EXPERIMENT-1

Aim- Identify actors and use cases in Amizone.

Software Used- Draw.io

Theory:-

Actor

An Actor is outside or external to the system.

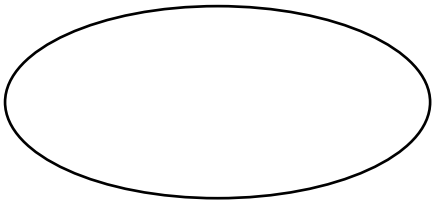
It can be a:

- Human, Peripheral device(hardware), External system or subsystem, Time or time-based event.
- Represented by stick figure.

Use-Case

- Each use case in a use case diagram describes one and only one function in which users interact with the system.
- Models a dialogue between an actor and the system.
- Represents the functionality provided by the system.

Represented by an oval.

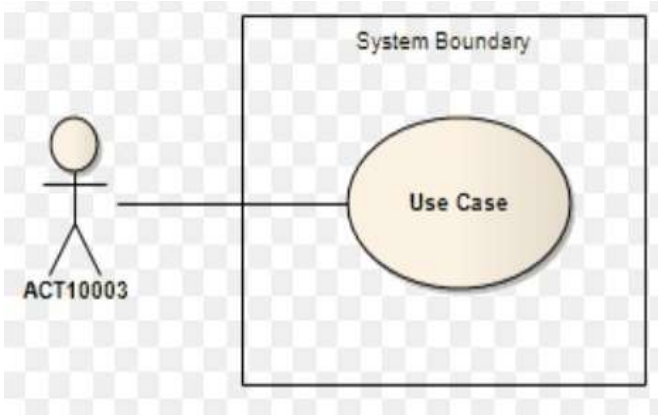


Relationship

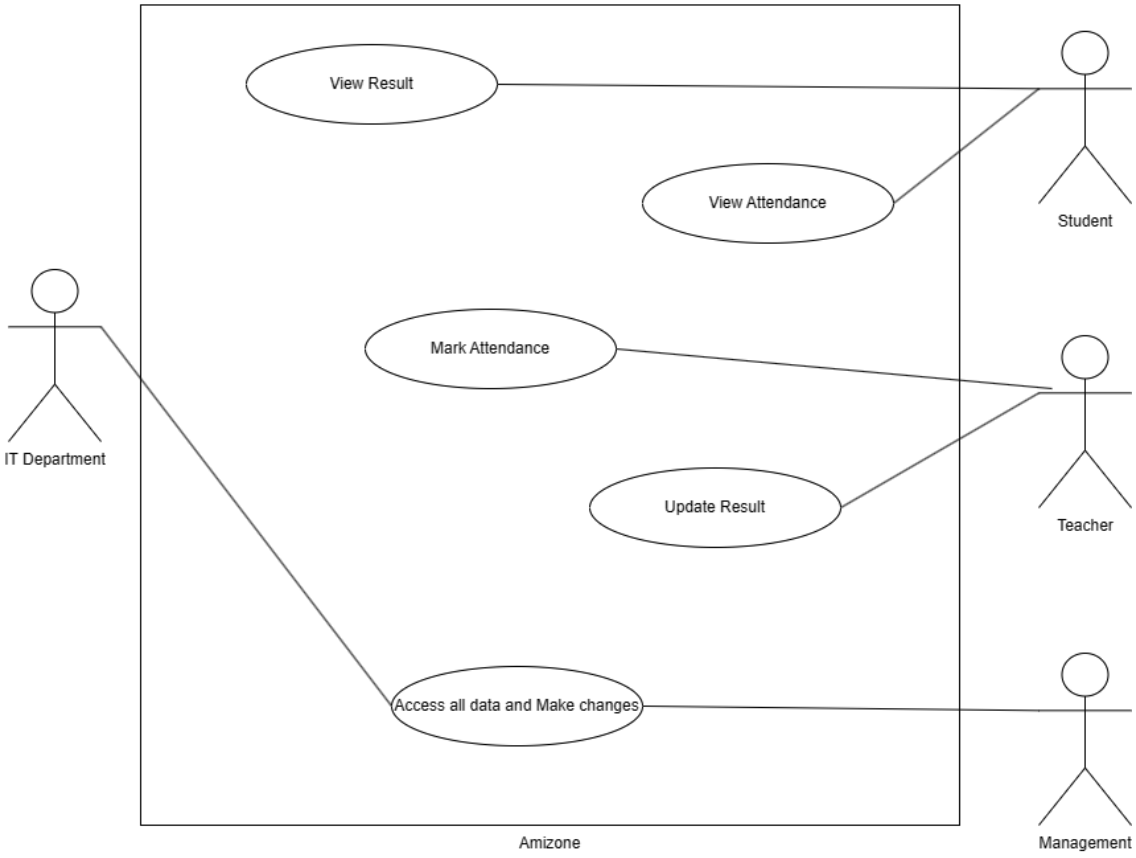
- Represented communication between actor and use case.
- Depicted by line or double-headed arrow line.
- Also called association relationship.

Boundary

- A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.



Use-Case Diagram



EXPERIMENT-1

Case Study 1.1

Aim- Use case diagram for CCS (Conference Chair System)

Software Used- Draw.io

Theory:-

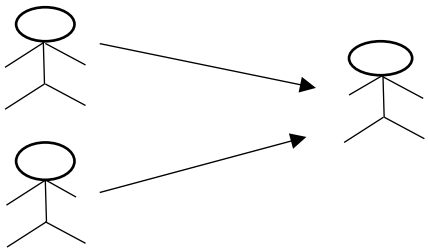
Association between actor and use case

- This is the foundation of any use case diagram.
- It represents a two-way communication between an actor, who interacts with the system, and a use case, which describes a specific functionality.
- Both actor and use case depend on each other - the actor initiates the use case, and the use case impacts the actor in some way.

Generalization of an actor

- This relationship shows inheritance between actors.
- A child actor inherits all the use cases of the parent actor and can have additional specific use cases of its own.
- This is useful for representing hierarchical roles within a system.

Represented by an arrow from child actor to parent actor.

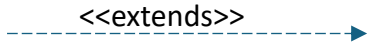


Extend between two use cases

- The "Extend" relationship allows one use case (the extending use case) to add optional functionality to another use case (the base use case).
- The extending use case only activates under specific conditions within the base use case.

Represented by proposed step.

<<extends>>



Include between two use cases

- The "Include" relationship refers to one use case (the including use case) incorporating the functionality of another use case (the included use case).

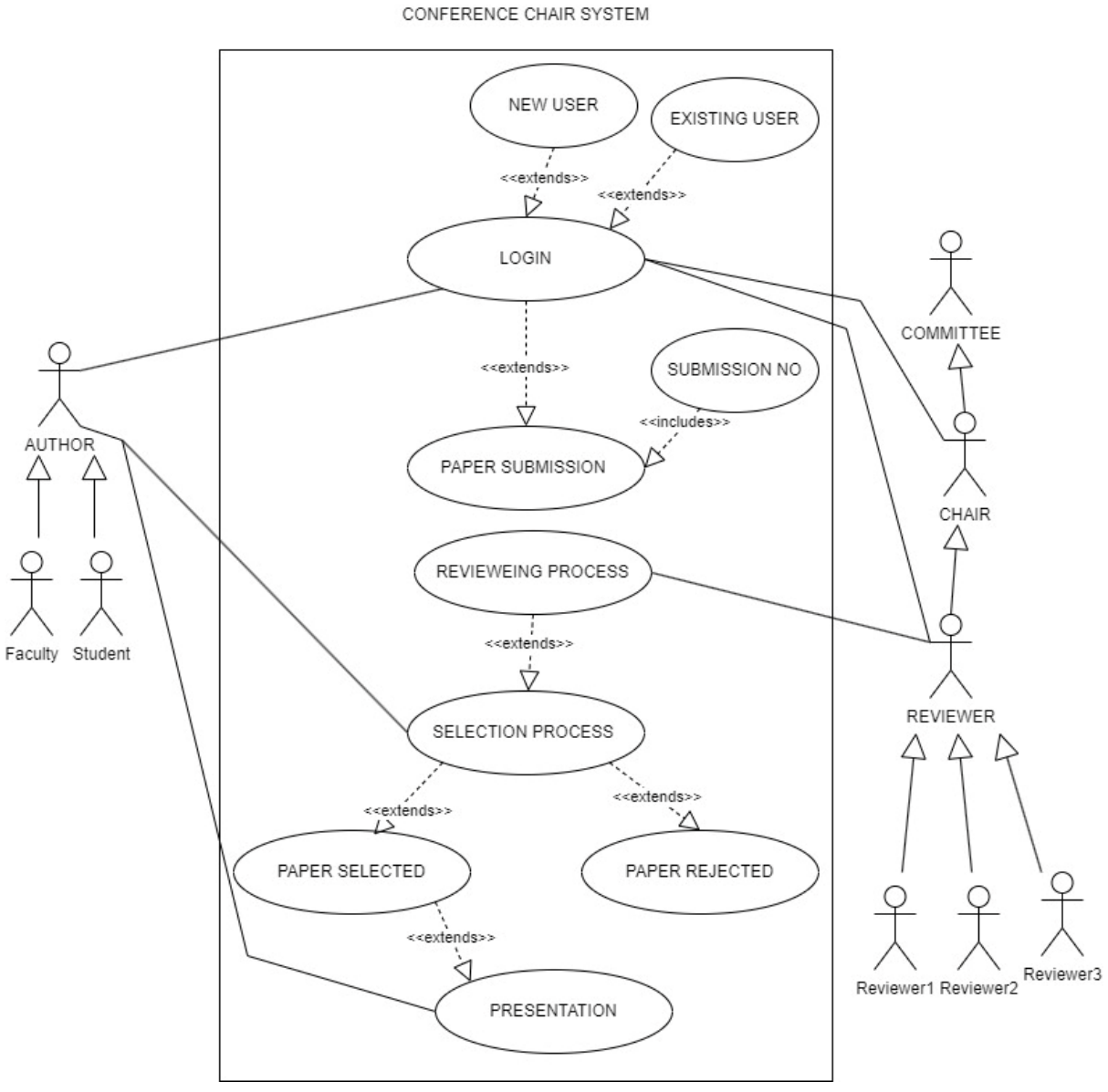
Represented by proposed step.

<<includes>>

Generalization of a use case

- This relationship shows inheritance between use cases.
- A child use case inherits the basic behaviour of the parent use case and adds its own specific steps or variations.
- This helps to avoid duplications and promotes reusability in common functionalities.

Use Case Diagram



EXPERIMENT-1

Case Study 1.2

Aim- Use case diagram for print on-demand service.

Software Used- Draw.io

Theory:-

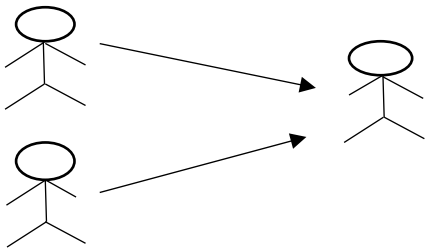
Association between actor and use case

- This is the foundation of any use case diagram.
- It represents a two-way communication between an actor, who interacts with the system, and a use case, which describes a specific functionality.
- Both actor and use case depend on each other - the actor initiates the use case, and the use case impacts the actor in some way.

Generalization of an actor

- This relationship shows inheritance between actors.
- A child actor inherits all the use cases of the parent actor and can have additional specific use cases of its own.
- This is useful for representing hierarchical roles within a system.

Represented by an arrow from child actor to parent actor.

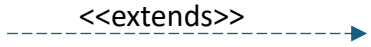


Extend between two use cases

- The "Extend" relationship allows one use case (the extending use case) to add optional functionality to another use case (the base use case).
- The extending use case only activates under specific conditions within the base use case.

Represented by proposed step.

<<extends>>



Include between two use cases

- The "Include" relationship refers to one use case (the including use case) incorporating the functionality of another use case (the included use case).

