unified modelling language

UML (Unified Modeling Language) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. It was initially started to capture the behavior of complex software and non-software system and now it has become an OMG standard

OMG is continuously making efforts to create a truly industry standard.

- **UML stands for Unified Modeling Language.**
- **UML is different from the other common programming languages such as C++, Java, COBOL, etc.**
- **UML is a pictorial language used to make software blueprints.**
- **UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.**
- **Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.**

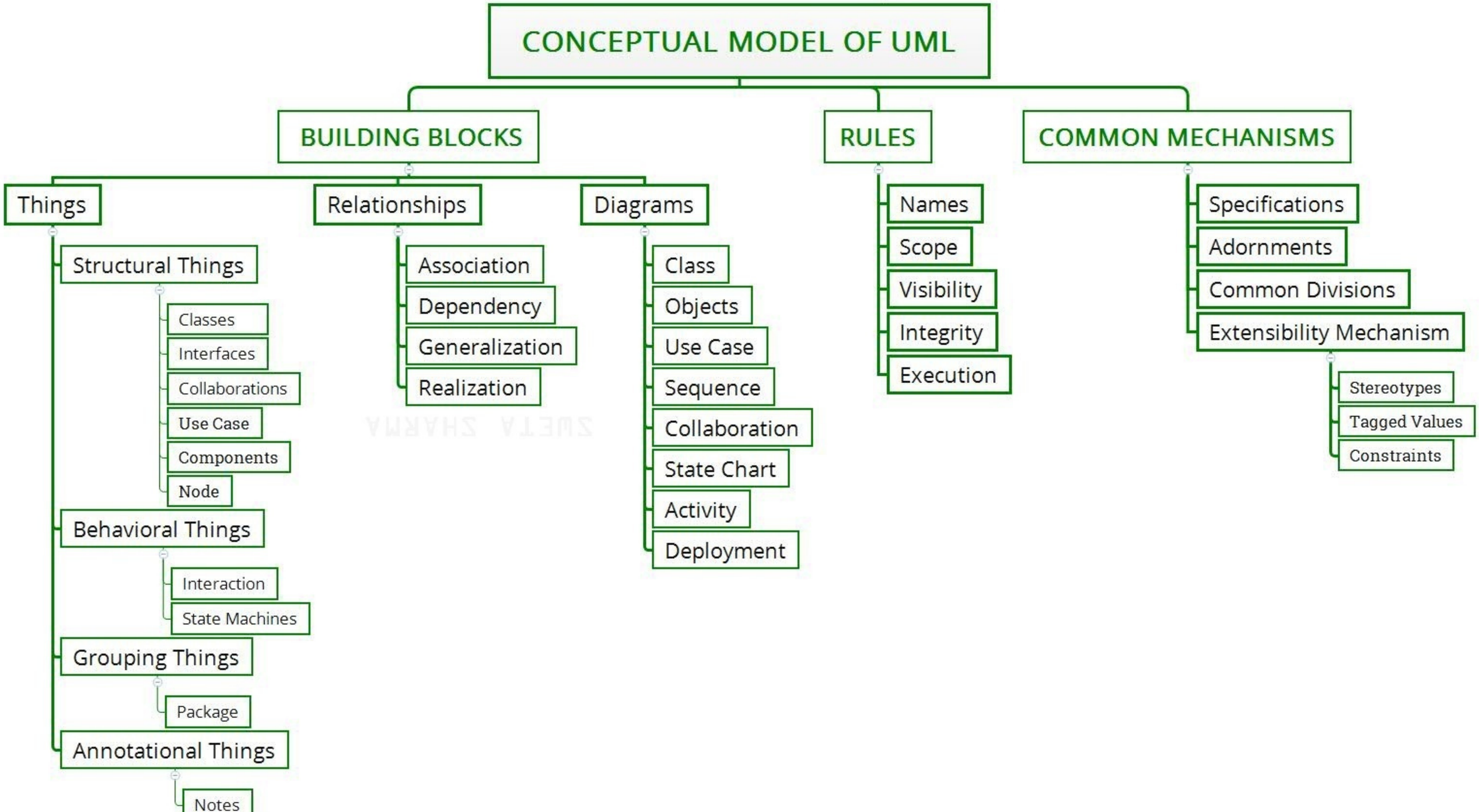
UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard.

