Software Engineering [IT301]

Practical File

Submitted By:

Anchal kumari

A12405218083

6CSE-3X

Submitted To:

Dr Rakesh Garg

AMITY SCHOOL OF ENGINEERING & TECHNOLOGY

AMITY UNIVERSITY CAMPUS, SECTOR- 125, NOIDA- 201303

Index

Ex No.	*Name of Program*	*Date of Allotment of Program*	*Date of Evaluation*	*Maximum Marks*	*Marks Obtained*	*Signature of faculty*
1.	To draw a Class Diagram for an ATM Machine System	08/01/2021	15/01/2021			
2.	To draw a Use Case Diagram for an ATM Machine System	15/01/2021	22/01/2021			
3.	To draw a Statechart Diagram for an ATM Machine System	22/01/2021	29/01/2021			
4.	To draw an Object Diagram for an ATM Machine System	29/01/2021	05/02/2021			
5.	To draw an Activity Diagram for an ATM Machine System	05/02/2021	12/02/2021			
6.	To draw a Sequence Diagram for an ATM Machine System	12/02/2021	19/02/2021			
7.	To draw a Component Diagram for an ATM Machine System	19/02/2021	26/02/2021			
8.	To draw a Collaboration Diagram for an ATM Machine System	26/02/2021	05/03/2021			

Professional Skills Development Activities (PSDA)						
1.	PSDA 1: Automatic Washing Machine	05/03/2021	19/03/2021			
2.	PSDA 2: Automatic Food Vending Machine	19/03/2021	26/03/2021			

Program 1

AIM: To draw a Class Diagram for an ATM Machine System

THEORY:

A **Class Diagram** in the Unified Modelling Language (UML) is a type of **static structure** diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

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 modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

A UML class diagram is made up of:

- A set of **classes** and
- A set of **relationships** between classes

A class notation consists of three parts:

- **Class Name:** The name of the class appears in the first partition.
- **Class Attributes:** Attributes are shown in the second partition, and attribute type is shown after the colon. Attributes map onto member variables (data members) in code.
- **Class Operations** (Methods)**:** Operations are shown in the third partition. They are the services that the class provides. The return type of a method is shown after the colon at the end of the method signature. The return type of method parameters is shown after the colon following the parameter name. Operations map onto class methods in code

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

- Inheritance or Generalization
- Simple Association
- Aggregation
- Composition
- Dependency

ANCHAL KUMARI
A12405218083
6CSE-3X

DIAGRAM:

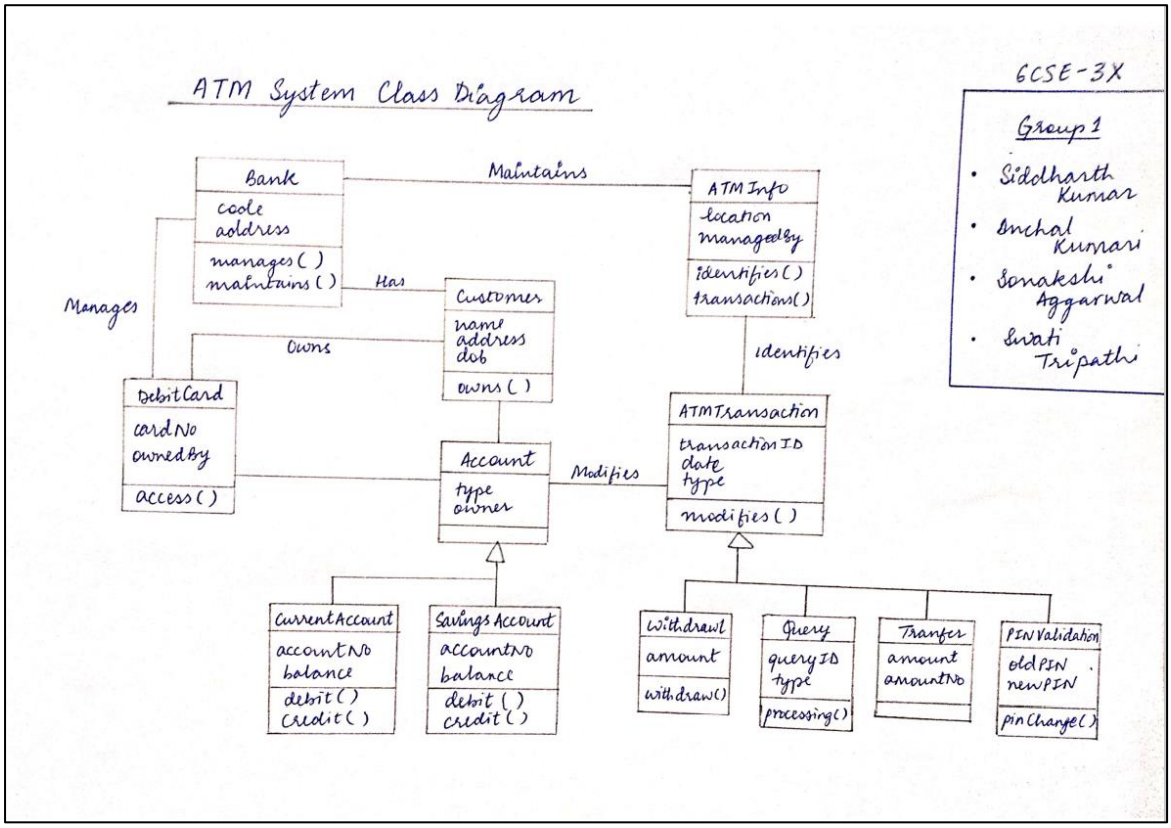


Fig: Hand-drawn Class Diagram

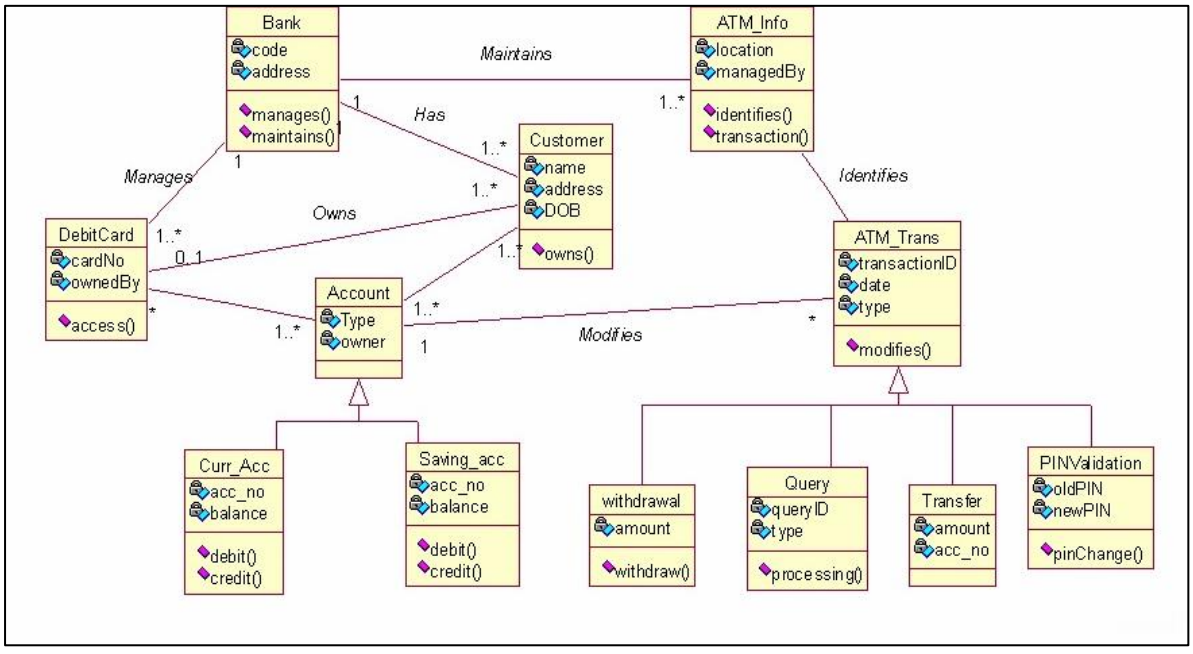


Fig1: Class Diagram made on Rational Rose software

Program 2

AIM: To draw a Use Case Diagram for an ATM Machine System

THEORY:

To model a system, the most important aspect is to capture the **dynamic behaviour**. Dynamic behaviour means the behaviour of the system when it is running/operating.

Only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour. The purpose of **Use Case diagram** is to capture the dynamic aspect of a system.

Use Case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analysed to gather its functionalities, use cases are prepared and actors are identified.

Use cases represent only the functional requirements of a system. Other requirements such as business rules, quality of service requirements, and implementation constraints must be represented separately, again, with other UML diagrams.

A use case notation consists of the following components:

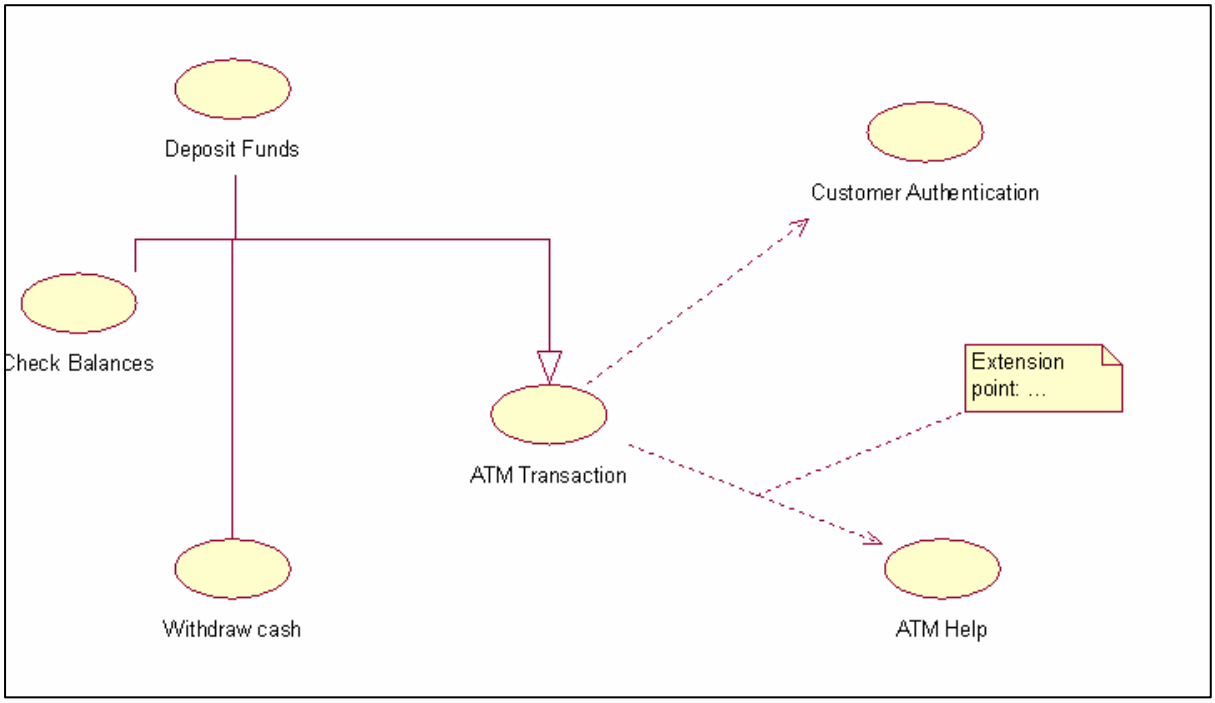
- **Actor:** Someone who interacts with a use case, Actor plays a particular role in the business. The concept of an Actor is similar to a user, the difference being that a user can play different roles. Actor triggers use case(s) and has a responsibility toward the system (inputs). Actor has expectations from the system (outputs).
- **Use Case:** System function (process - automated or manual). Each Actor must be linked to a use case, while some use cases may not be linked to actors.
- **Communication Link:** The participation of an actor in a use case is shown by connecting an actor to a use case by a **solid link (straight line)**. Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.
- **Boundary of System:** The system boundary is potentially the entire system as defined in the requirements document. For large and complex systems, each module may be the system boundary.