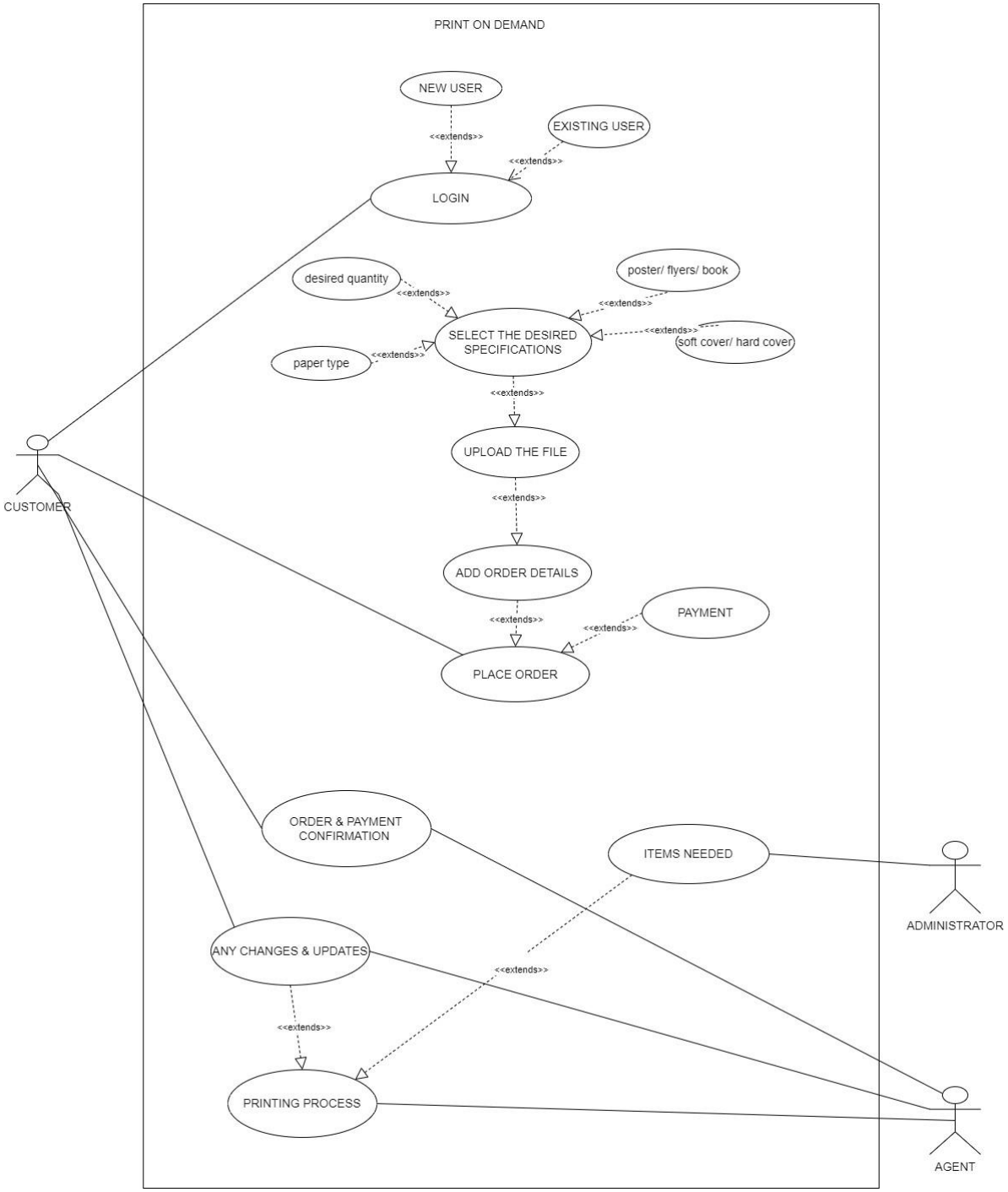
Represented by proposed step.

<<includes>>

Generalization of a use case

- This relationship shows inheritance between use cases.
- A child use case inherits the basic behaviour of the parent use case and adds its own specific steps or variations.
- This helps to avoid duplications and promotes reusability in common functionalities.

Use Case Diagram



EXPERIMENT-1

Case Study 1.3

Aim- Use case diagram for Restaurant table reservation system.

Software Used- Draw.io

Theory:-

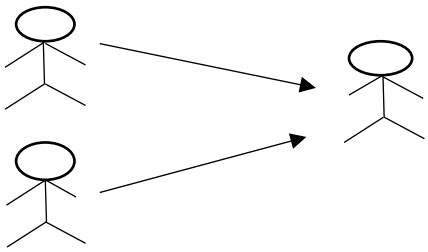
Association between actor and use case

- This is the foundation of any use case diagram.
- It represents a two-way communication between an actor, who interacts with the system, and a use case, which describes a specific functionality.
- Both actor and use case depend on each other - the actor initiates the use case, and the use case impacts the actor in some way.

Generalization of an actor

- This relationship shows inheritance between actors.
- A child actor inherits all the use cases of the parent actor and can have additional specific use cases of its own.
- This is useful for representing hierarchical roles within a system.

Represented by an arrow from child actor to parent actor.

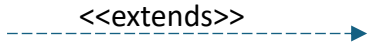


Extend between two use cases

- The "Extend" relationship allows one use case (the extending use case) to add optional functionality to another use case (the base use case).
- The extending use case only activates under specific conditions within the base use case.

Represented by proposed step.

<<extends>>



Include between two use cases

- The "Include" relationship refers to one use case (the including use case) incorporating the functionality of another use case (the included use case).

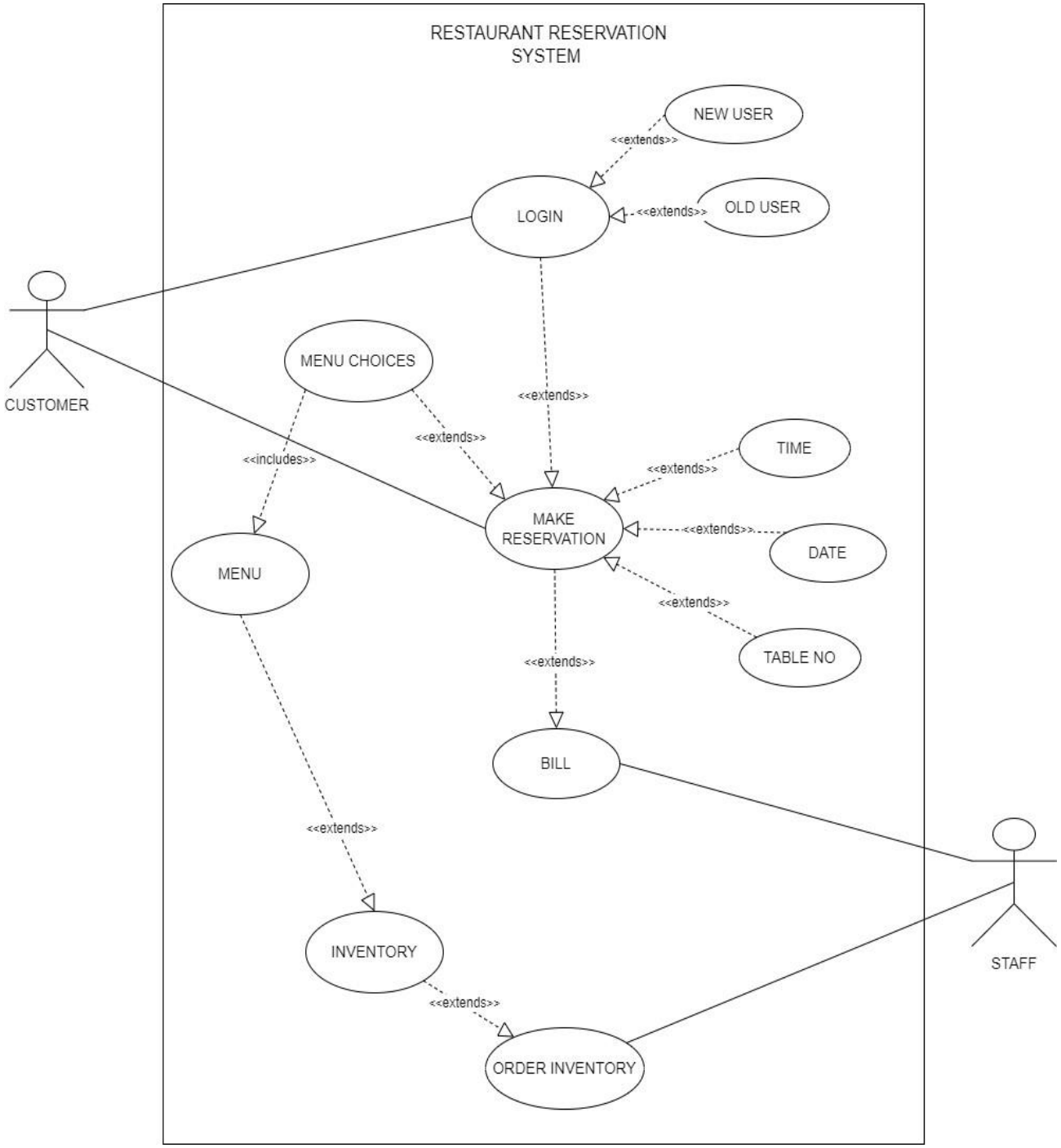
Represented by proposed step.

<<includes>>

Generalization of a use case

- This relationship shows inheritance between use cases.
- A child use case inherits the basic behaviour of the parent use case and adds its own specific steps or variations.
- This helps to avoid duplications and promotes reusability in common functionalities.

Use Case Diagram:



EXPERIMENT-2

Aim- To draw a class diagram of Amizone.

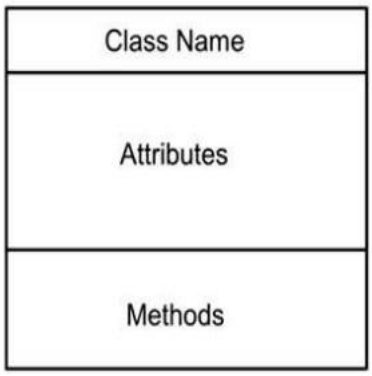
Software Used- Draw.io

Theory:-

Class Diagram: Class diagram is one of the important UML diagrams for software development which shows

- Object classes of the software
- Association b/w the classes.

Class diagram is static structure diagram that describes the structure of by showing the system classes, their attributes and operations, and relationship among them.
Represented by-



Attributes

It represents the information, data, or properties that belong to instances of a classifier.
Visibility of attributes-

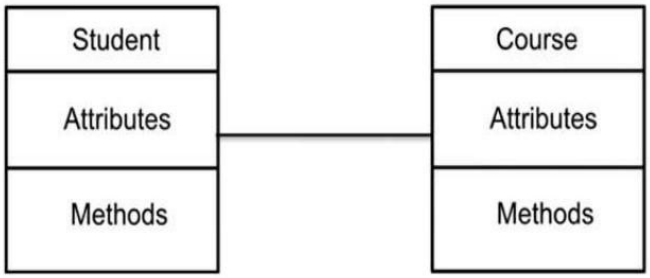
Private	-
Public	+
Protected	#
Package	~
Derived	/
Static	underline

Methods

It represents the operations that can be done within the class.

Association

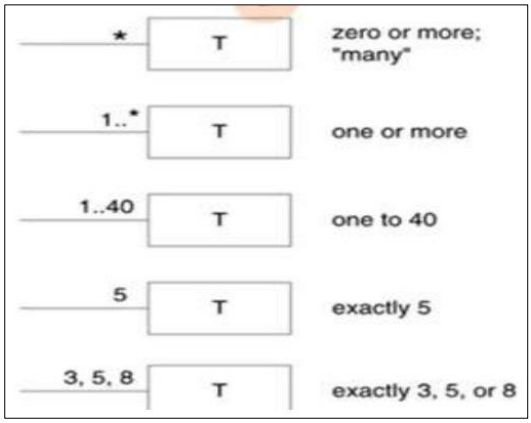
An association represents a relationship between two classes. It indicates that objects of one class have a relationship with objects of another class.



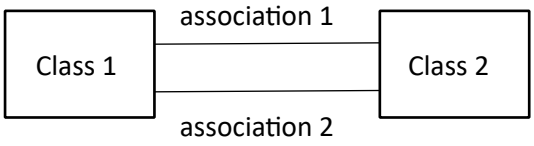
Whenever there is an association between two classes of a class diagram, there must be ***Multiplicity***

There are types of Multiplicity.

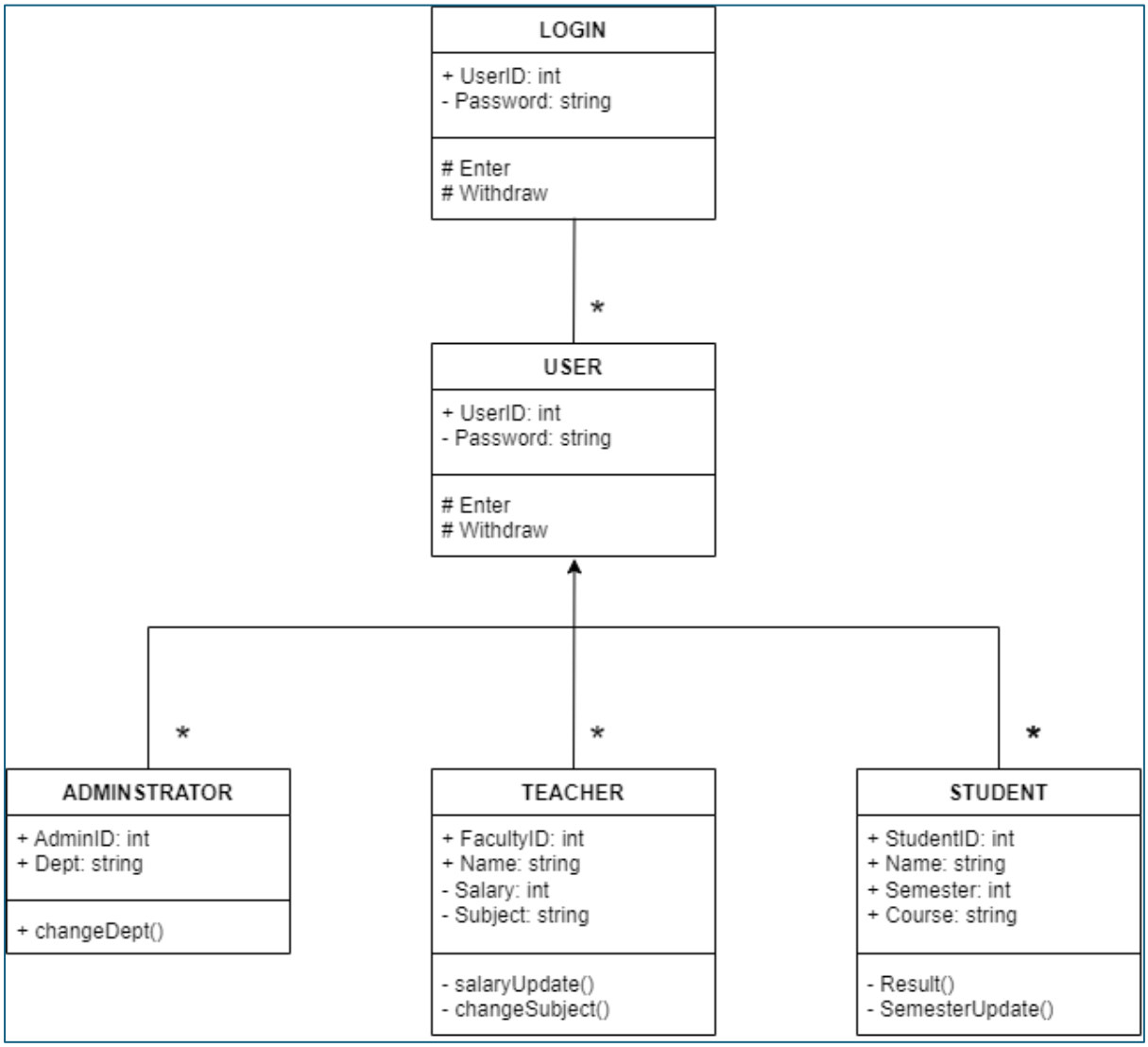
Exactly One	1
Zero or More	0..*
One or More	1..*
Specified Rage	2..4
Multiple Disjoint	2,4..6,8
One or Other	2,4



There can be multiple associations between two classes.



