - The display screen prompts the cardholder through each step of the transaction process. Leased-line machines commonly use a monochrome or color CRT (cathode ray tube) display. Dial-up machines commonly use a monochrome or color LCD.

Receipt printer - The receipt printer provides the cardholder with a paper receipt of the transaction.

Cash dispenser - The heart of an ATM is the safe and cash-dispensing mechanism. The entire bottom portion of most small ATMs is a safe that contains the cash.

Transaction Management

i. First, the customer inserts his/her card and PIN into the ATM at a local bank abroad.
ii. The bank then replies to the message via the network, stating that the local bank can provide the customer with the amount requested.
iii. The local bank will not dispense the cash until it knows the funds are available in the customer's own account.
iv. The customer retrieves the money and card and goes on his/her way.
v. As soon as the home country bank receives the request and checks the balance, the money is debited from the account.
vi. Later that day a settlement occurs between the two banks facilitated by the central network.

vii. An electronic message is sent to the central network which is passed on to the customer's own bank in his/her home country.

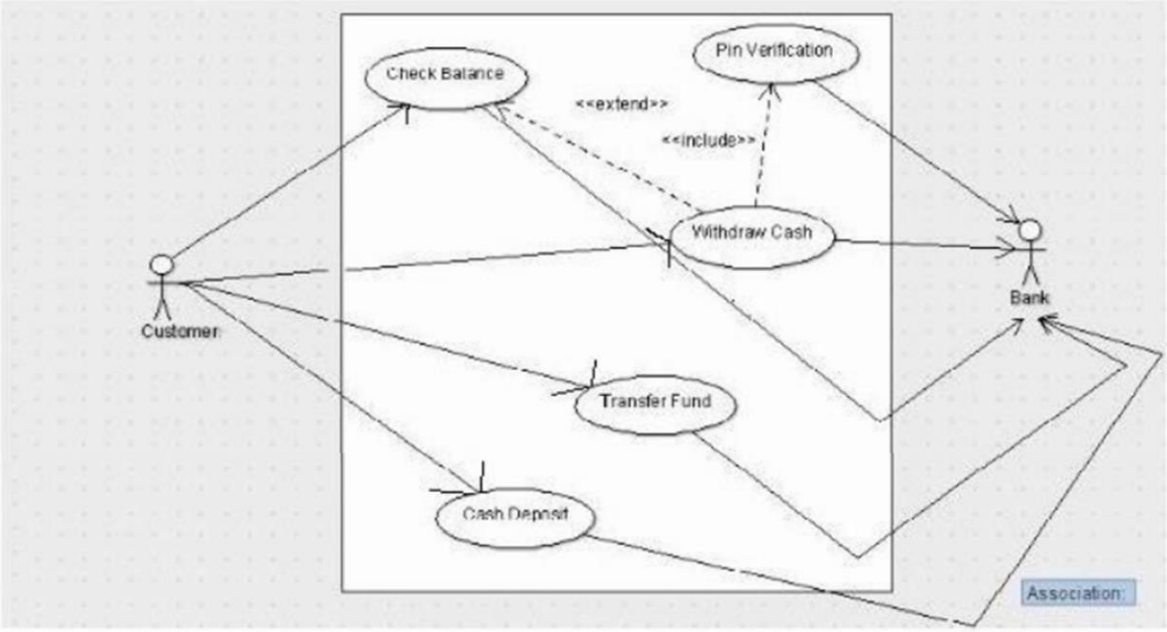

Fig 1- Use Case Diagram

EXPERIMENT – 3

Aim:

To make Sequence diagram for Automated Teller Machine (ATM)

Theory:

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects. For example, lifelines in a sequence diagram for a banking scenario can represent a customer, bank teller, or bank manager. The communication between the customer, teller, and manager are represented by messages passed between them. The sequence diagram shows the objects and the messages between the objects.