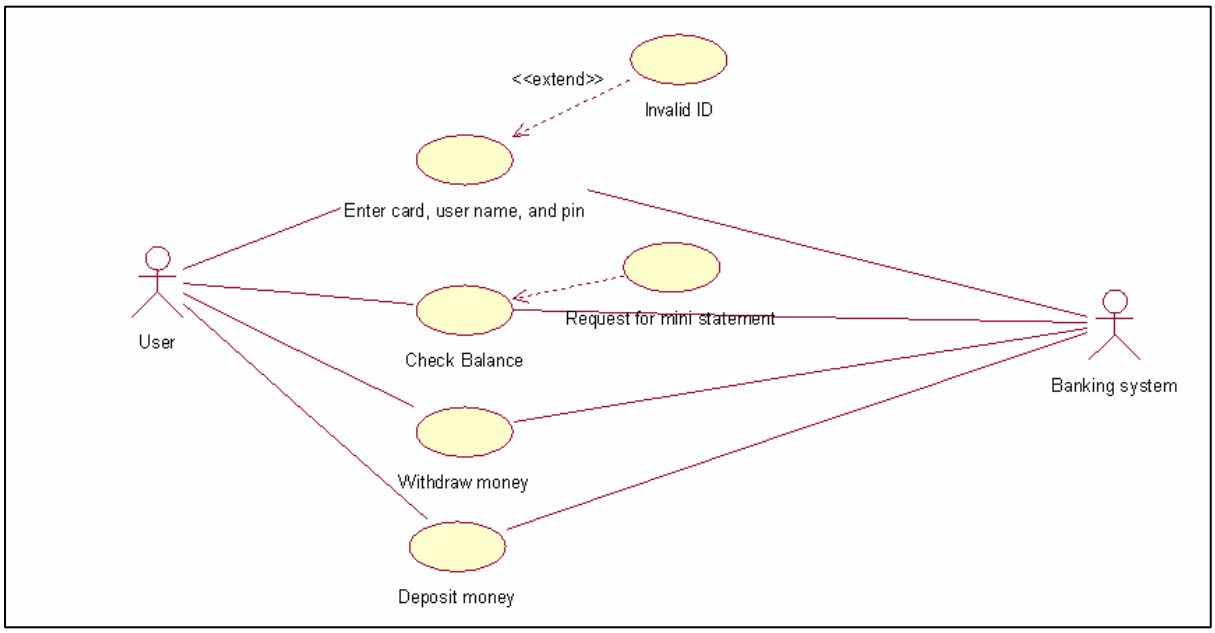
Use cases share different kinds of relationships. Defining the relationship between two use cases is the decision of the software analysts of the use case diagram. A relationship between two use cases is basically modelling the dependency between the two use cases. Use case relationships are listed as the following:

- Extends
- Include
- Generalisation

ANCHAL KUMARI
A12405218083
6CSE-3X

DIAGRAM:





ANCHAL KUMARI
A12405218083
6CSE-3X

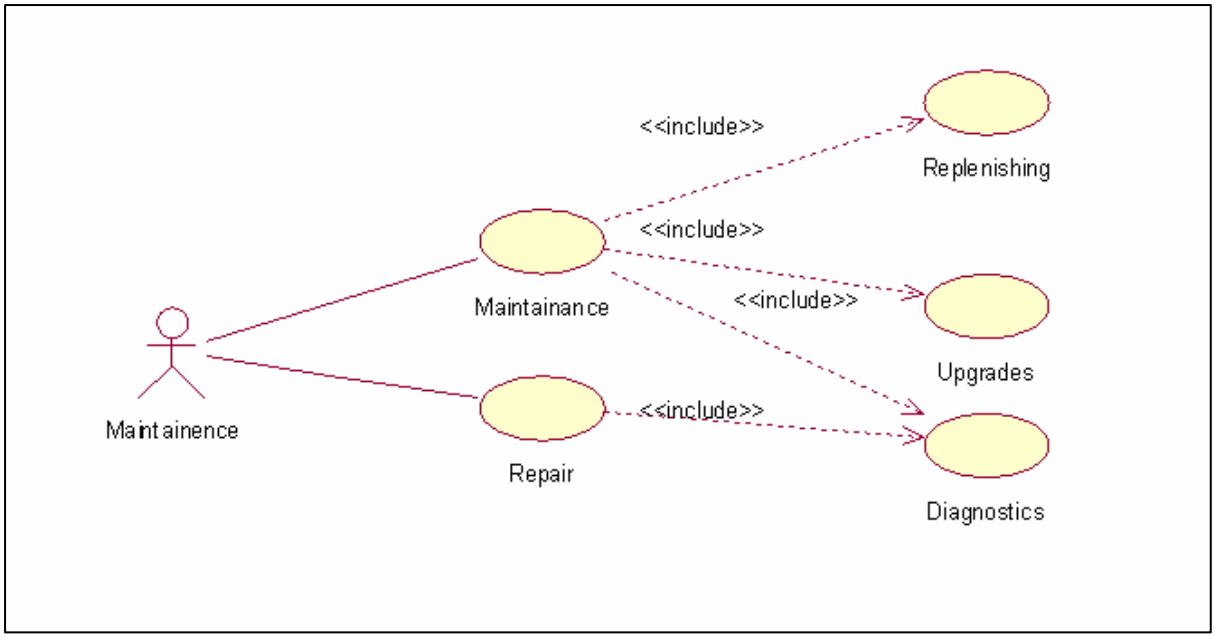


Fig2: Use Case Diagram made on Rational Rose software

ANCHAL KUMARI
A12405218083
6CSE-3X

Program 3

AIM: To draw a Statechart Diagram for an ATM Machine System

THEORY:

A **Statechart diagram** describes different states of a component in a system. The states are specific to a component/object of a system. In other words, a **Statechart diagram** describes a state machine.

A **State Machine** can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. Statechart diagrams are, thus, also known as State Machine diagrams.

State machine diagrams, typically, are used to describe **state-dependent behaviour** for an object. An object responds differently to the same event depending on what state it is in. State machine diagrams are usually applied to objects but can be applied to any element that has behaviour to other entities such as: actors, use cases, methods, subsystems systems and etc. and they are typically used in conjunction with interaction diagrams (usually sequence diagrams).

A **state** is a constraint or a situation in the life cycle of an object, in which a constraint holds, the object executes an activity or waits for an event. There are several characteristics of states in general, regardless of their types:

- A state occupies an interval of time.
- A state is often associated with an abstraction of attribute values of an entity satisfying some condition(s).
- An entity changes its state not only as a direct consequence of the current input, but it is also dependent on some past history of its inputs.

A state machine diagram is a diagram consisting of:

- States (simple states or composite states)
- State transitions connecting the states

An event signature is described as **Event-name** (comma-separated-parameter-list). Events appear in the internal transition compartment of a state or on a transition between states. Events pass information, which is elaborated by Objects operations. Objects realize Events.

An event may be one of four types:

- **Signal event:** corresponding to the arrival of an asynchronous message or signal
- **Call event:** corresponding to the arrival of a procedural call to an operation
- **Time event:** a time event occurs after a specified time has elapsed
- **Change event:** a change event occurs whenever a specified condition is met

ANCHAL KUMARI
A12405218083
6CSE-3X

DIAGRAM:

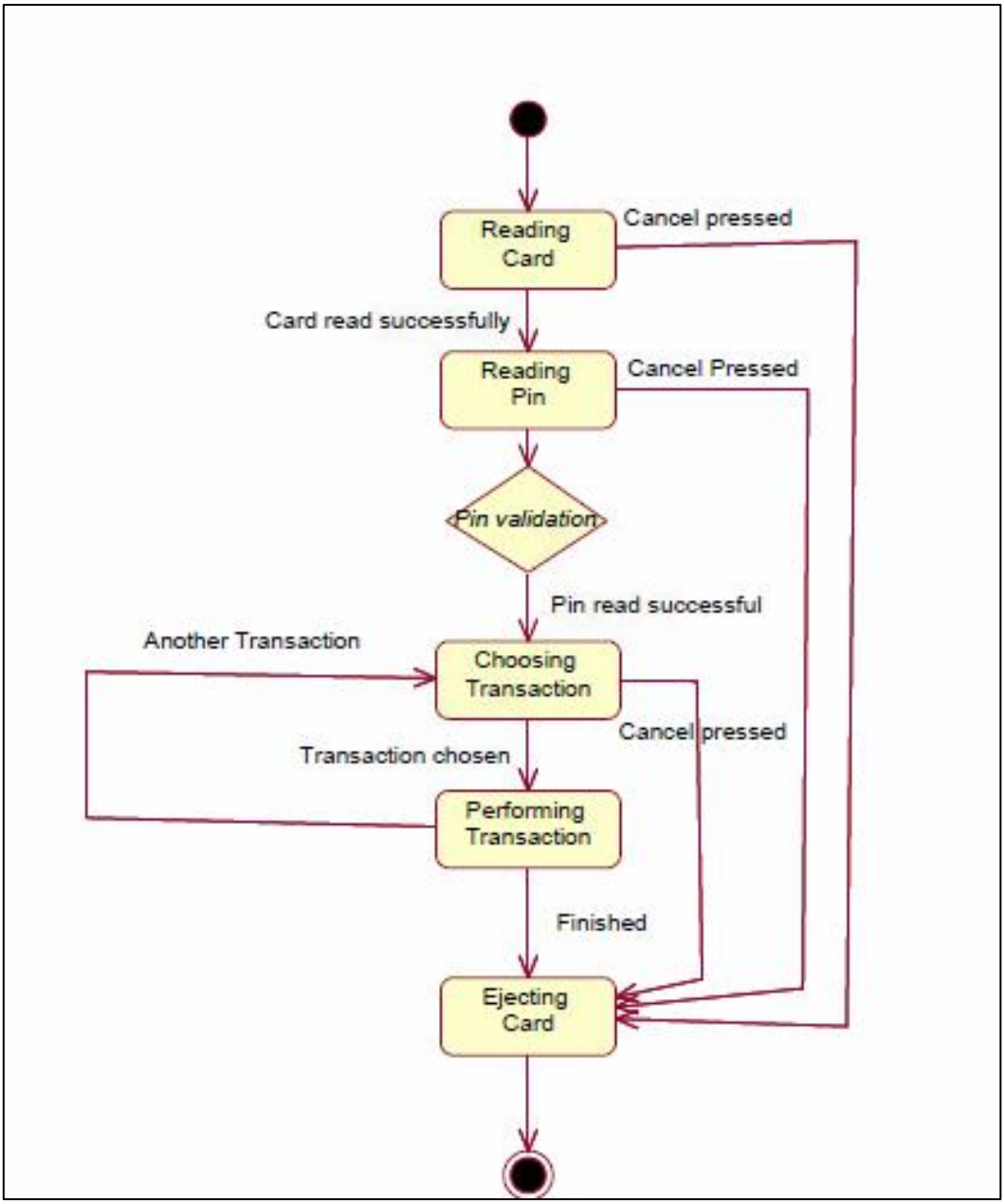


Fig3: Statechart Diagram made on Rational Rose software

ANCHAL KUMARI
A12405218083
6CSE-3X

Program 4

AIM: To draw an Object Diagram for an ATM Machine System

THEORY:

An Object is an instance of a class, at a particular moment in runtime, that can have its own state and data values. Likewise, a static UML **Object diagram** is an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Thus, as object diagrams are derived from class diagrams, object diagrams are dependent upon class diagrams.

An object diagram notation consists of three parts:

- **Object Name:** Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined, divided with a colon.
- **Object Attributes:** Similar to classes, object attributes can be listed inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.
- **Links:** Links tend to be instances associated with associations. A link can be drawn while using the lines utilized in class diagrams.

DIAGRAM:

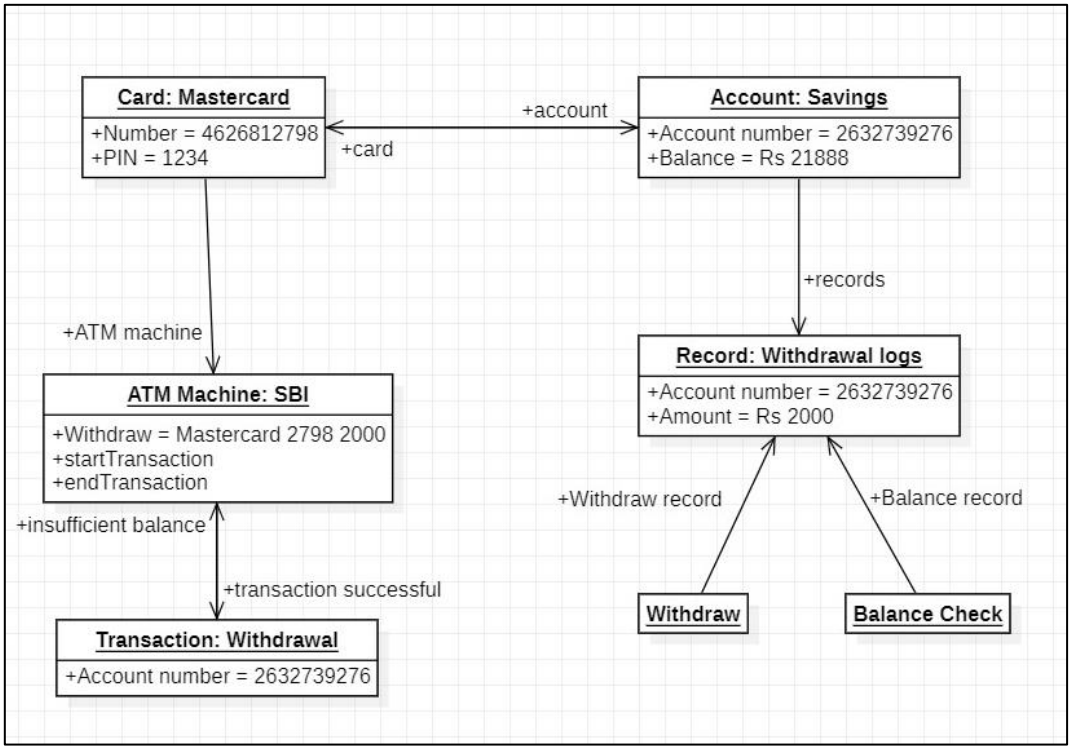


Fig4: Object Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

Program 5

AIM: To draw an Activity Diagram for an ATM Machine System

THEORY:

Activity diagram is another important behavioural diagram in UML diagram to describe **dynamic aspects** of the system. Activity diagram is essentially an advanced version of flow chart that modelling the flow from one activity to another activity.

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases co-ordinate to represent business workflows

An activity diagram notation consists of the following components:

- **Activity:** The categorization of behaviour into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.
- **Swimlane and Partition:** The swimlane is used to cluster or group all the related activities in one column or one row. It can be either vertical or horizontal. It is used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.
- **Fork Node:** A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.
- **Join Node:** Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.

An Activity diagram constitutes the following notations:

- **Initial Node:** It depicts the initial stage or beginning of a set of actions or activities.
- **Activity Final Node:** It is the stage where all the control flows and object flows end.
- **Object Node:** It represents an object that is connected to a set of Object Flows
- **Decision Node:** It represents a test condition to ensure that the control flow or object flow only goes down one path.

ANCHAL KUMARI
A12405218083
6CSE-3X

- **Action:** It represents the set of actions or tasks that are to be performed.
- **Control Flow:** Shows the sequence of execution
- **Object Flow:** Show the flow of an object from one activity (or action) to another activity (or action).

DIAGRAM:

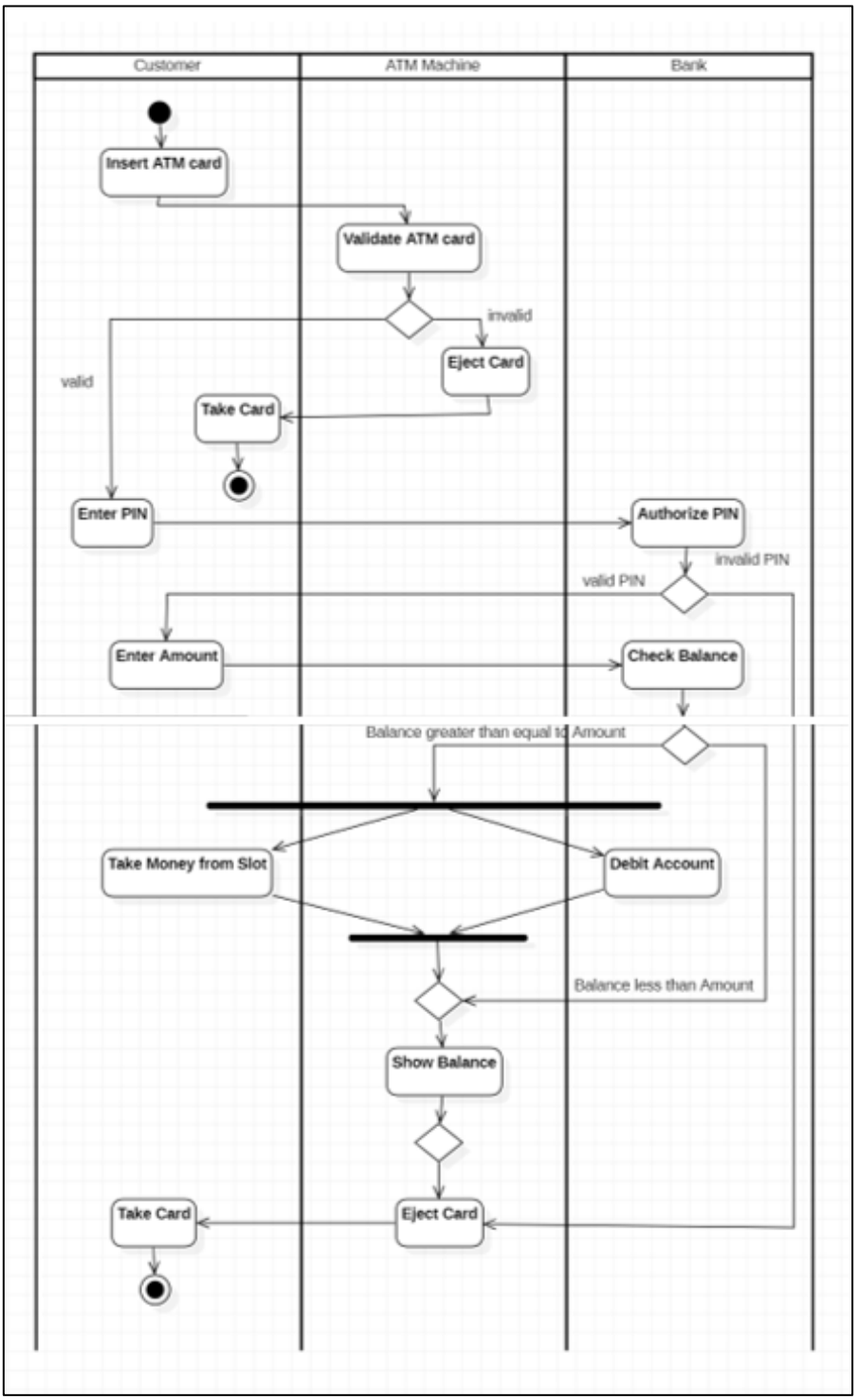


Fig5: Activity Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

Program 6

AIM: To draw a Sequence Diagram for an ATM Machine System

THEORY:

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are **time focus** and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

The sequence diagram represents the flow of messages in the system. It helps in envisioning several dynamic scenarios. It portrays the **communication between any two lifelines** as a time-ordered sequence of events, such that these lifelines took part at the run time.

An activity diagram notation consists of the following components:

- **Actor:** A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.
- **Lifeline:** An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.
- **Activation:** It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.
- **Messages:** The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

  Following are the types of messages:

  - **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.
  - **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.
  - **Self-Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.
  - **Recursive Message:** A self-message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self-message as it represents the recursive calls.

ANCHAL KUMARI
A12405218083
6CSE-3X

- **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.
- **Destroy Message:** It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.
- **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modelling a system.