- **The conceptual model of UML can be mastered by learning the following three major elements –**
- **UML building blocks**
- **Rules to connect the building blocks**
- **Common mechanisms of UML**

There are three basic steps where the OO concepts are applied and implemented. The steps can be defined as
OO Analysis → OO Design → OO implementation using OO languages

The building blocks of UML can be defined as −

- **Things**
- **Relationships**
- **Diagrams**


CONCEPTUAL MODEL OF UML
BUILDING BLOCKS
RULES
COMMON MECHANISMS
Things
Relationships
Diagrams
Structural Things
Classes
Interfaces
Collaborations
Use Case
Components
Node
Behavioral Things
Interaction
State Machines
Grouping Things
Package
Annotational Things
Notes
Association
Dependency
Generalization
Realization
Class
Objects
Use Case
Sequence
Collaboration
State Chart
Activity
Deployment
Names
Scope
Visibility
Integrity
Execution
Specifications
Adornments
Common Divisions
Extensibility Mechanism
Stereotypes
Tagged Values
Constraints

- **Structural Things**
- **Structural things define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.**
- **Class − Class represents a set of objects having similar responsibilities.**
- **Interface − Interface defines a set of operations, which specify the responsibility of a class.**
- **Collaboration −Collaboration defines an interaction between elements.**
- **Use case −Use case represents a set of actions performed by a system for a specific goal.**
- **Component −Component describes the physical part of a system.**
- **Node − A node can be defined as a physical element that exists at run time.**

- **Behavioral Things**
- **A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things −**
- **Interaction − Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.**

- **Grouping Things**
- **Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available −**
- **Package − Package is the only one grouping thing available for gathering structural and behavioral things.**

- **Annotational Things**
- **Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note - It is the only one Annotational thing available. A note is used to render comments, constraints, etc. of an UML element.**

- **Relationship**
- **Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.**
- **There are four kinds of relationships available.**
- **Dependency**
- **Dependency is a relationship between two things in which change in one element also affects the other.**
- **Association**
- **Association is basically a set of links that connects the elements of a UML model. It also describes how many objects are taking part in that relationship.**
- **Generalization**
- **Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.**
- **Realization**
- **Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.**

- **UML Diagrams**
- **UML diagrams are the ultimate output of the entire discussion. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system.**
- **The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete.**
- **UML includes the following nine diagrams, the details of which are described in the subsequent chapters.**
- **Class diagram**
- **Object diagram**
- **Use case diagram**
- **Sequence diagram**
- **Collaboration diagram**
- **Activity diagram**
- **Statechart diagram**
- **Deployment diagram**
- **Component diagram**

- **UML plays an important role in defining different perspectives of a system. These perspectives are −**
- **Design**
- **Implementation**
- **Process**
- **Deployment**
- **The center is the Use Case view which connects all these four. A Use Case represents the functionality of the system. Hence, other perspectives are connected with use case.**
- **Design of a system consists of classes, interfaces, and collaboration. UML provides class diagram, object diagram to support this.**
- **Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support the implementation perspective.**
- **Process defines the flow of the system. Hence, the same elements as used in Design are also used to support this perspective.**
- **Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective.**

modelling

- **Structural Modeling**
- **Structural modeling captures the static features of a system. They consist of the following –**
- **Classes diagrams**
- **Objects diagrams**
- **Deployment diagrams**
- **Package diagrams**
- **Composite structure diagram**
- **Component diagram**
- **Structural model represents the framework for the system and this framework is the place where all other components exist. Hence, the class diagram, component diagram and deployment diagrams are part of structural modeling. They all represent the elements and the mechanism to assemble them.**
- **The structural model never describes the dynamic behavior of the system. Class diagram is the most widely used structural diagram.**