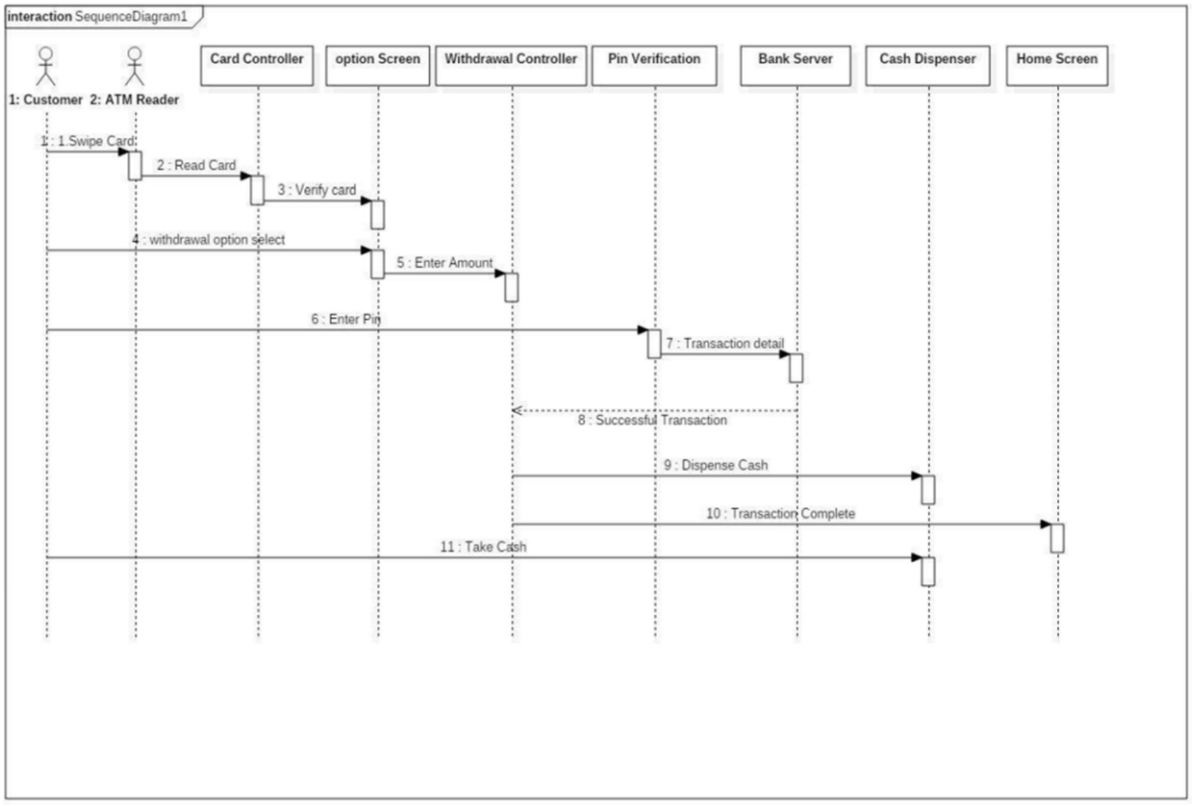

Fig 2-Sequence diagram (Cash Withdrawl)

EXPERIMENT – 4

Aim:

To make Collaboration diagram for Automated Teller Machine (ATM)

Theory:

The collaboration diagram 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. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