- **Note:** A note is the capability of attaching several remarks to the element. It basically carries useful information for the mode

DIAGRAM:

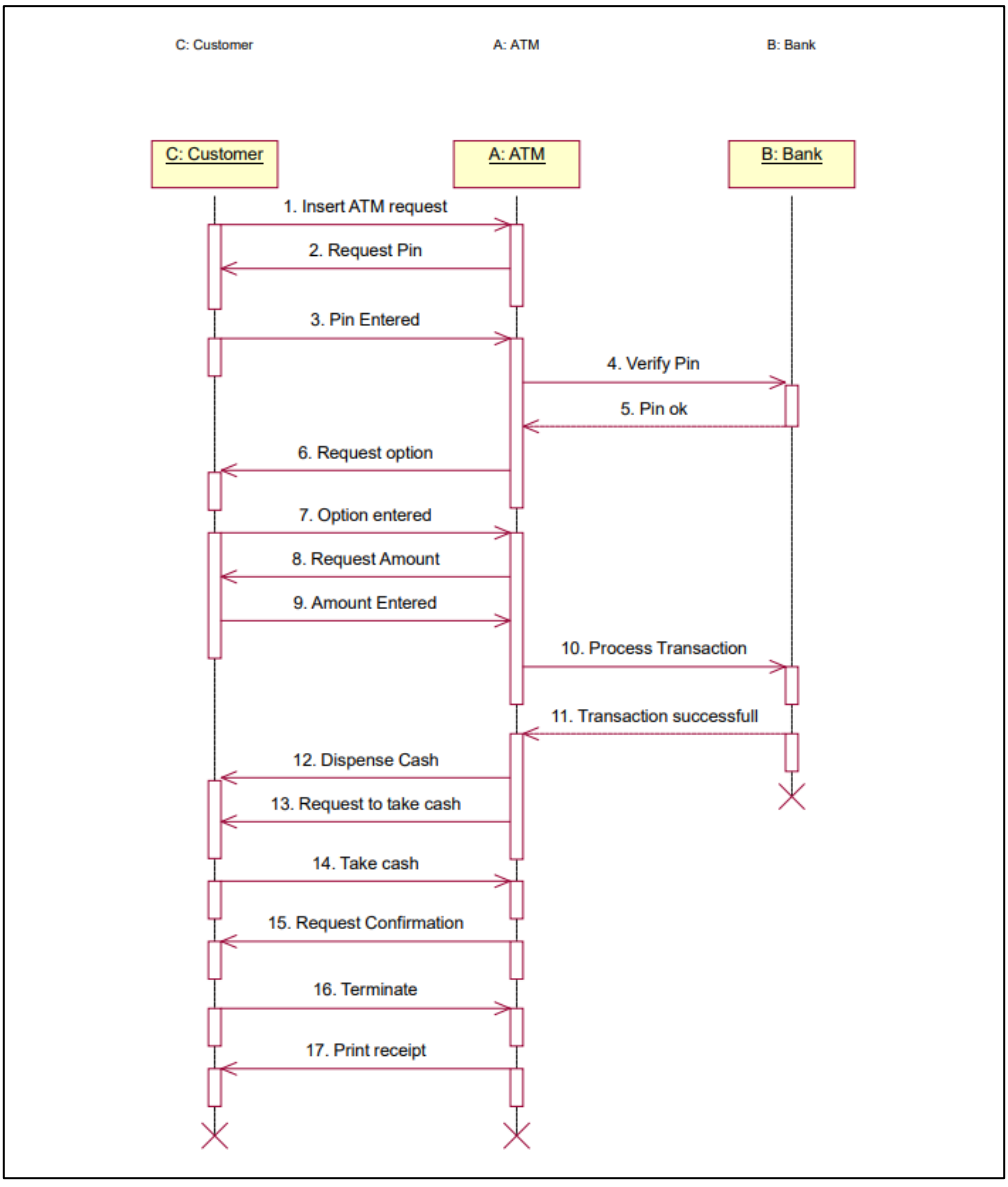


Fig6: Sequence Diagram made on Rational Rose software

Program 7

AIM: To draw a Component Diagram for an ATM Machine System

THEORY:

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are **time focus** and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

The sequence diagram represents the flow of messages in the system. It helps in envisioning several dynamic scenarios. It portrays the **communication between any two lifelines** as a time-ordered sequence of events, such that these lifelines took part at the run time.

DIAGRAM:

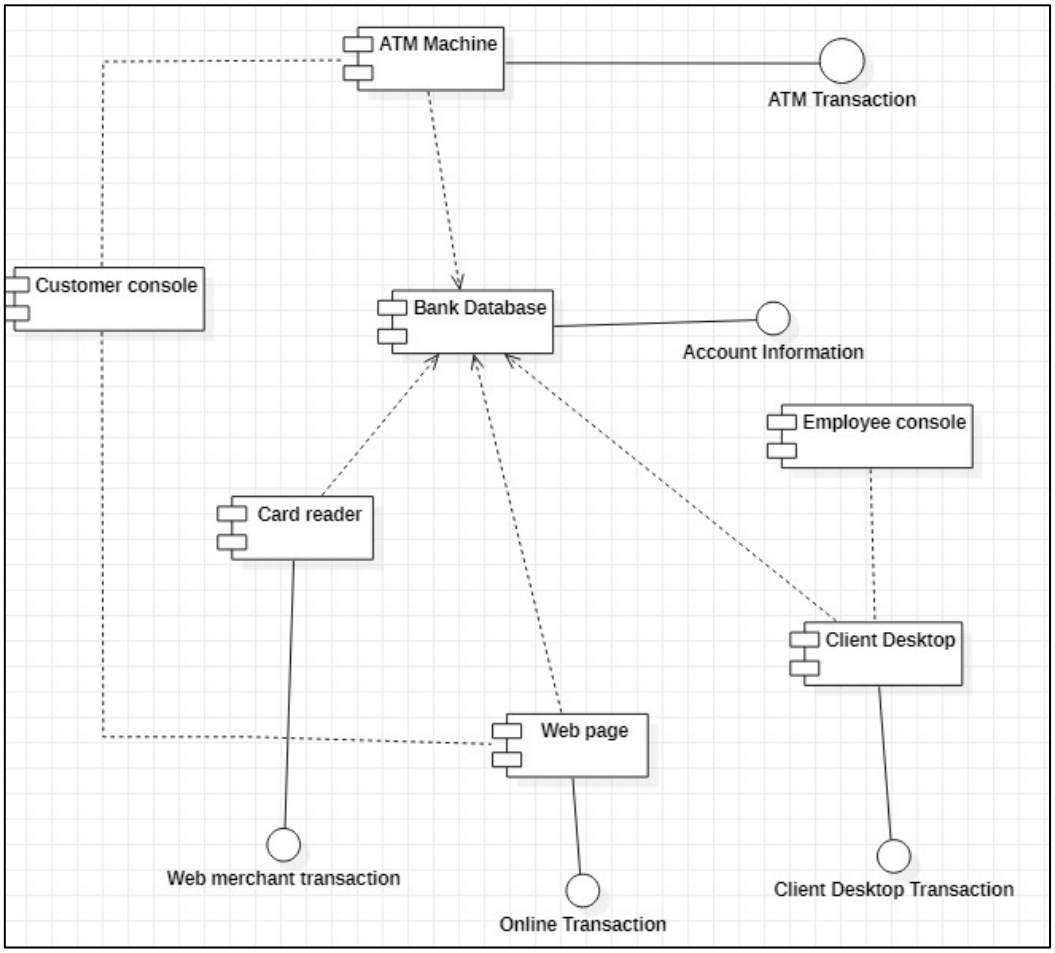


Fig7: Component Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

Program 8

AIM: To draw a Collaboration Diagram for an ATM Machine System

THEORY:

Collaboration diagrams (also known as Communication diagrams) are used to show how objects interact to perform the behaviour of a particular use case, or a part of a use case. Along with sequence diagrams, collaborations are used by designers to define and clarify the roles of the objects that perform a particular flow of events of a use case. They are the primary source of information used to determining class responsibilities and interfaces.

A collaboration diagram is an illustration of the relationships and interactions among **software** objects in UML. These diagrams can be used to portray the dynamic behaviour of a particular use case and define the role of each object.

A collaboration diagram notation consists of the following components:

- **Objects**: An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon. Each object in the collaboration is named and has its class specified. There may be more than one object of a class. An object's class can be unspecified.
- **Actors:** A role played by an entity that interacts with the subject is called as an actor. Normally an actor instance occurs in the collaboration diagram as the invoker of the interaction. Each Actor is named and has a role. One actor will be the initiator of the use case
- **Links:** Links connect objects and actors, and are instances of associations. Each link corresponds to an association in the class diagram. A link is a relationship among objects across which messages can be sent. In collaboration diagrams, a link is shown as a solid line between two objects. An object interacts with, or navigates to, other objects through its links to these objects. Message flows are attached to links
- **Messages:** A message is a communication between objects that conveys information with the expectation that activity will ensue. In collaboration diagrams, a message is shown as a **labelled arrow placed near a link**. The message is directed from sender to receiver. The receiver must understand the message. The association must be navigable in that direction.

ANCHAL KUMARI
A12405218083
6CSE-3X

DIAGRAM:

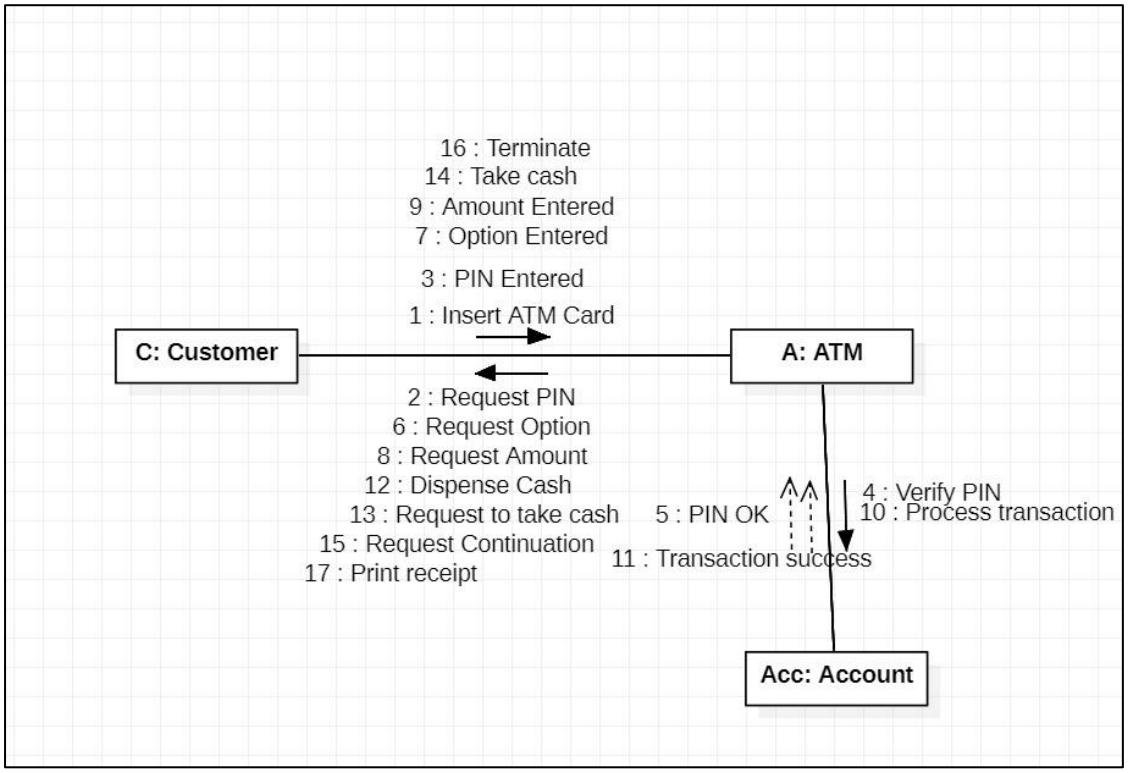


Fig8: Collaboration Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

Professional Skills Development Activities (PSDA)