Class Diagram:



EXPERIMENT-2

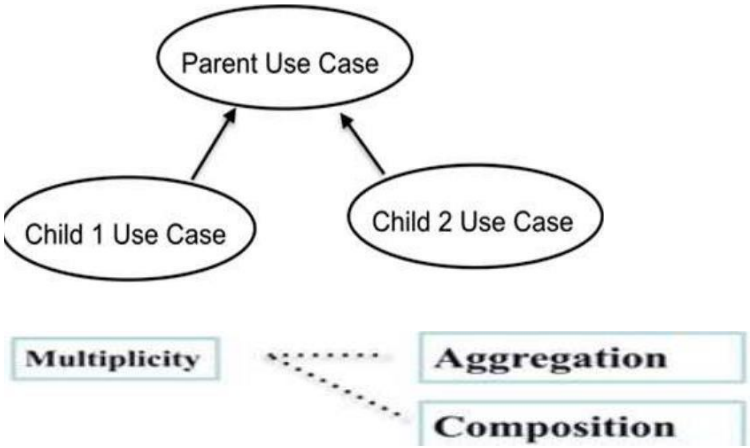
Case Study 2.1

Aim- To draw a class diagram for exam designer model.

Software Used- Draw.io

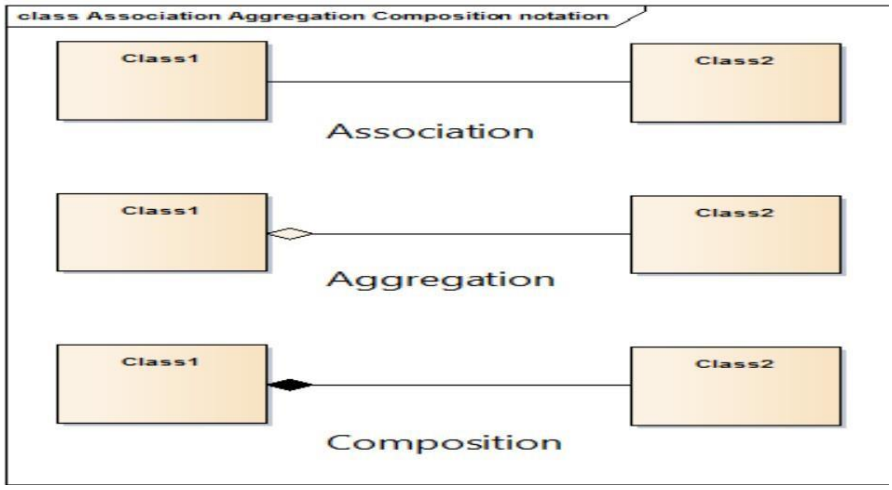
Theory:-

Inheritance: It means showing parent-child relationship.

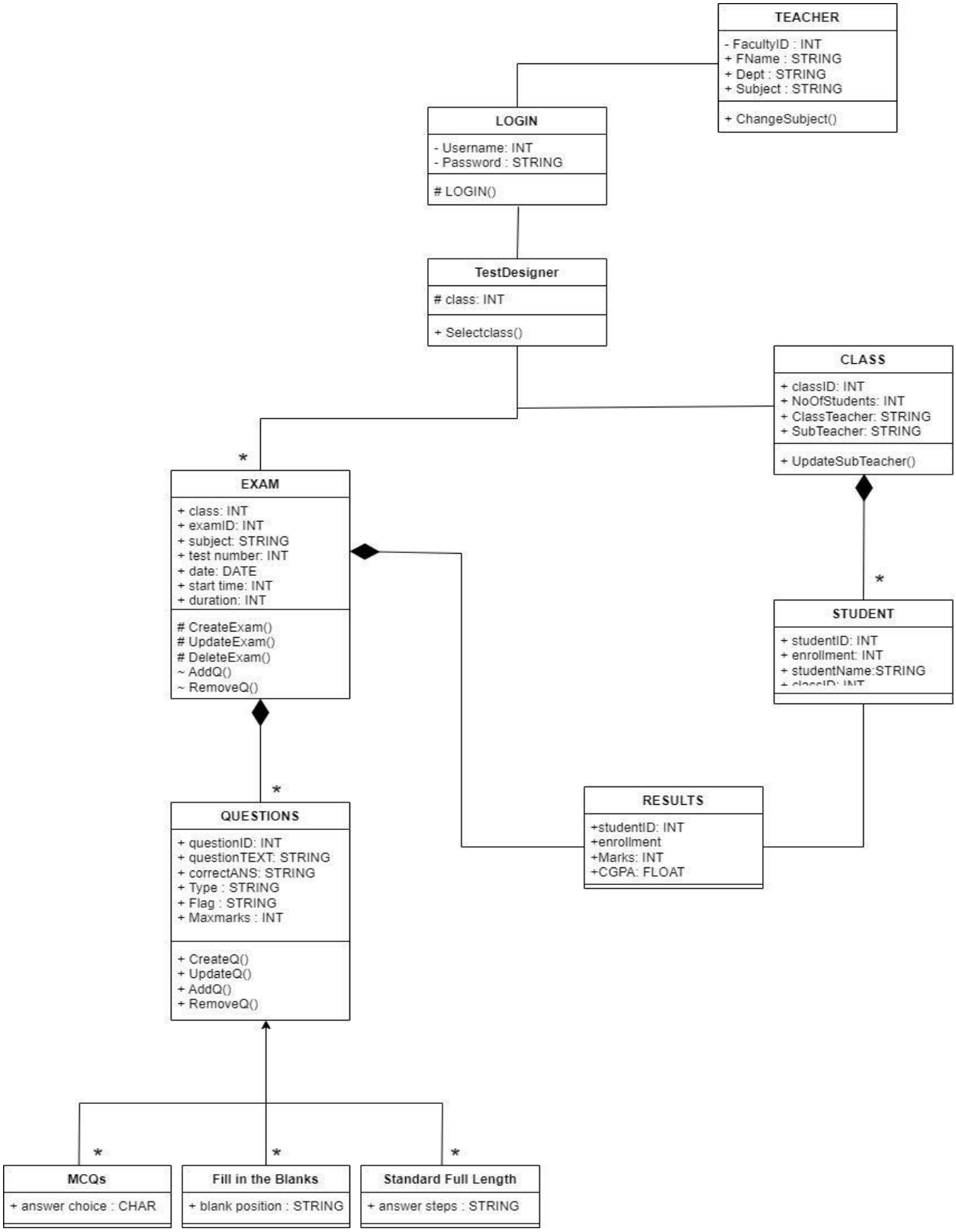


Aggregation: implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition: Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child).



Class Diagram:



EXPERIMENT-2

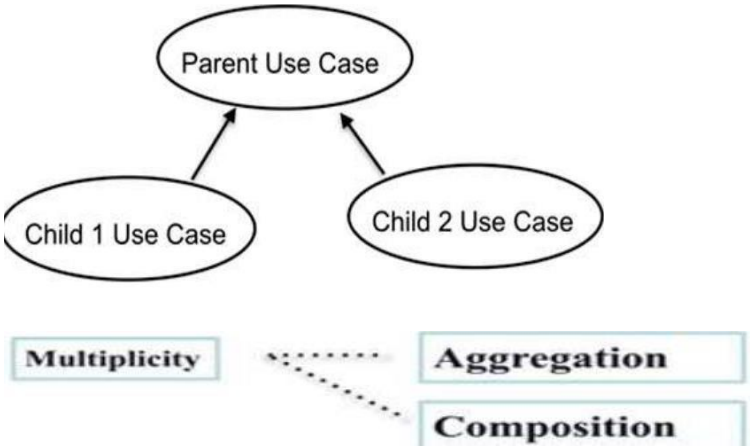
Case Study 2.2

Aim- To draw a class diagram for Student Information System.

Software Used- Draw.io

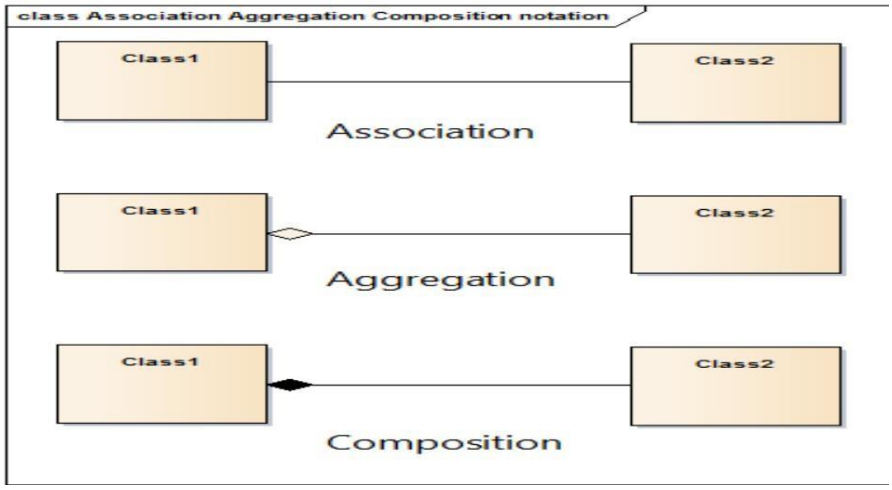
Theory:-

Inheritance: It means showing parent-child relationship.

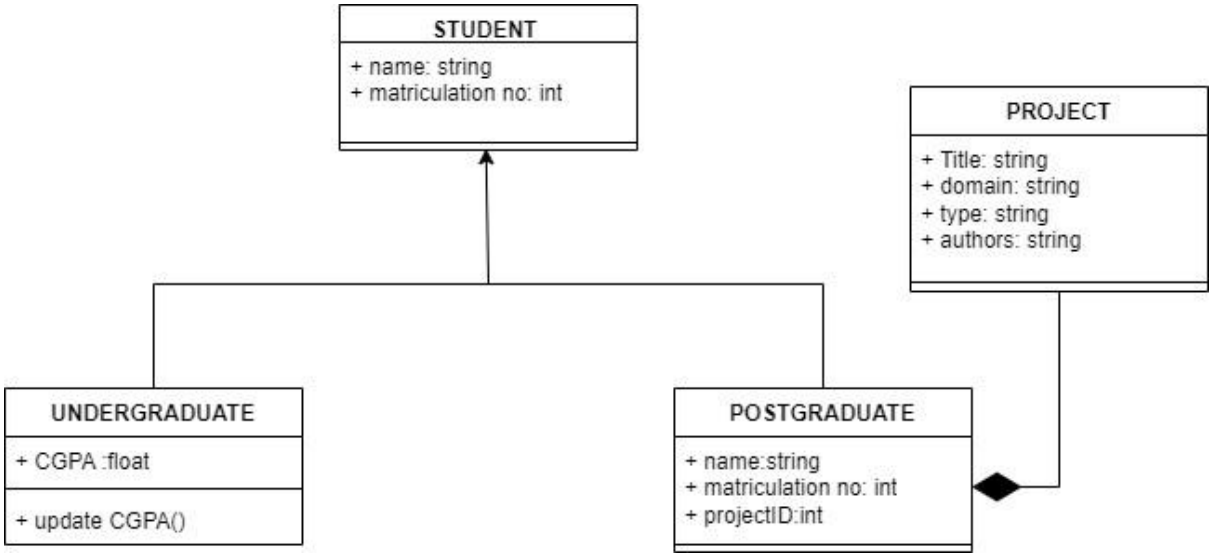


Aggregation: implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition: Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child).



Class Diagram:



EXPERIMENT-2

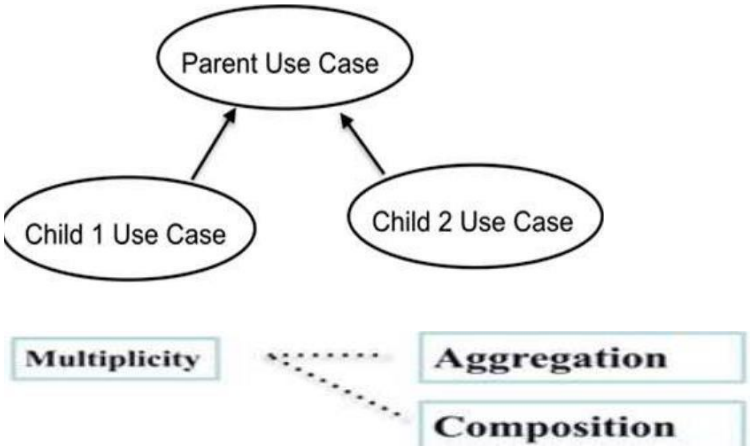
Case Study 2.3

Aim- To draw a class diagram for Quiz Application.