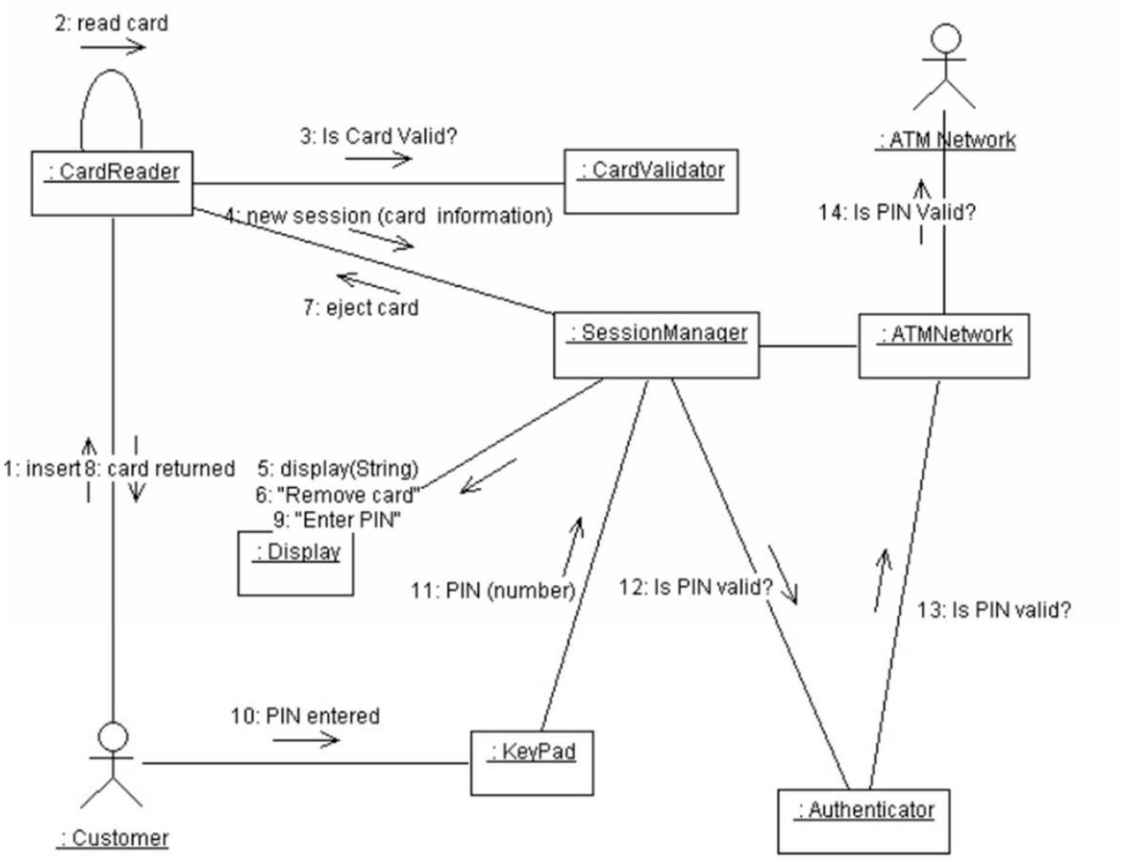

Fig 3-Collaboration diagram

EXPERIMENT – 5

Aim:

To make the Class diagram for Automated Teller Machine (ATM)

Theory:

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

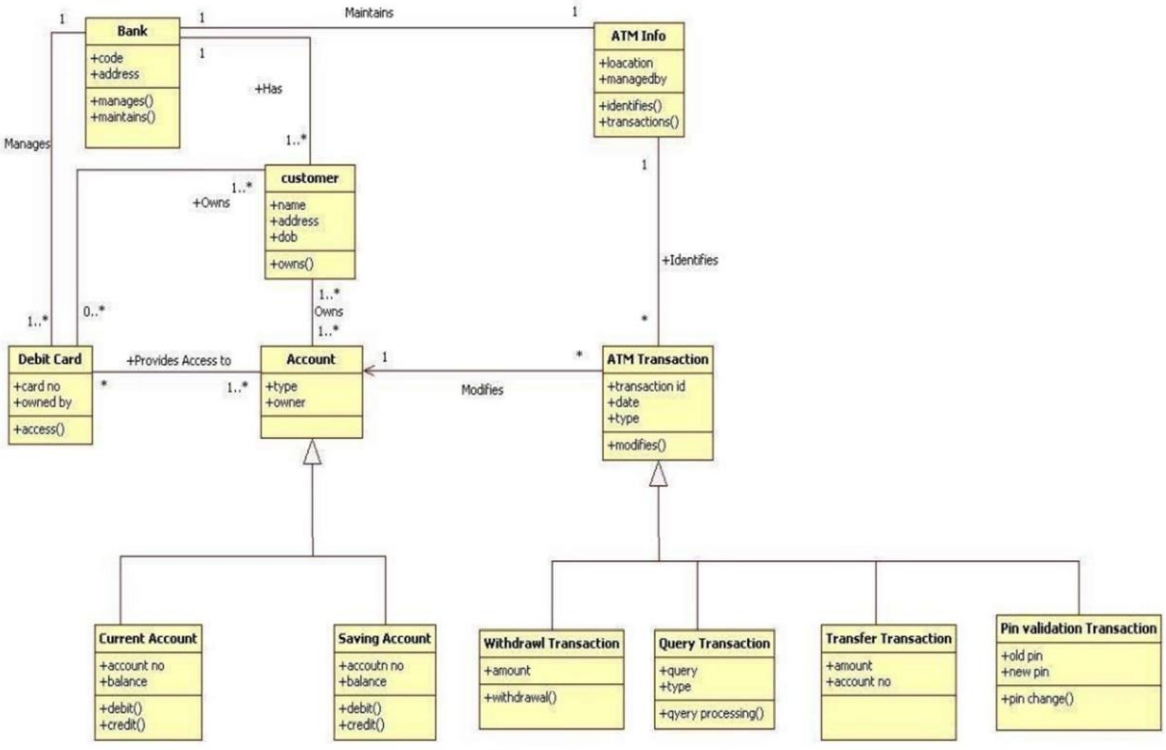

Fig 4- Class Diagram

EXPERIMENT - 6

Aim:

To make the State Chart diagram for Automated Teller Machine (ATM)

Theory:

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state, but we don't model every class using State diagrams. We prefer to model the states with three or more states.