ANCHAL KUMARI
A12405218083
6CSE-3X

PSDA: 1

AIM: To draw all UML diagrams for an Automatic Washing Machine

DIAGRAMS:

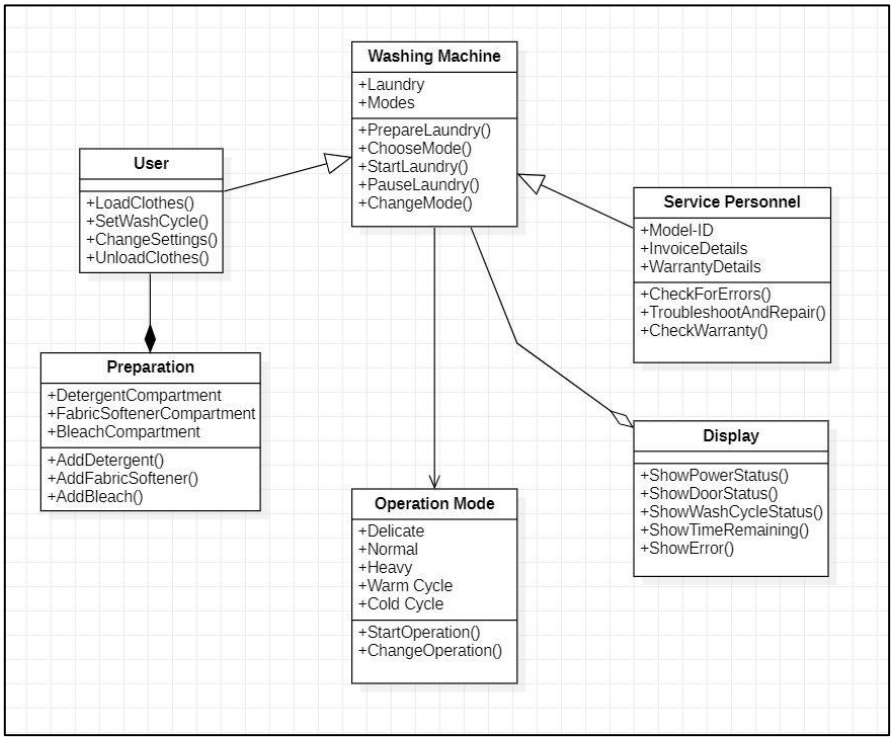


Fig9: Class Diagram made on Staruml software

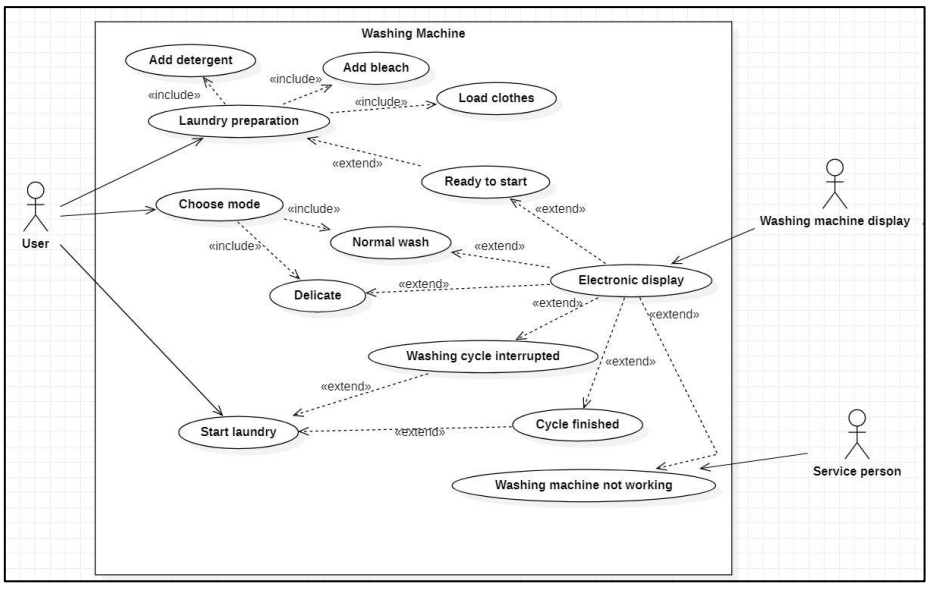


Fig10: Use Case Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

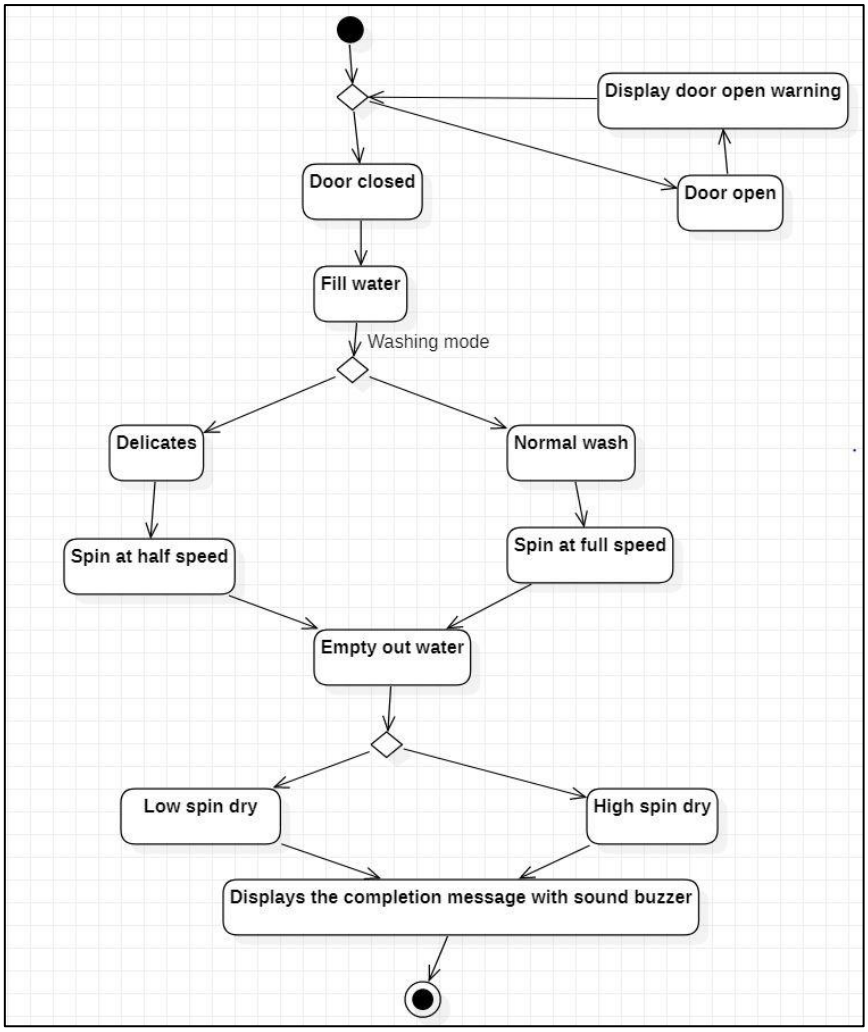


Fig11: Statechart Diagram made on Staruml software

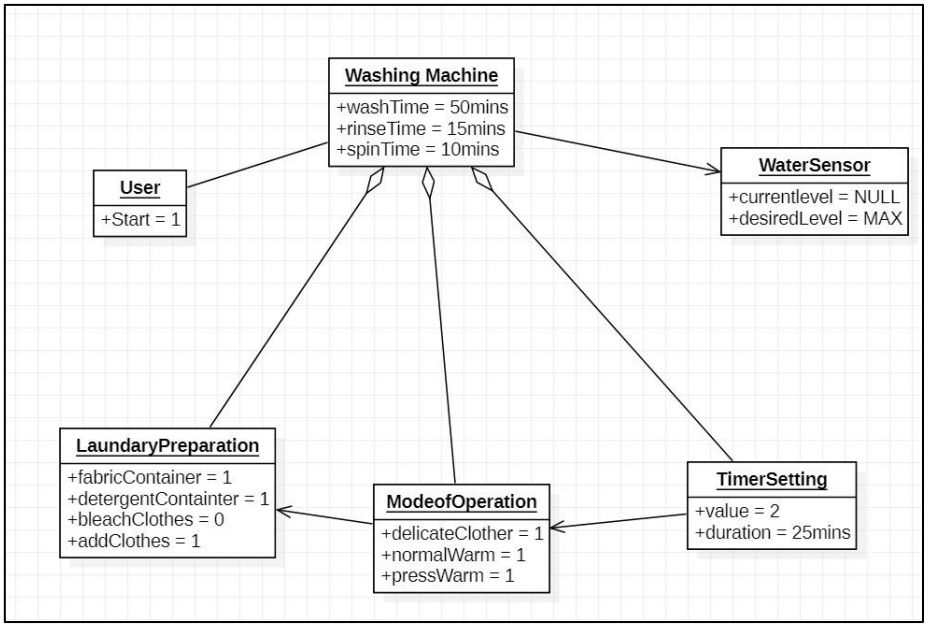


Fig12: Object Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

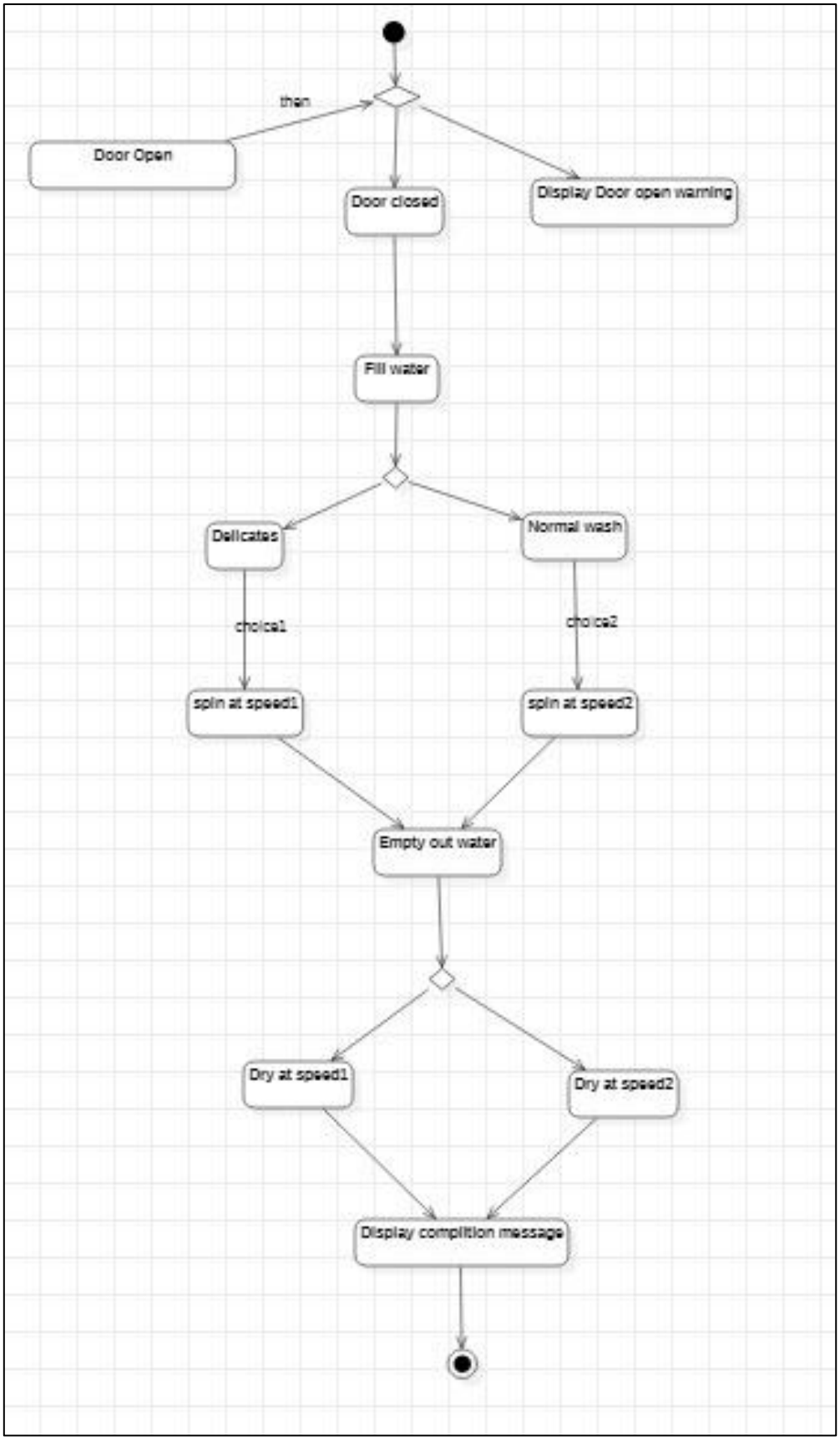