Software Used- Draw.io

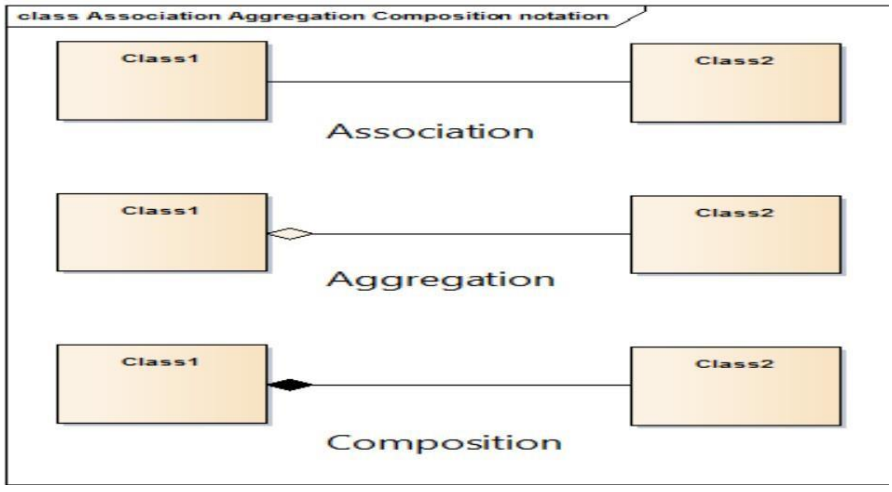
Theory:-

Inheritance: It means showing parent-child relationship.

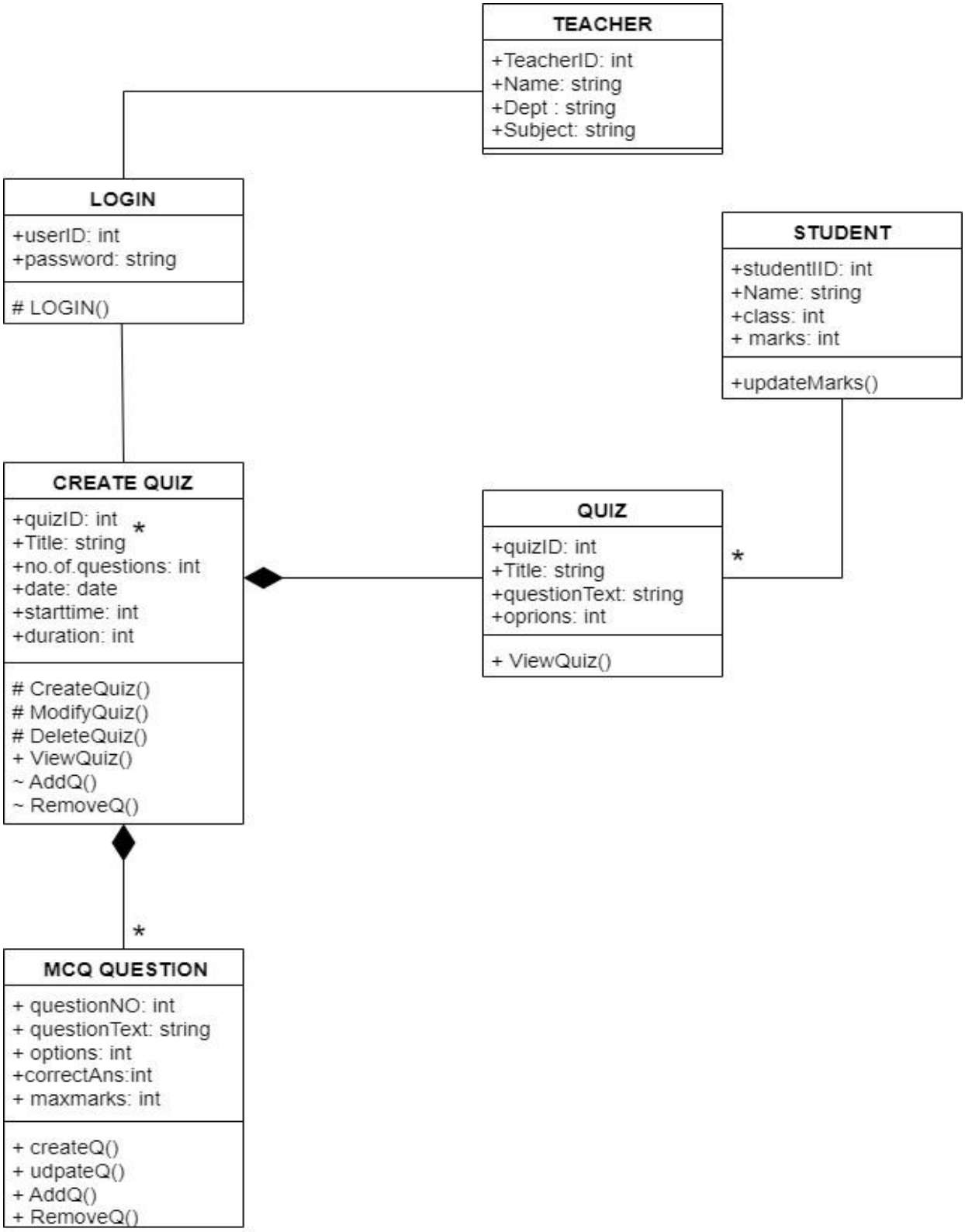


Aggregation: implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition: Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child).



Class Diagram:



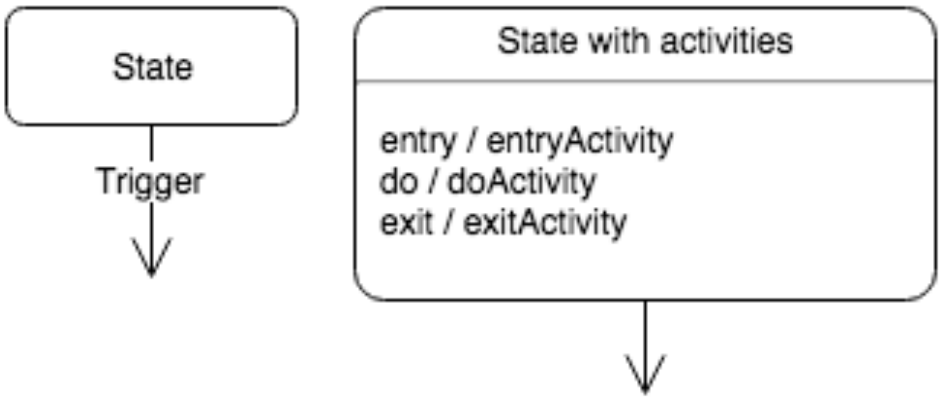
EXPERIMENT-3

Aim- To draw the state diagram of Amizone.

Software Used- Draw.io

Theory:-

State Diagram: A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system.
State diagrams emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems.
State Machine: A state machine is a behaviour that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
State*:* A state is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.



Transition: A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.

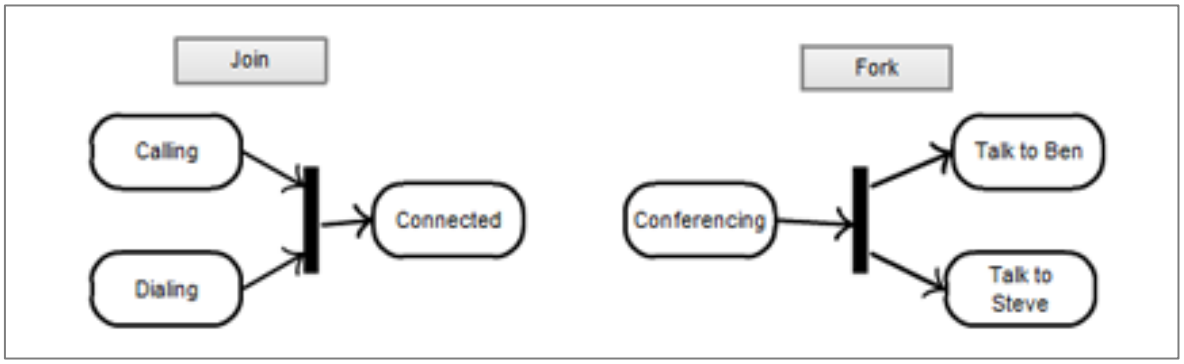
Initial State: A filled circle followed by an arrow represents the object's initial state.

Final State: An arrow pointing to a filled circle nested inside another circle represents the object's final state.

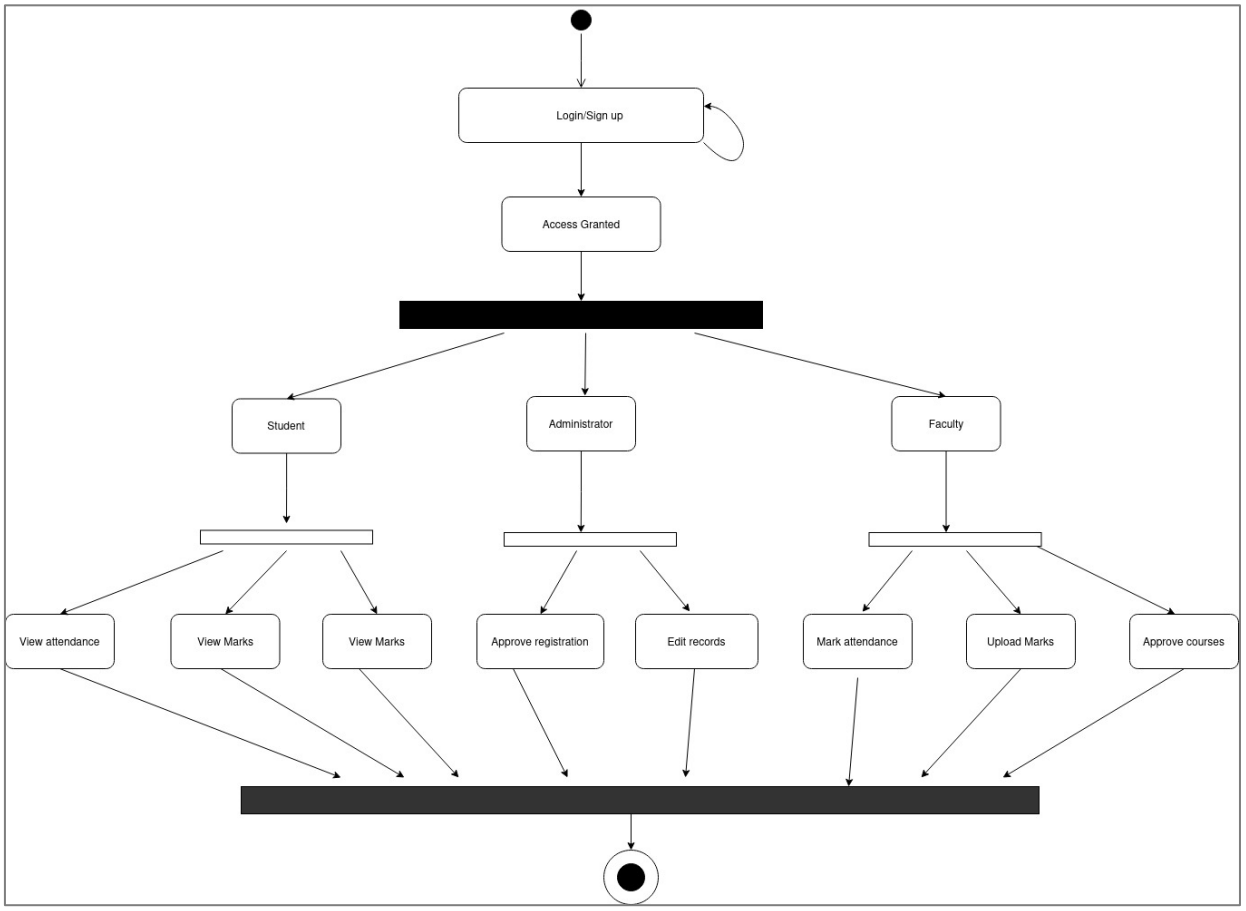
Final State

Fork: It is a control node that splits a flow into multiple concurrent flows.

Join: It is a control node that synchronizes multiple flows.



State Diagram:



EXPERIMENT-3

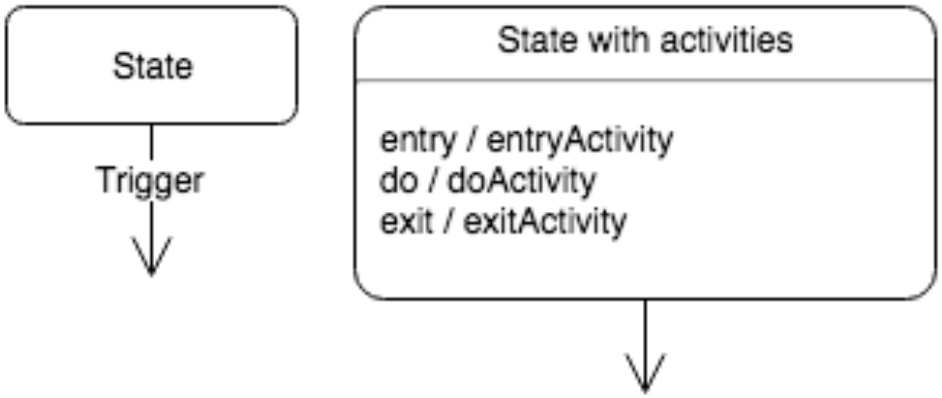
Case Study 3.1

Aim- To draw the state diagram for Smart Home System.

Software Used- Draw.io

Theory:-

State Diagram: A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system.
State diagrams emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems.
State Machine: A state machine is a behaviour that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
State*:* A state is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.