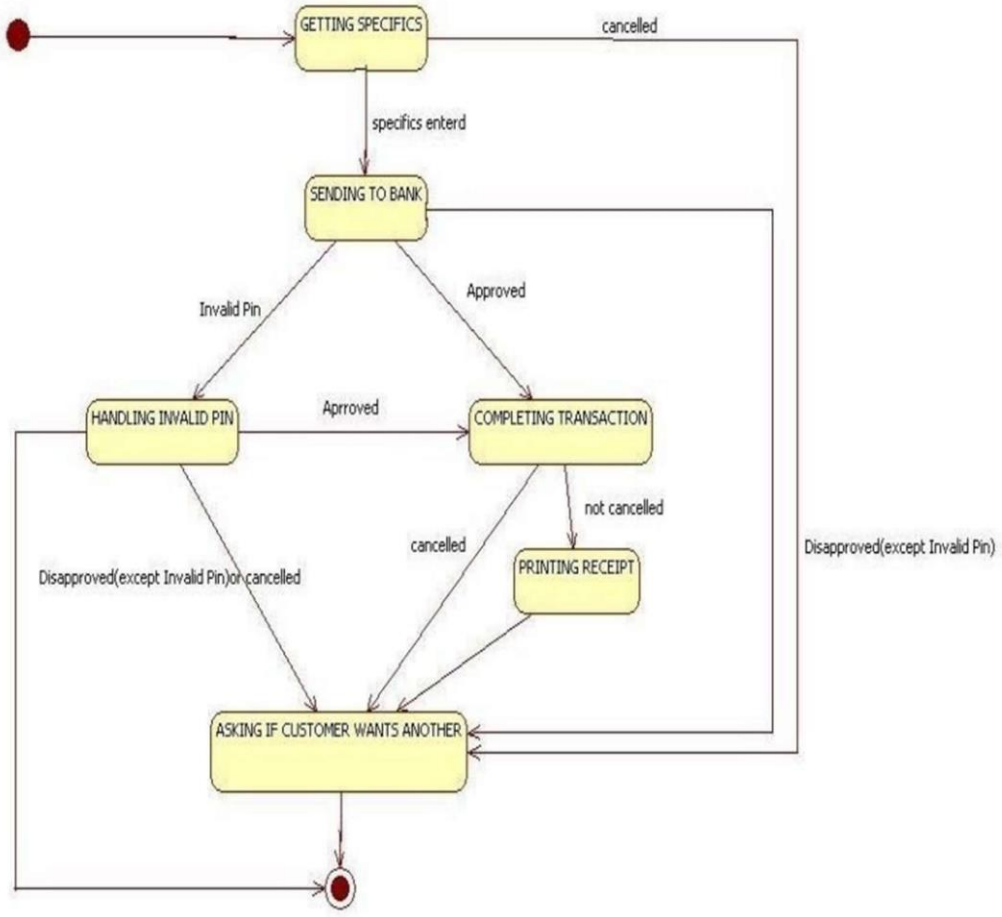

Fig 5-State Diagram

EXPERIMENT – 7

Aim:

To make the Activity diagram for Automated Teller Machine (ATM)