Fig13: Activity Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

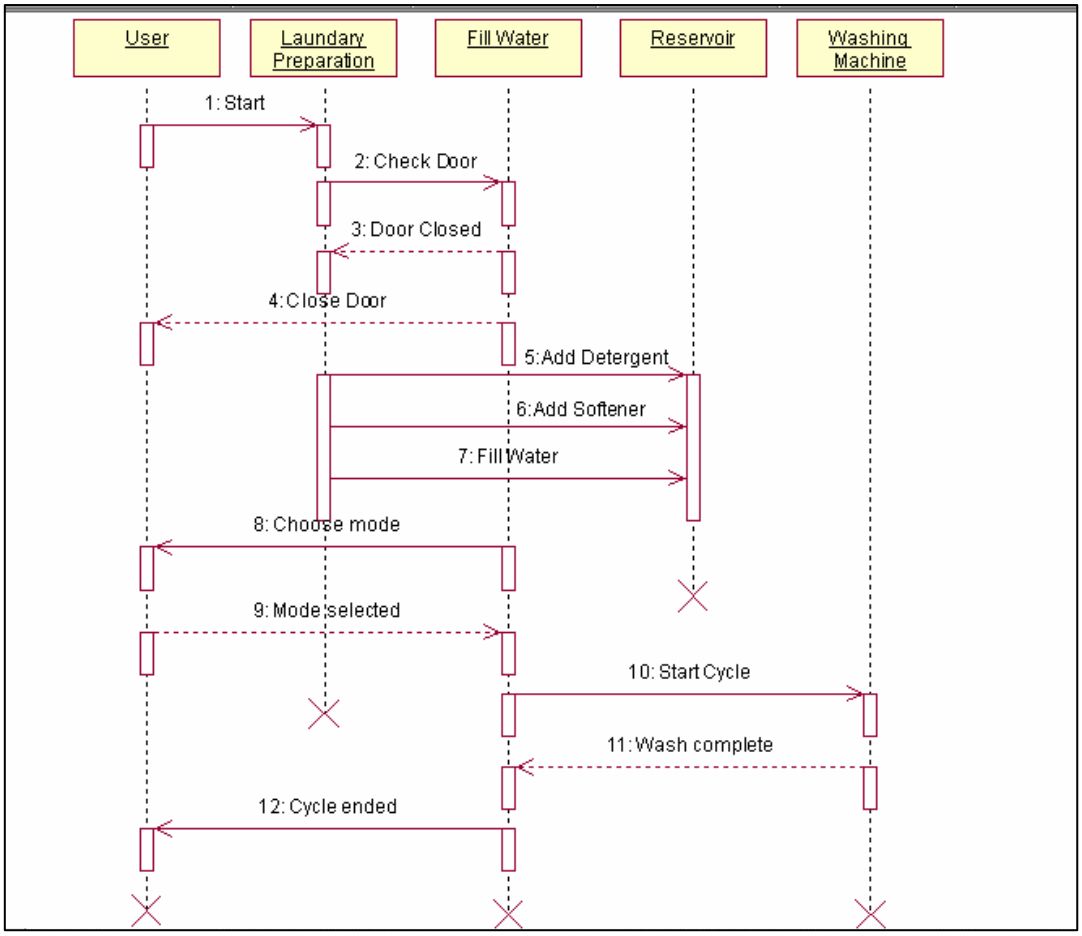


Fig14: Sequence Diagram made on Rational Rose software

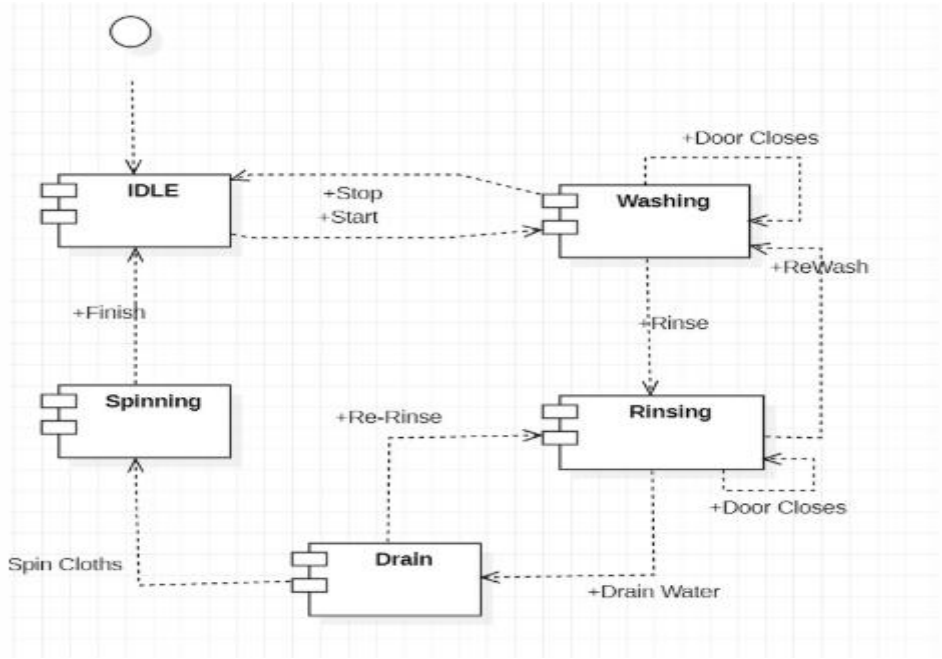


Fig15: Component Diagram made on Rational Rose software

ANCHAL KUMARI
A12405218083
6CSE-3X

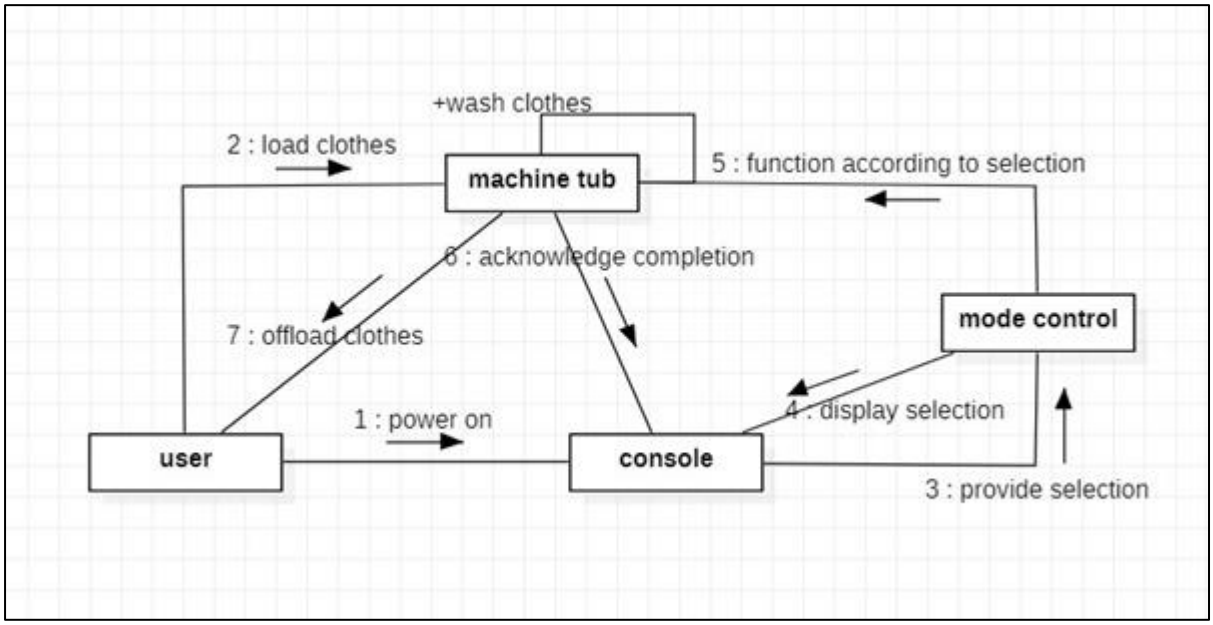


Fig16: Collaboration Diagram made on Rational Rose software

ANCHAL KUMARI
A12405218083
6CSE-3X

PSDA: 2

AIM: To draw all UML diagrams for an Automatic Food Vending Machine

DIAGRAMS:

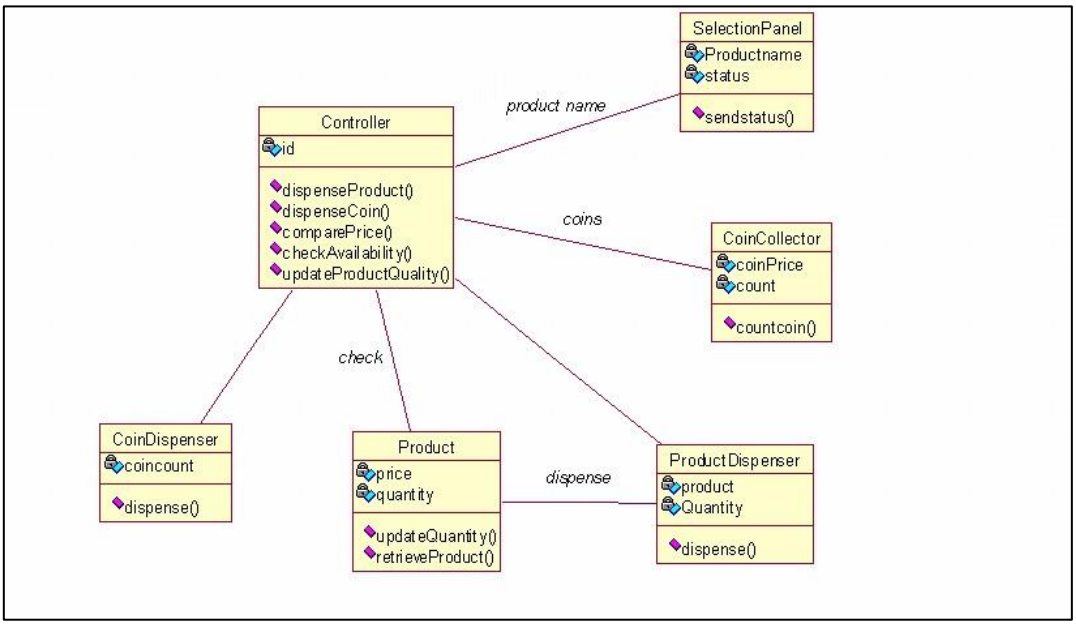


Fig17: Class Diagram made on Rational Rose software

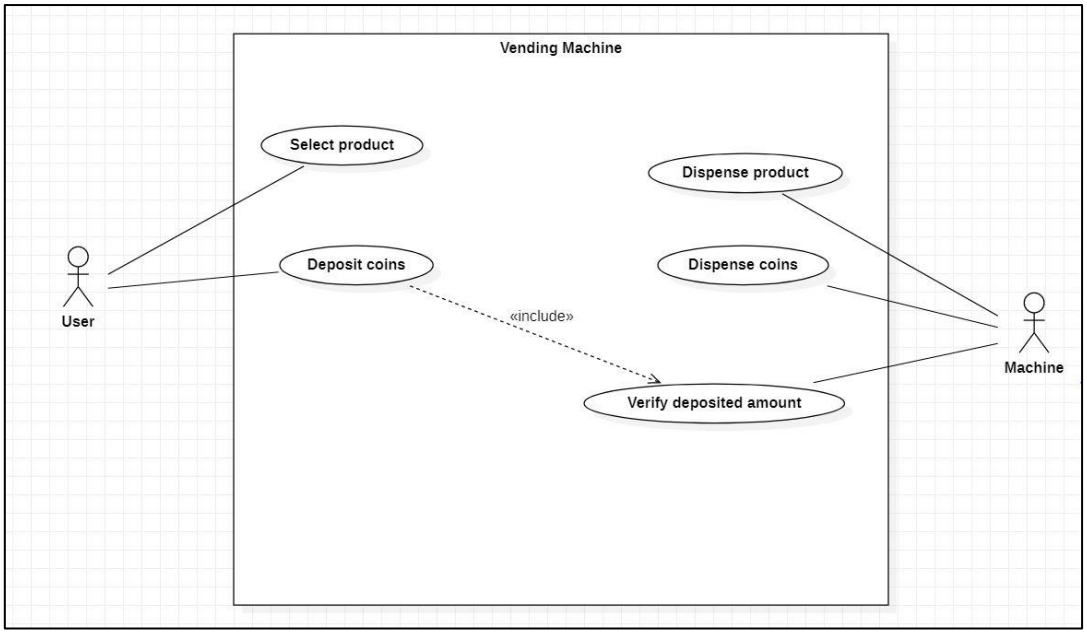


Fig18: Use Case Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

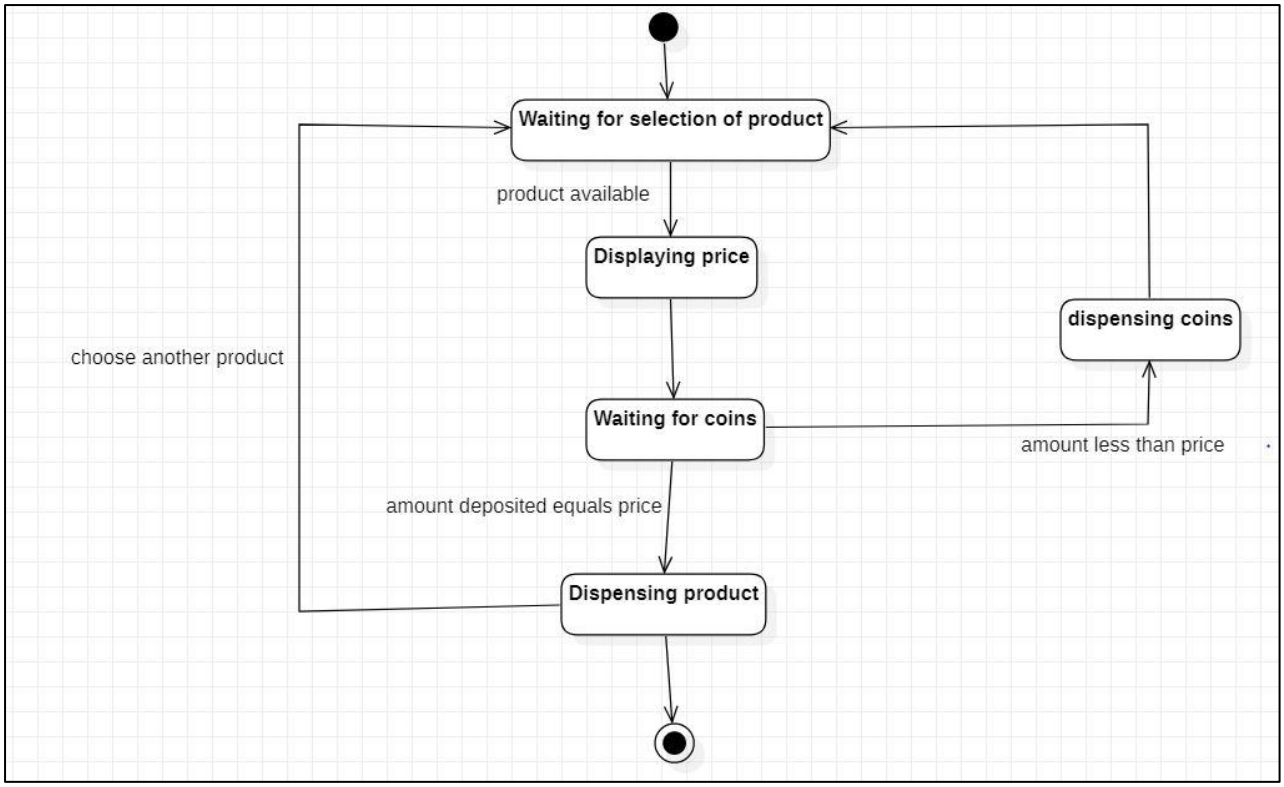


Fig19: Statechart Diagram made on Staruml software

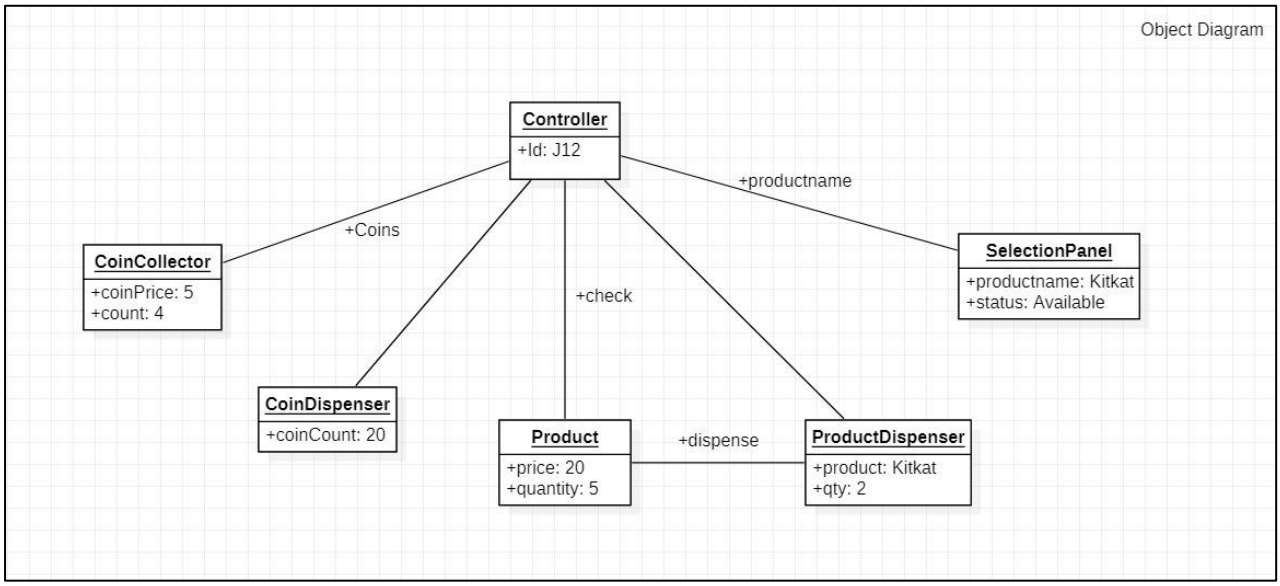


Fig20: Object Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X