Transition: A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.

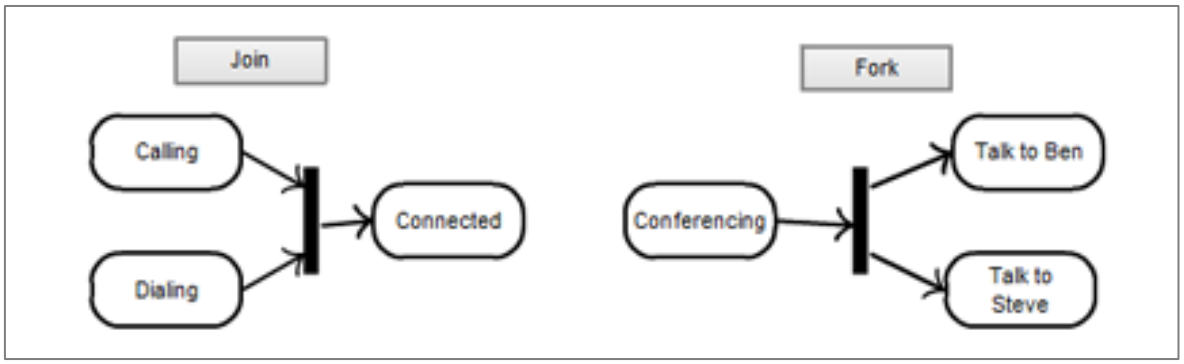
Initial State: A filled circle followed by an arrow represents the object's initial state.

Final State: An arrow pointing to a filled circle nested inside another circle represents the object's final state.

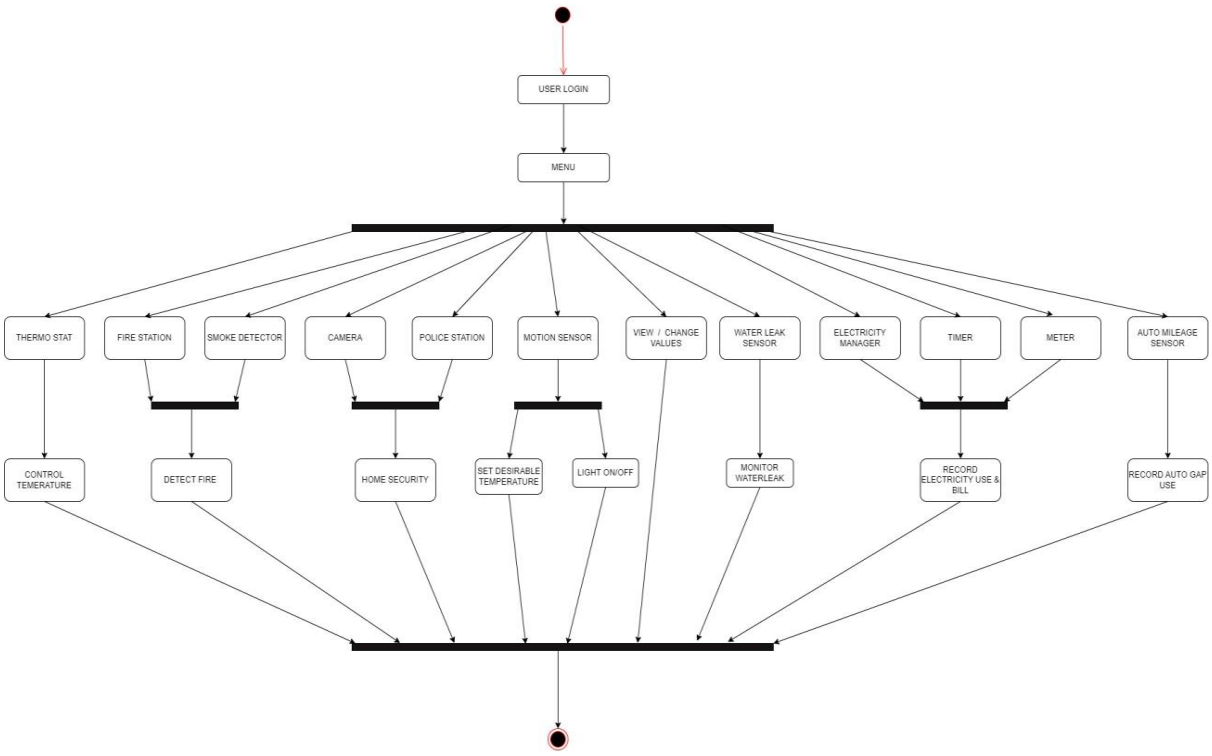
Final State

Fork: It is a control node that splits a flow into multiple concurrent flows.

Join: It is a control node that synchronizes multiple flows.



State Diagram:



EXPERIMENT-3

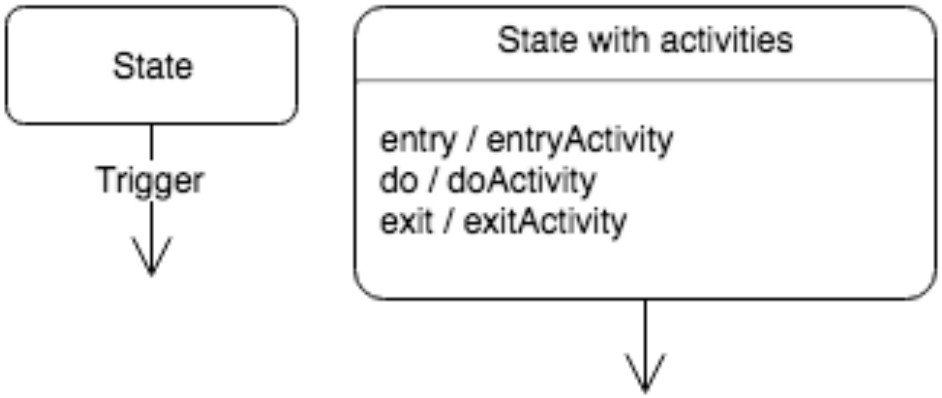
Case Study 3.2

Aim- To draw the state diagram for Reservation System.

Software Used- Draw.io

Theory:-

State Diagram: A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system.
State diagrams emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems.
State Machine: A state machine is a behaviour that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
State*:* A state is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.



Transition: A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.

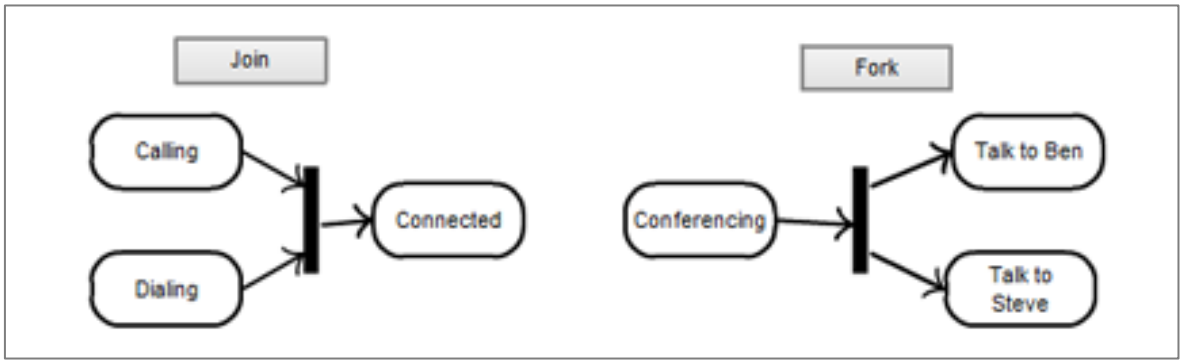
Initial State: A filled circle followed by an arrow represents the object's initial state.

Final State: An arrow pointing to a filled circle nested inside another circle represents the object's final state.

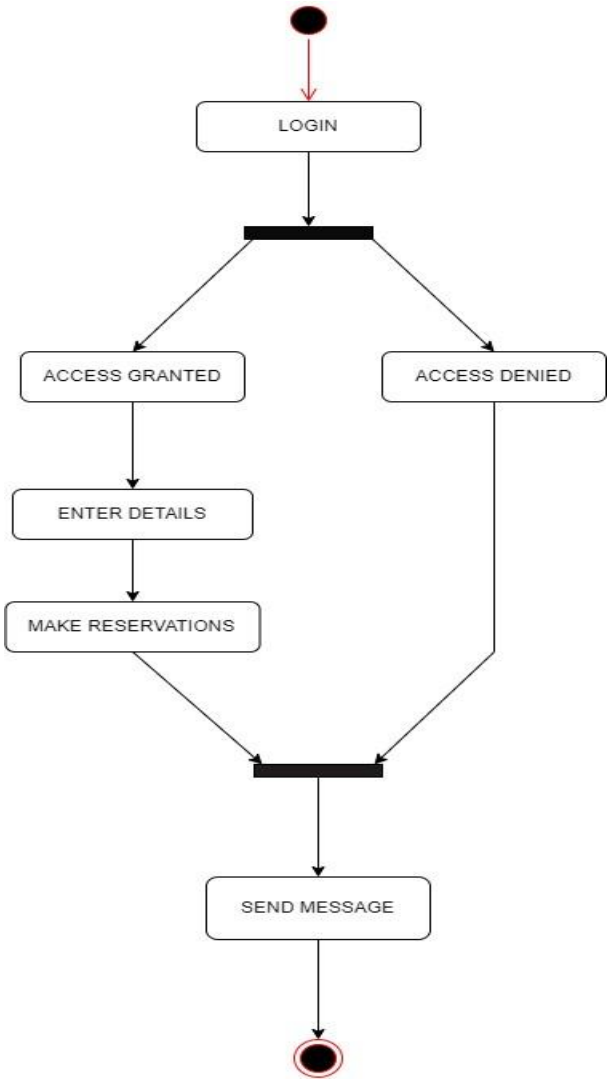
Final State

Fork: It is a control node that splits a flow into multiple concurrent flows.

Join: It is a control node that synchronizes multiple flows.



State Diagram:



EXPERIMENT-3

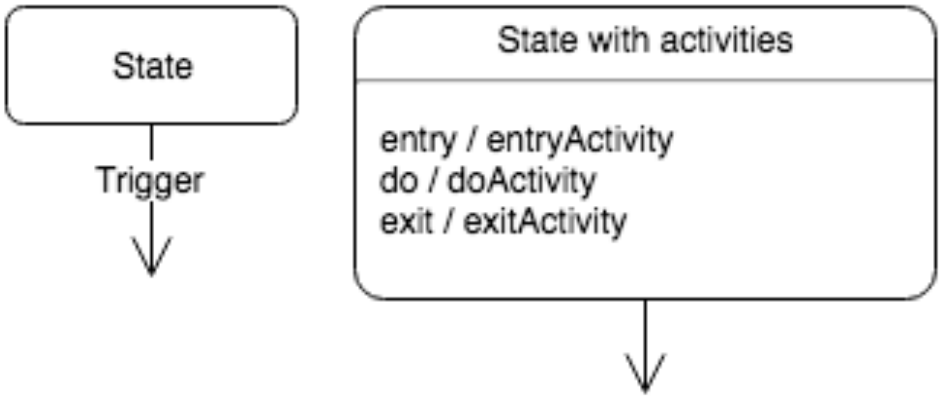
Case Study 3.3

Aim- To draw the state diagram for Course Registration System.

Software Used- Draw.io

Theory:-

State Diagram: A state diagram consists of states, transitions, events, and activities. You use state diagrams to illustrate the dynamic view of a system.
State diagrams emphasize the event-ordered behaviour of an object, which is especially useful in modelling reactive systems.
State Machine: A state machine is a behaviour that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
State*:* A state is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.



Transition: A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.

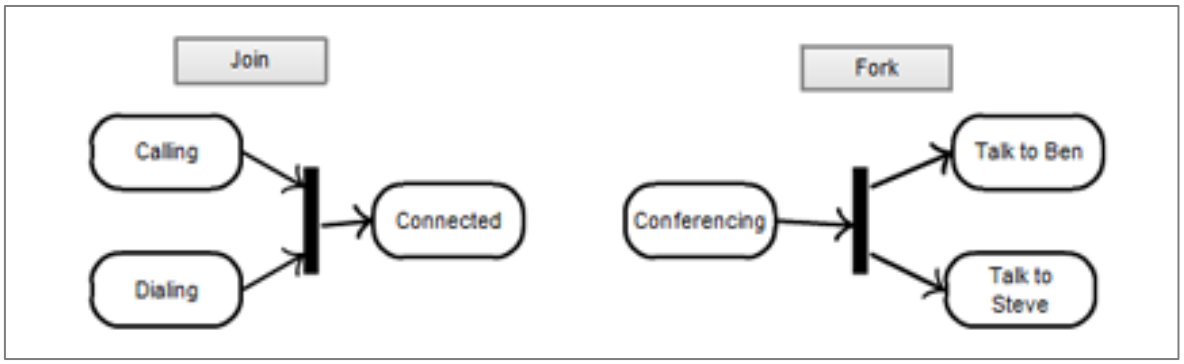
Initial State: A filled circle followed by an arrow represents the object's initial state.

Final State: An arrow pointing to a filled circle nested inside another circle represents the object's final state.