Theory:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

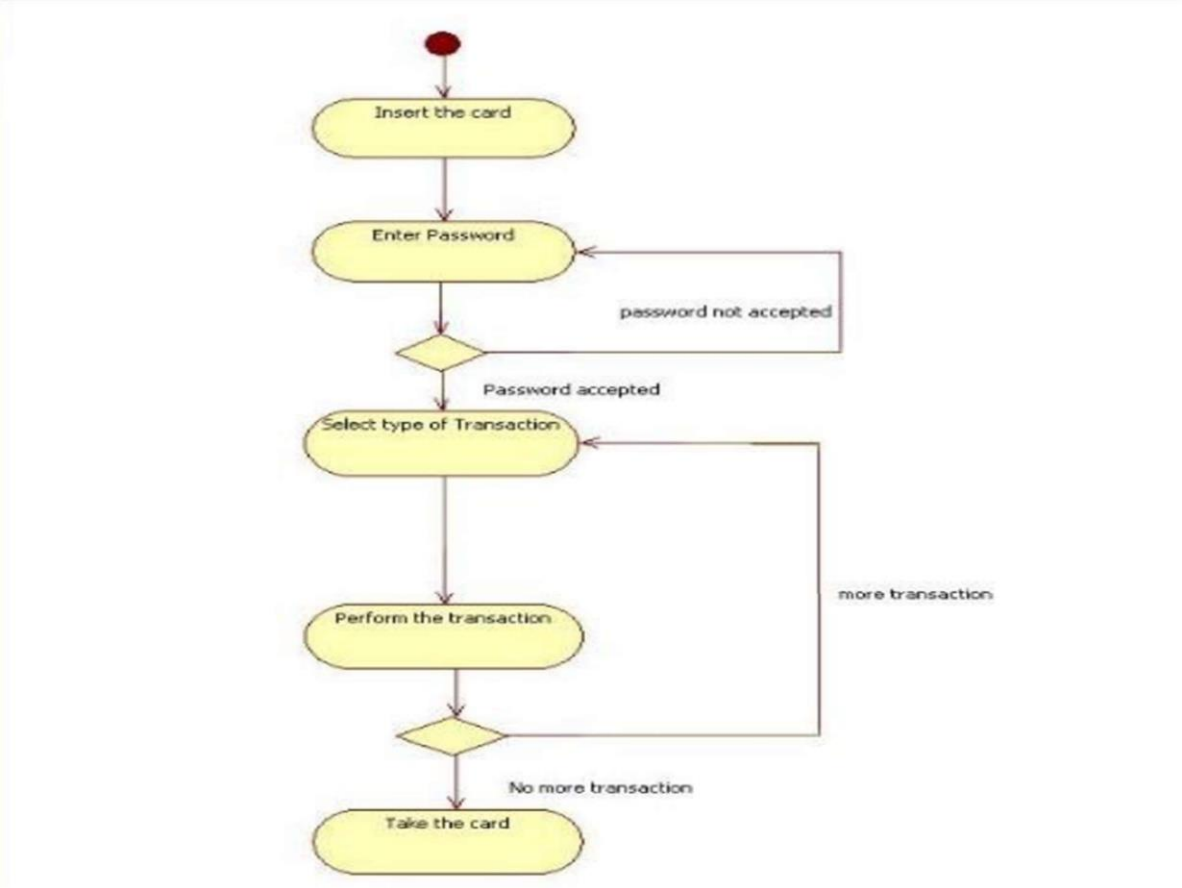

Fig 6- Activity Diagram

EXPERIMENT – 8

Aim:

To make the Component diagram for Automated Teller Machine (ATM)

Theory:

A component diagram is used to break down a large object-oriented system into the smaller components, to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

It visualizes the relationships as well as the organization between the components present in the system. It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable. The implementation details of a component are hidden, and it necessitates an interface to execute a function. It is like a black box whose behaviour is explained by the provided and required interfaces.