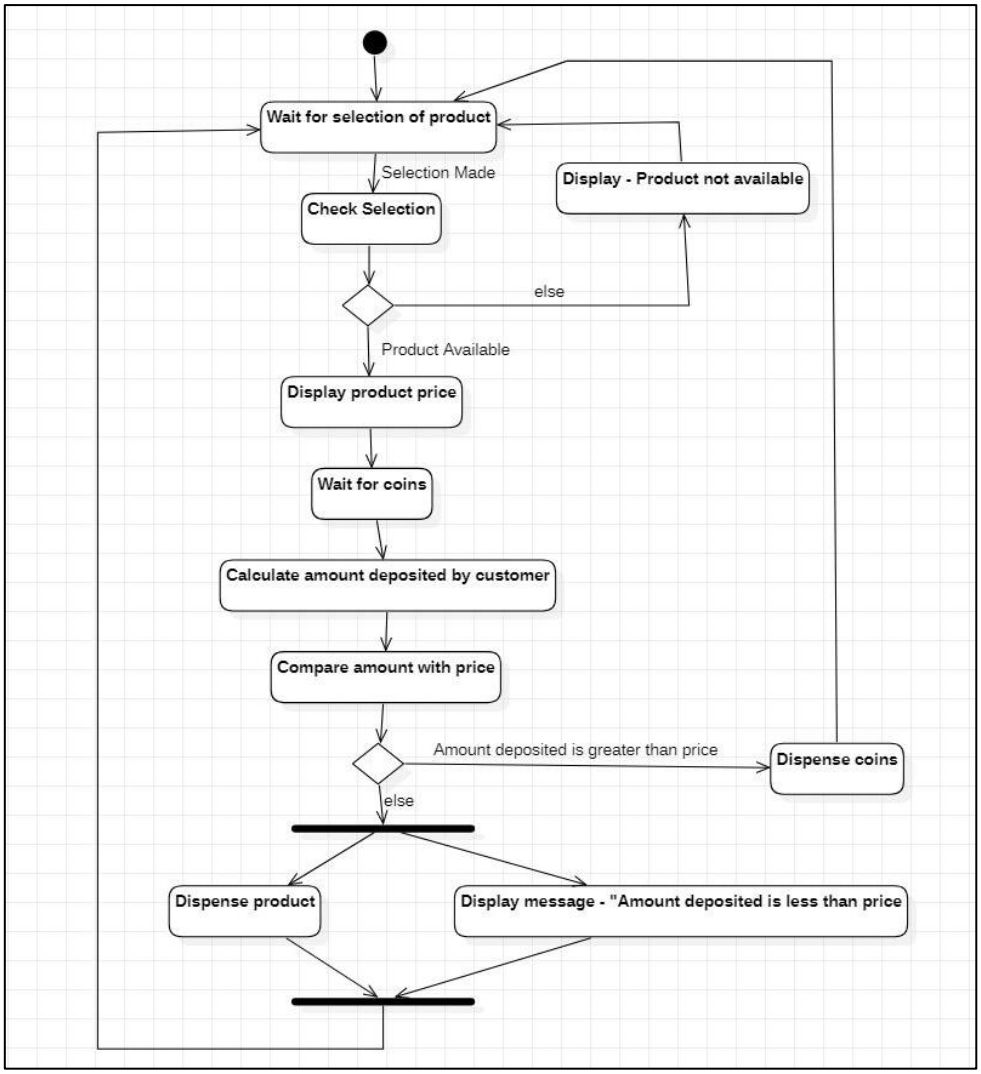


Fig21: Activity Diagram made on Staruml software

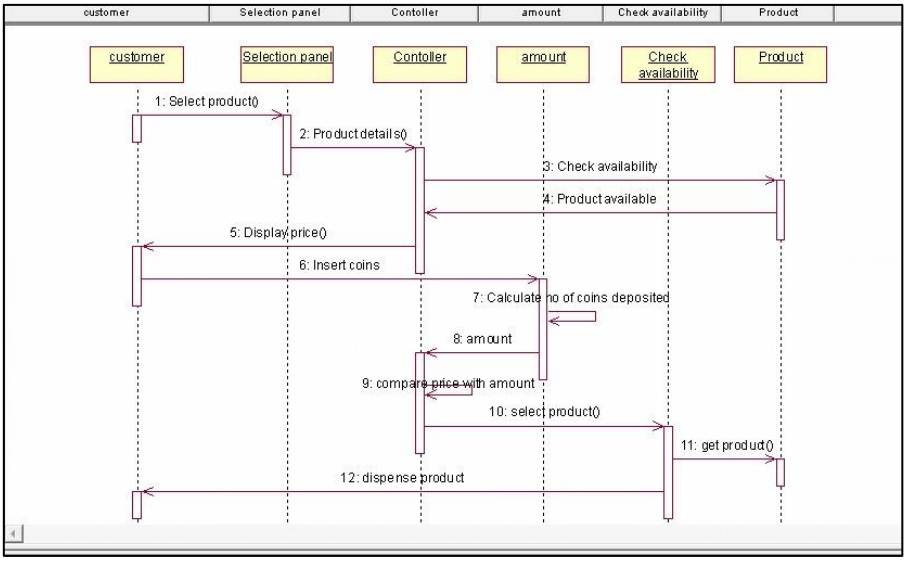


Fig22: Sequence Diagram made on Rational Rose software

ANCHAL KUMARI
A12405218083
6CSE-3X

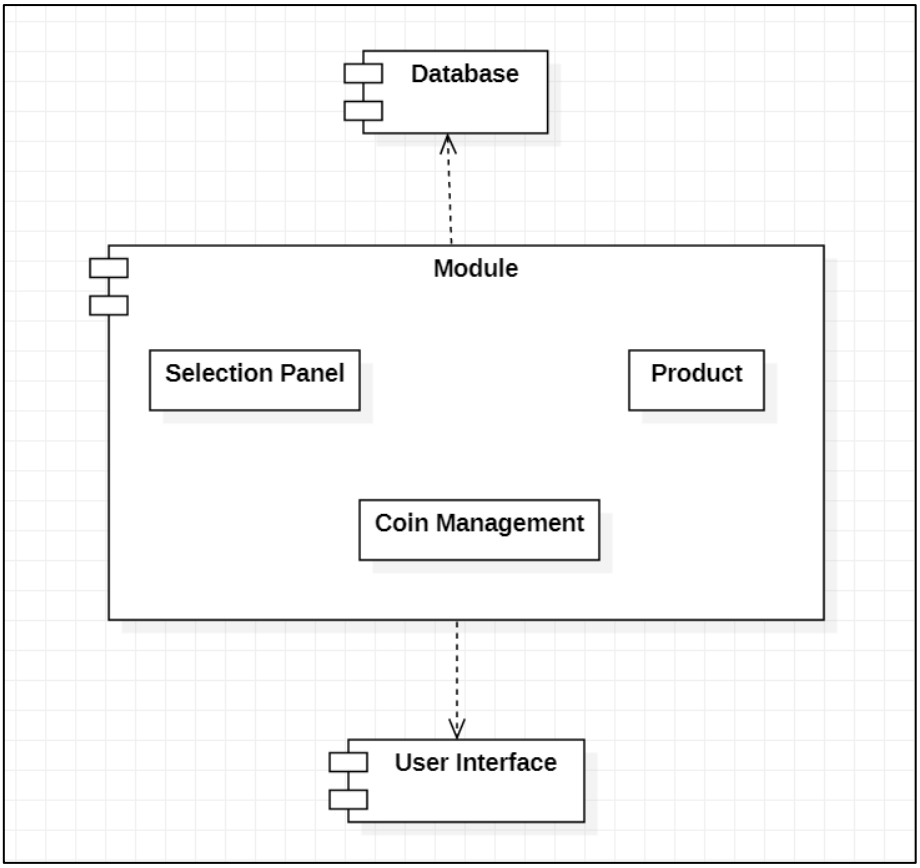


Fig23: Component Diagram made on Staruml software

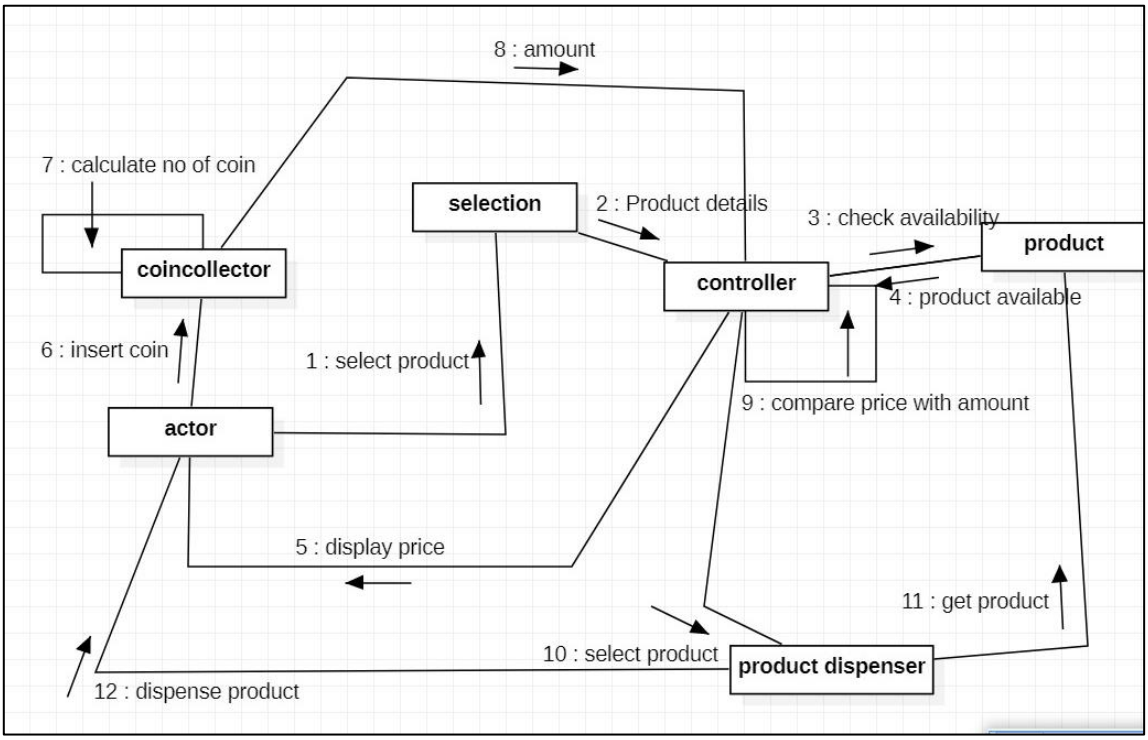


Fig24: Collaboration Diagram made on Staruml software

ANCHAL KUMARI
A12405218083
6CSE-3X