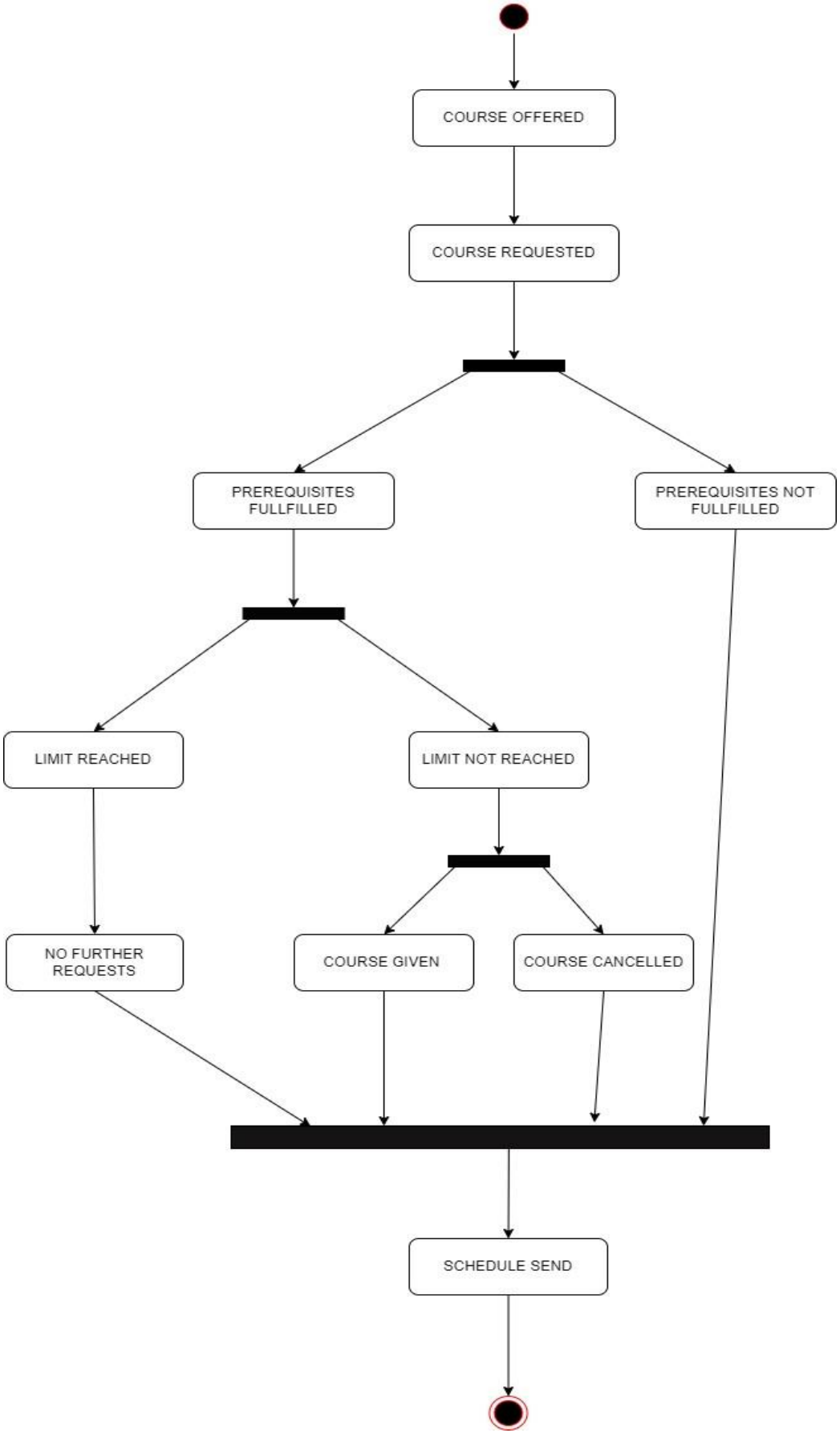
Final State

Fork: It is a control node that splits a flow into multiple concurrent flows.

Join: It is a control node that synchronizes multiple flows.



State Diagram:



EXPERIMENT-4

Aim- To make an activity diagram for Amizone.

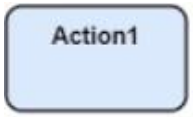
Software Used- Draw.io

Theory:-

Activity Diagram: Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.
It is also suitable for modelling how a collection of use cases coordinates to represent business workflows.

Activity:
Is used to represent a set of actions.

Action:
A task to be performed.



Control Flow:
Shows the sequence of execution.

Initial Node:
Portrays the beginning of a set of actions or activities.

Final Node:
Stop all control flows and object flows in an activity (or action).

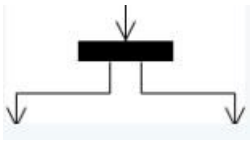
Decision Mode:
Represent a test condition to ensure that the control flow or object flow only goes down one path.

Merge Node:

Bring back together different decision paths that were created using a decision node.

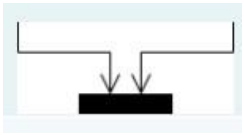
Fork Node:

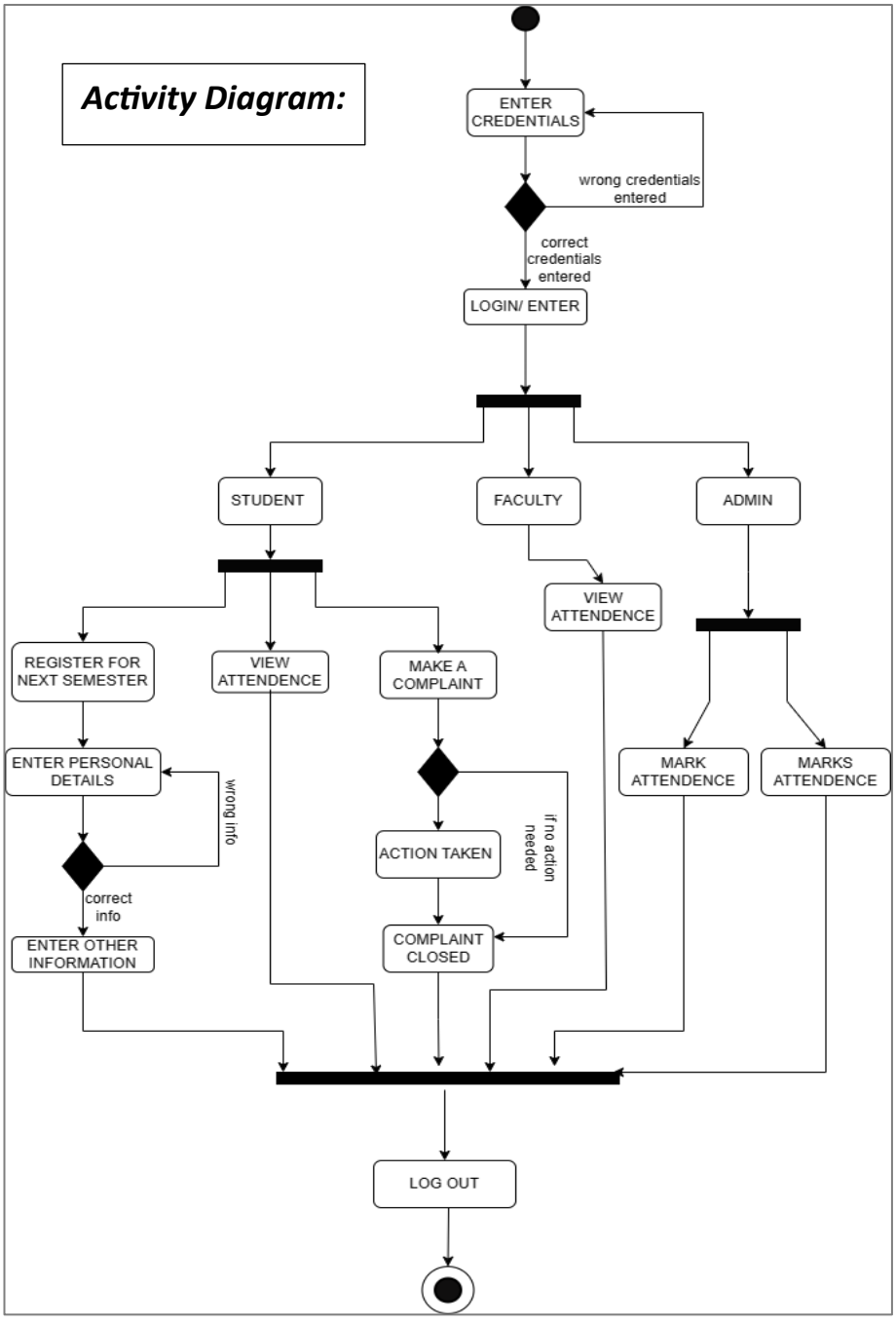
Split behaviour into a set of parallel or concurrent flows of activities (or actions).



Join Node:

Bring back together a set of parallel or concurrent flows of activities (or actions).





EXPERIMENT-4

Case Study 4.1

Aim- To make an activity diagram for Reservation System.

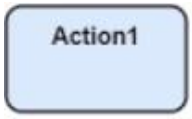
Software Used- Draw.io

Theory:-

Activity Diagram: Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction.
It is also suitable for modelling how a collection of use cases coordinates to represent business workflows.

Activity:
Is used to represent a set of actions.

Action:
A task to be performed.



Control Flow:
Shows the sequence of execution.

Initial Node:
Portrays the beginning of a set of actions or activities.

Final Node:
Stop all control flows and object flows in an activity (or action).

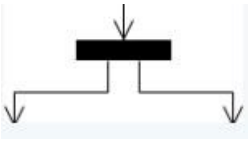
Decision Mode:
Represent a test condition to ensure that the control flow or object flow only goes down one path.

Merge Node:

Bring back together different decision paths that were created using a decision node.

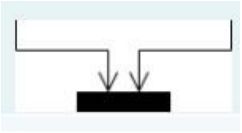
Fork Node:

Split behaviour into a set of parallel or concurrent flows of activities (or actions).

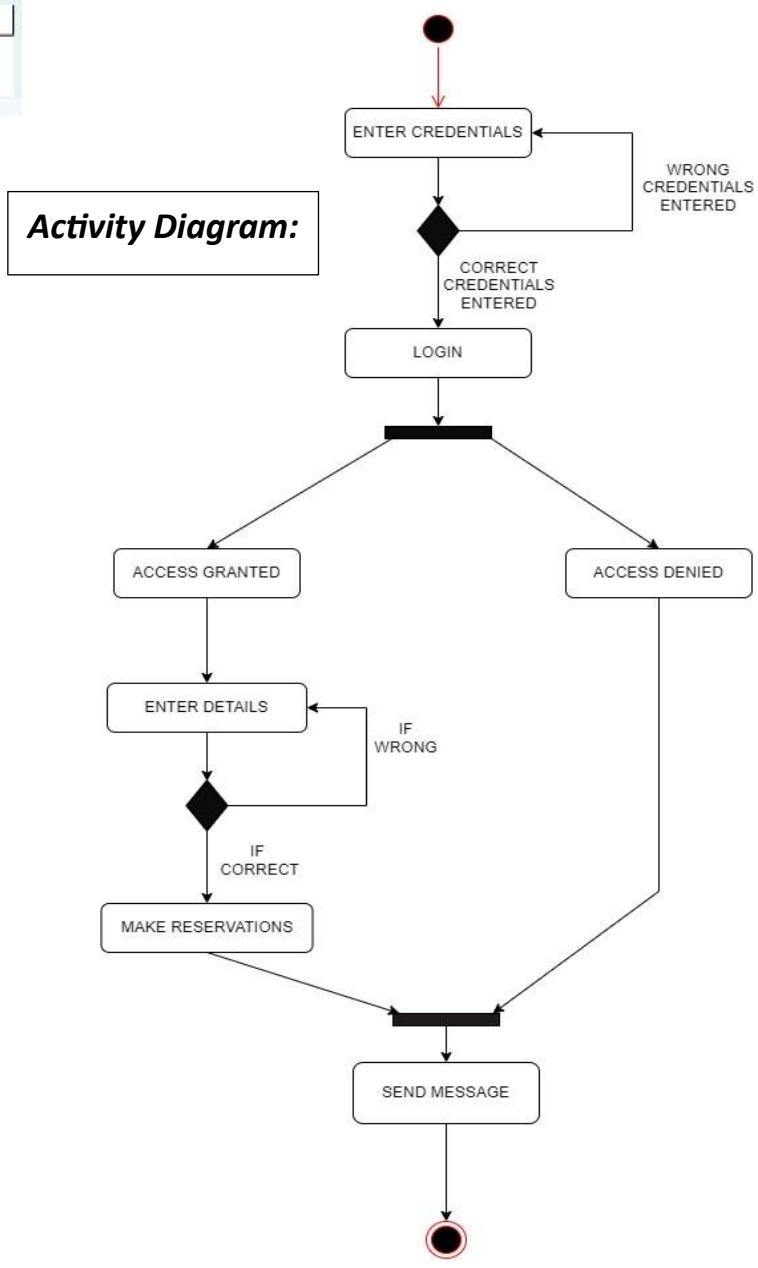


Join Node:

Bring back together a set of parallel or concurrent flows of activities (or actions).



Activity Diagram:



EXPERIMENT-5

Aim- To create object diagram for Amizone.

Software Used- Draw.io

Theory:-

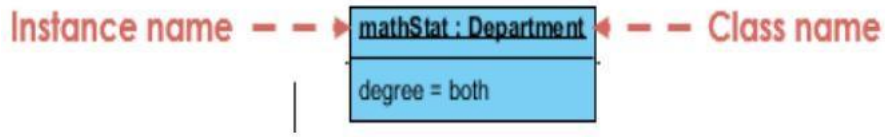
A UML object diagram represents a specific instance of a class diagram at a certain moment in time.

An object diagram focuses on the attributes of a set of objects and how those objects relate to each other.

Object diagram elements

Objects

Objects are instances of a class.