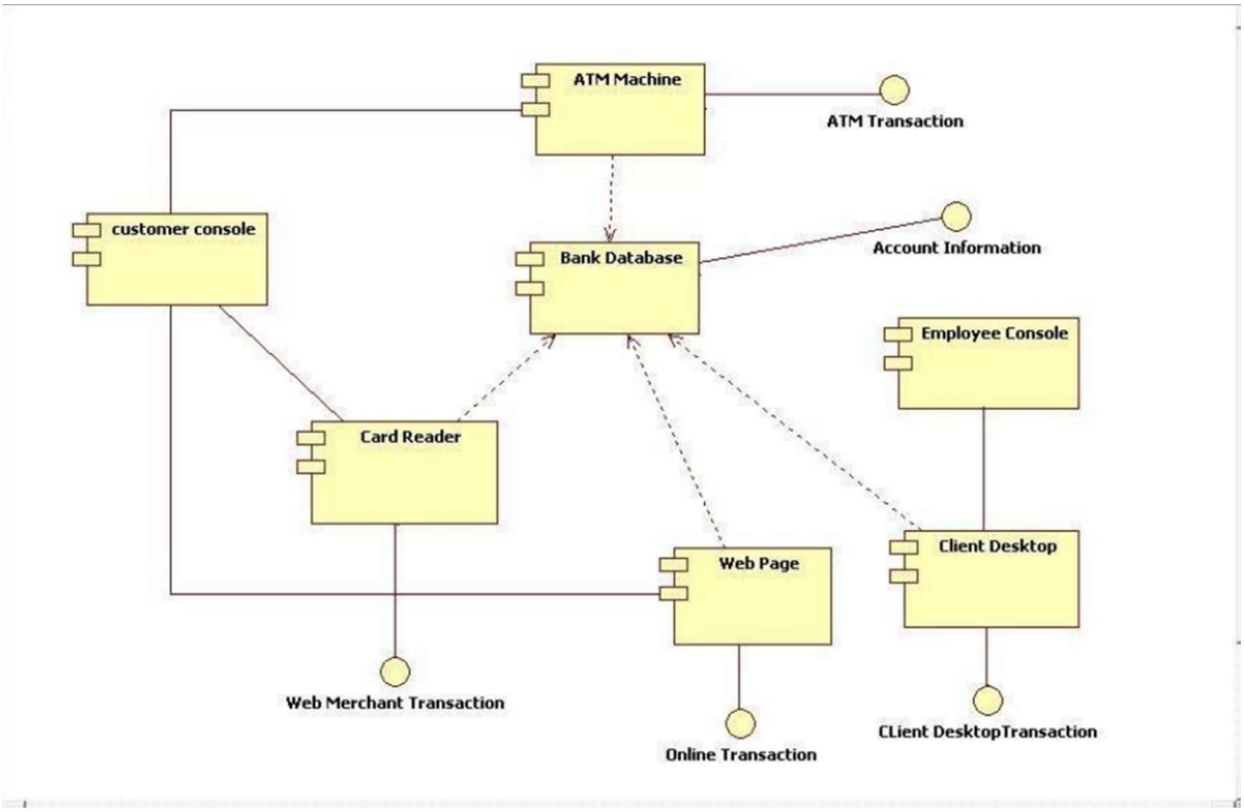

Fig 7- Component Diagram

EXPERIMENT – 9

Aim:

To make the Deployment diagram for Automated Teller Machine (ATM)

Theory:

In UML, deployment diagrams model the physical architecture of a system. Deployment diagrams show the relationships between the software and hardware components in the system and the physical distribution of the processing.

Deployment diagrams, which you typically prepare during the implementation phase of development, show the physical arrangement of the nodes in a distributed system, the artifacts that are stored on each node, and the components and other elements that the artifacts implement. Nodes represent hardware devices such as computers, sensors, and printers, as well as other devices that support the runtime environment of a system. Communication paths and deploy relationships model the connections in the system.

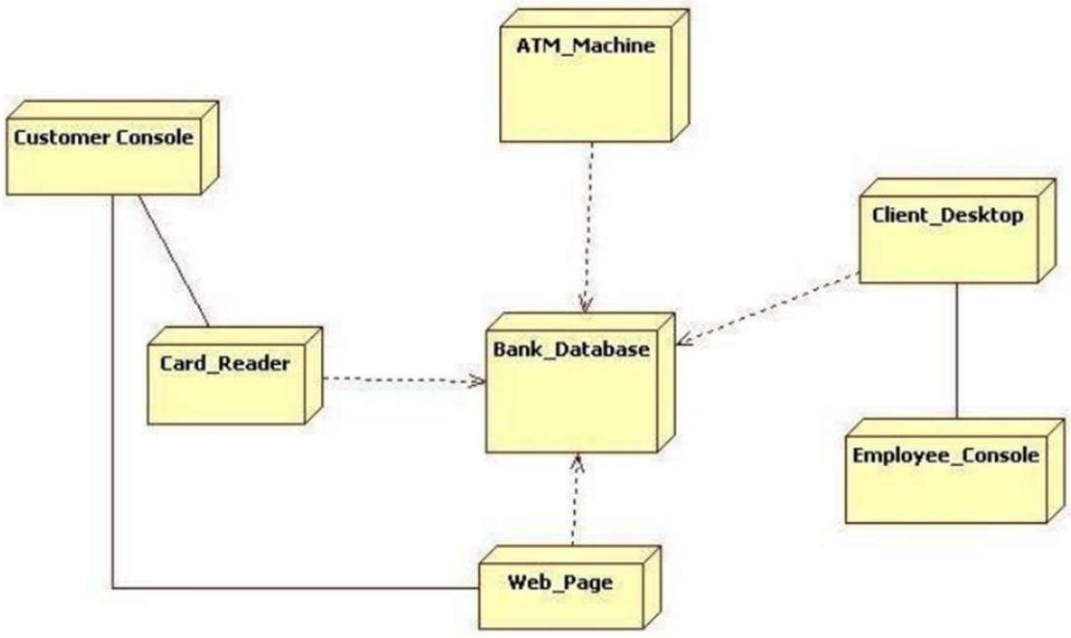

Fig 8-Deployment diagram