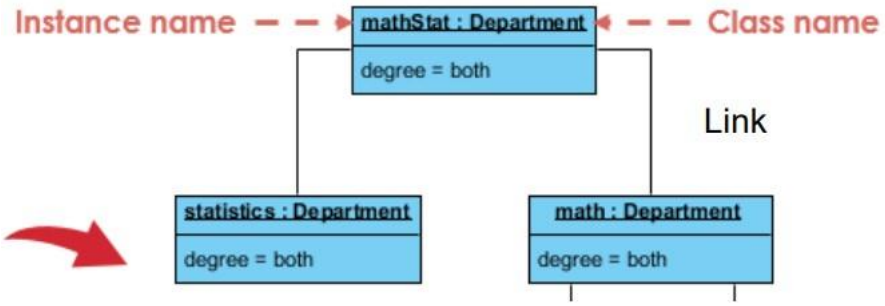


Class titles

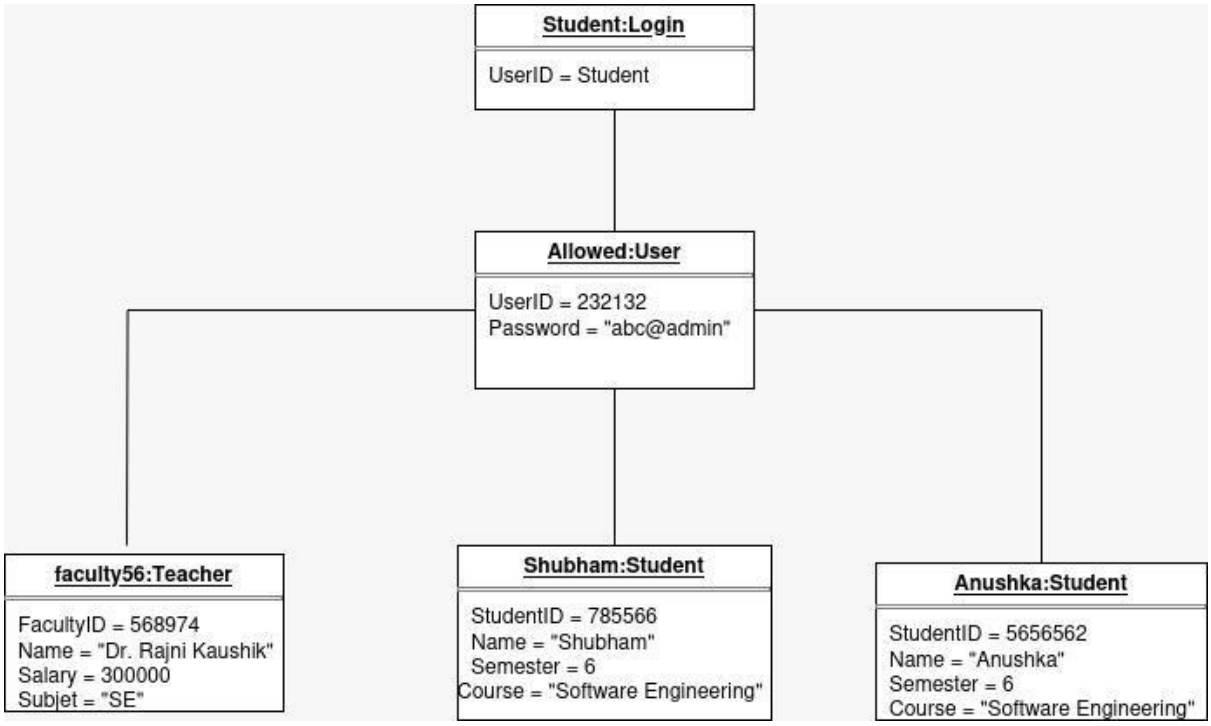
Class titles are the specific attributes of a given class.

Links

Links are the lines that connect two shapes of an object diagram to each other.



Object Diagram



EXPERIMENT-5

Case Study 5.1

Aim- To create object diagram for Healthcare Institution.

Software Used- Draw.io

Theory:-

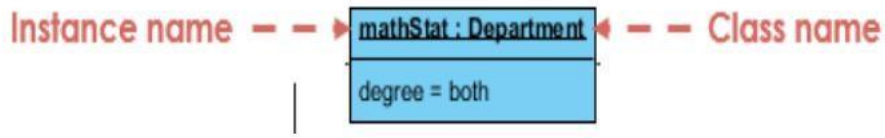
A UML object diagram represents a specific instance of a class diagram at a certain moment in time.

An object diagram focuses on the attributes of a set of objects and how those objects relate to each other.

Object diagram elements

Objects

Objects are instances of a class.

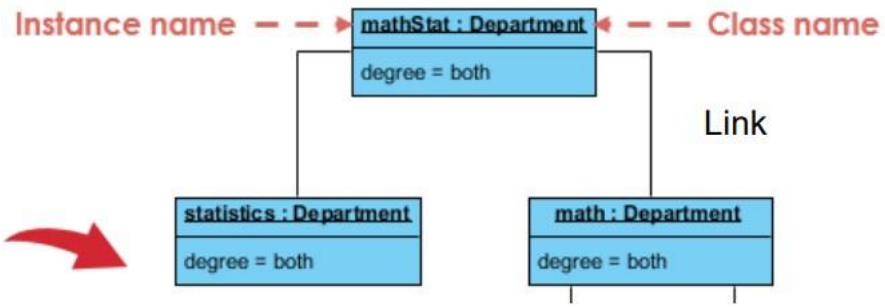


Class titles

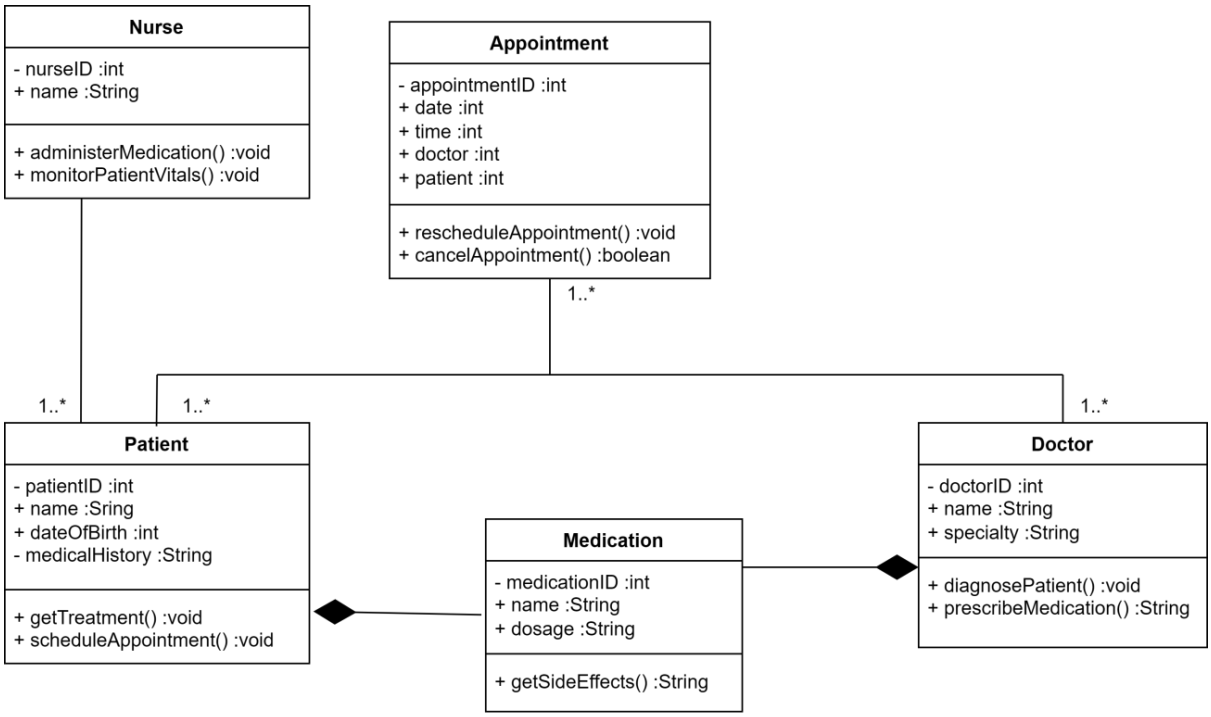
Class titles are the specific attributes of a given class.

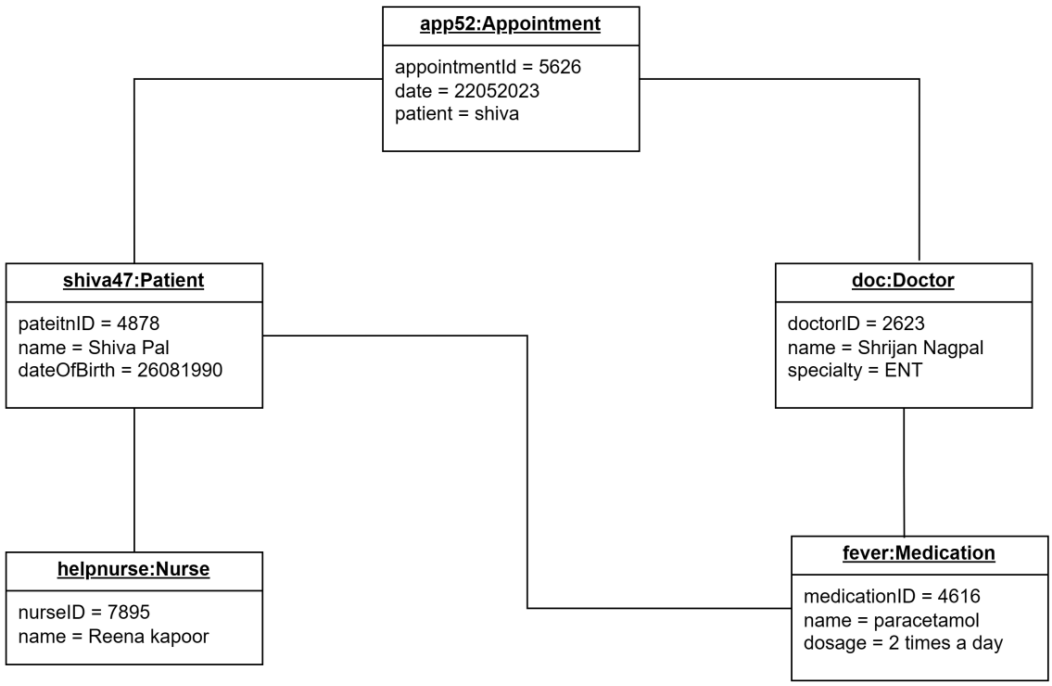
Links

Links are the lines that connect two shapes of an object diagram to each other.



Object Diagram





EXPERIMENT-5

Case Study 5.2

Aim- To create object diagram for Vehicle Locator.

Software Used- Draw.io

Theory:-

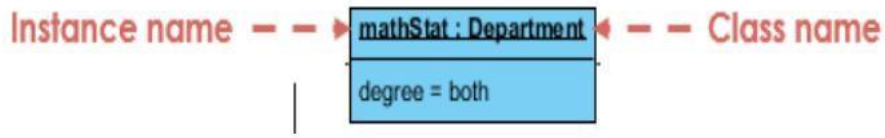
A UML object diagram represents a specific instance of a class diagram at a certain moment in time.

An object diagram focuses on the attributes of a set of objects and how those objects relate to each other.

Object diagram elements

Objects

Objects are instances of a class.

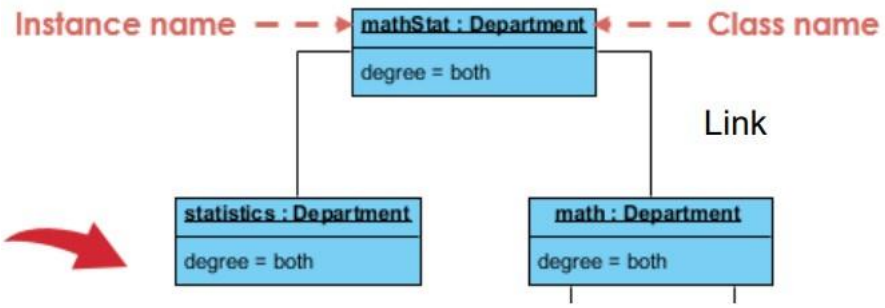


Class titles

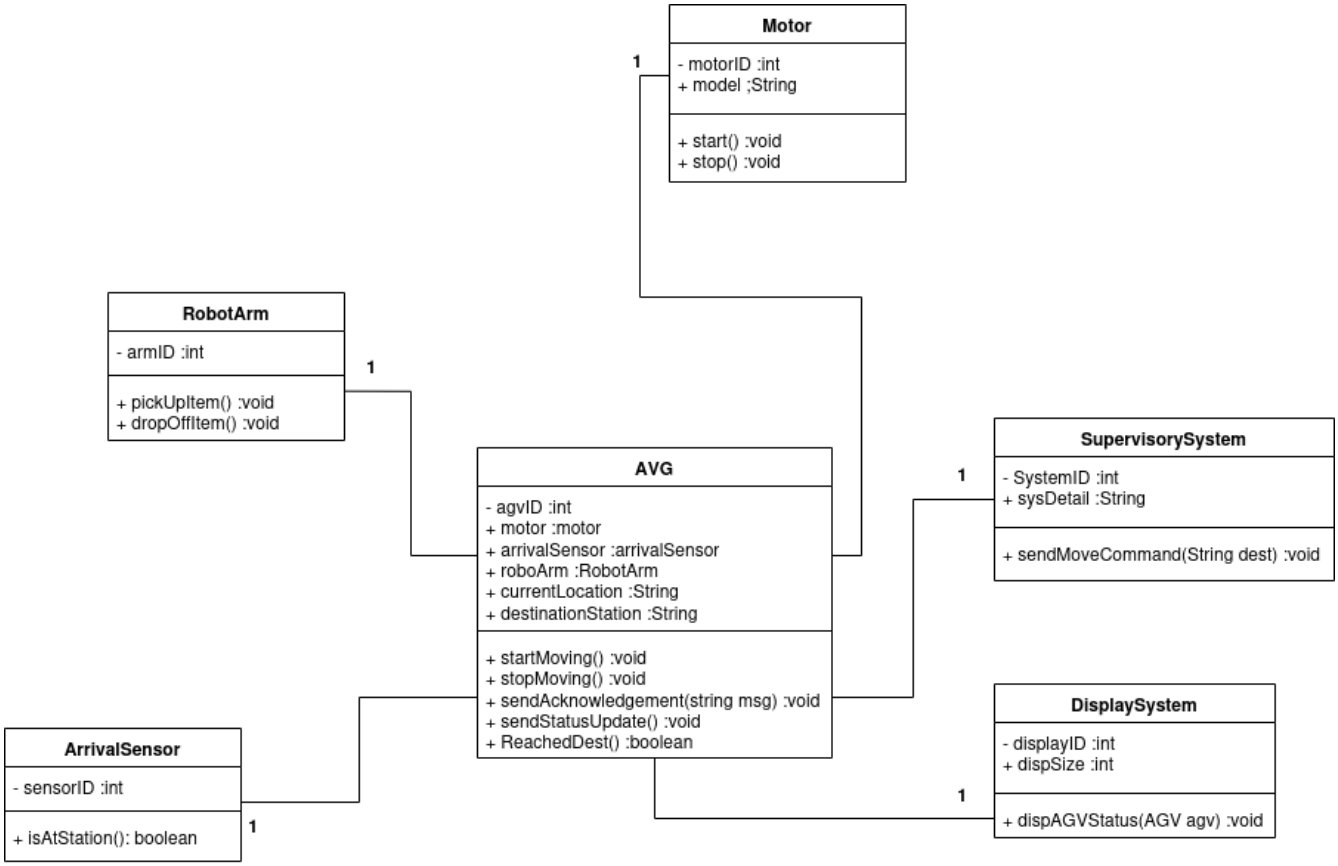
Class titles are the specific attributes of a given class.

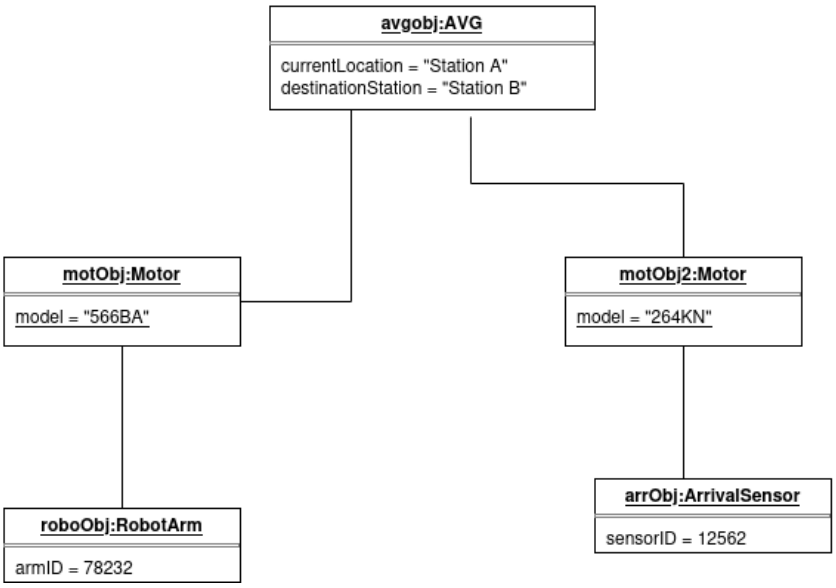
Links

Links are the lines that connect two shapes of an object diagram to each other.



Object Diagram





EXPERIMENT-6

Aim- Create a sequence diagram of Amizone website.

Software Used- Draw.io

Theory:-

Sequence Diagram:

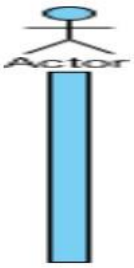
- Sequence Diagrams are interaction diagrams that detail how operations are carried out.
- They capture the interaction between objects in the context of a collaboration.
- Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to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

Dimensions:

- It has two dimensions.
- The horizontal axis shows the objects.
- The vertical axis shows the time from top to bottom.

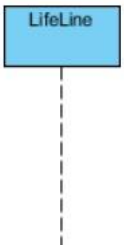
Actor:

- A type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data).
- External to the subject.
- Represent roles played by human users, external hardware, or other subjects.



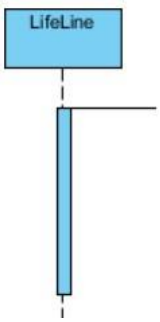
Lifeline:

A lifeline represents an individual participant in the Interaction.



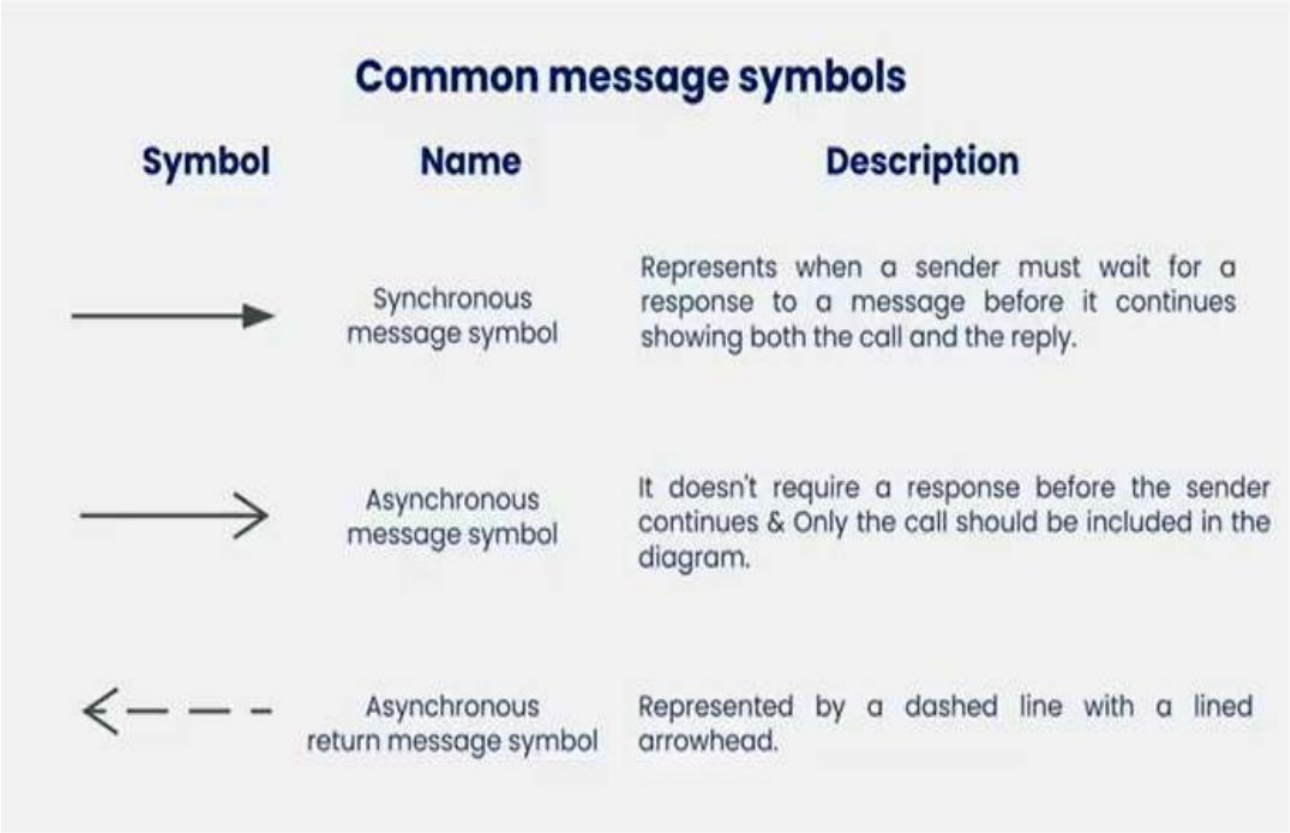
Activations:

- A thin rectangle on a lifeline) represents the period during which an element is performing an operation.
- The top and the bottom of the rectangle are aligned with the initiation and the completion time respectively.



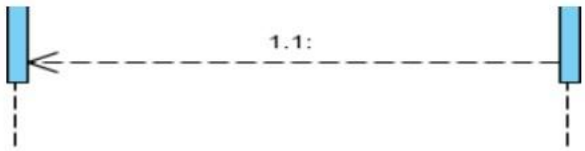
Call Message:

- A message defines a particular communication between Lifelines of an Interaction.
- Call message is an invocation of operation of target lifeline.



Return Message:

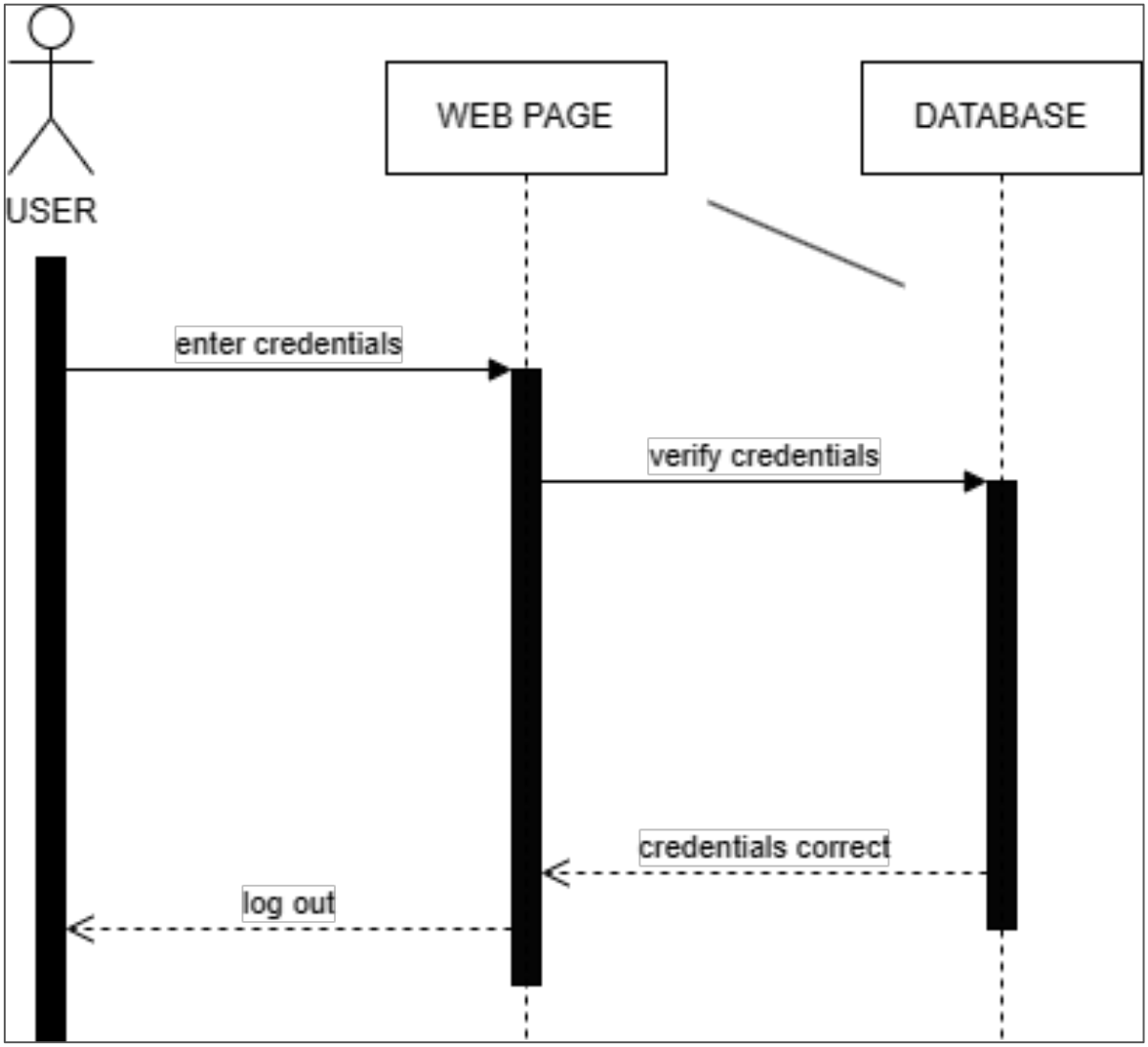
- A message defines a particular communication between Lifelines of an Interaction.
- Return message is a kind of message that represents the pass of information back to the caller of a corresponded former message.



Create Message:

- A message defines a particular communication between Lifelines of an Interaction.
- Create message is a kind of message that represents the instantiation of (target) lifeline.

Sequence Diagram:



EXPERIMENT-7

Aim- Create a Collaboration Diagram for Amizone.

Software Used- Draw.io

Theory:-

A Collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some tasks.

A Collaboration defines a set of participants and relationships that are meaningful for a given set of purposes.

Actors

Normally an actor instance occurs in the collaboration diagram, as the invoker of the interaction.

Objects

An object is represented by an object symbol showing the name of the object and its class underlined, separated by a colon:

Object_name : class_name

Links

Links connect objects and actors and are instances of associations and each link corresponds to an association in the class diagram.

Messages

A message is a communication between objects that conveys information with the expectation that activity will ensue.

The message is directed from sender to receiver.

Collaboration Diagram