- **Behavioral Modeling**
- **Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system. They consist of the following –**
- **Activity diagrams**
- **Interaction diagrams**
- **Use case diagrams**
- **All the above show the dynamic sequence of flow in a system.**

- **Architectural Modeling**
- **Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system. Architectural model can be defined as the blueprint of the entire system. Package diagram comes under architectural modeling.**

Structural Things

Graphical notations used in structural things are most widely used in UML. These are considered as the nouns of UML models. Following are the list of structural things.

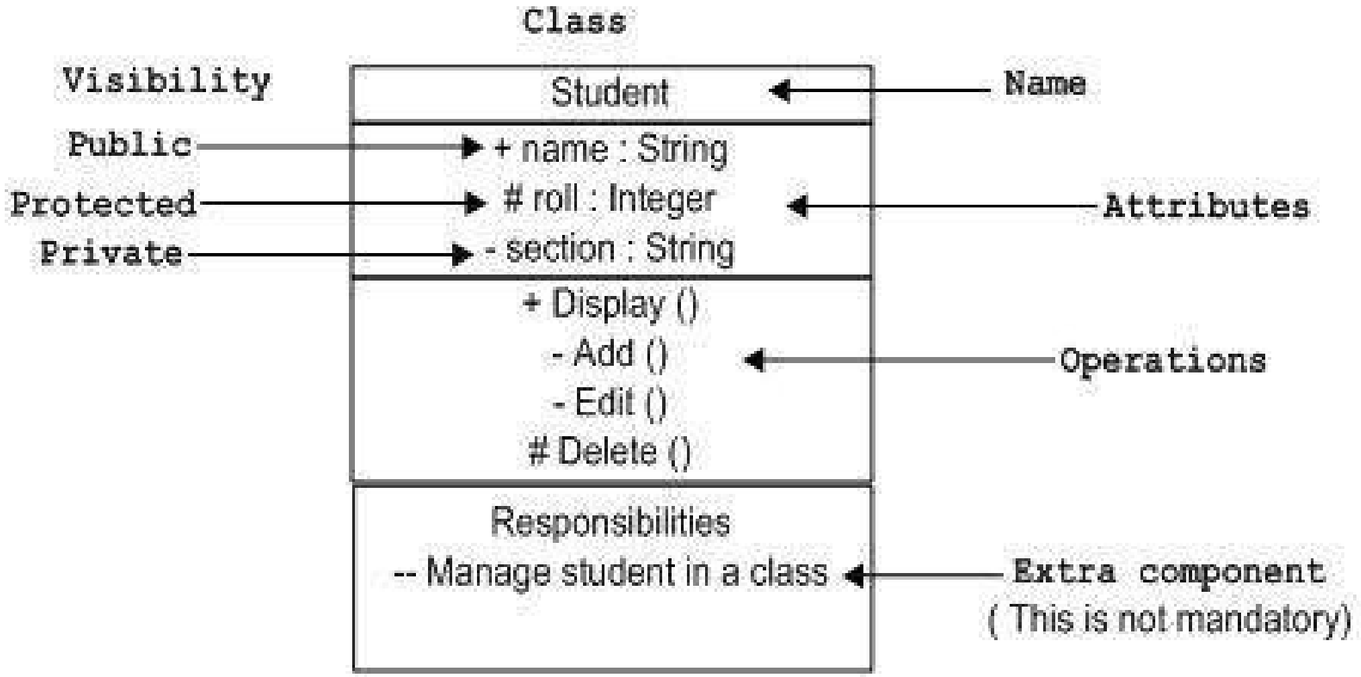
- **Classes**
- **Object**
- **Interface**
- **Collaboration**
- **Use case**
- **Active classes**
- **Components**
- **Nodes**

Class Notation

UML class is represented by the following figure. The diagram is divided into four parts.

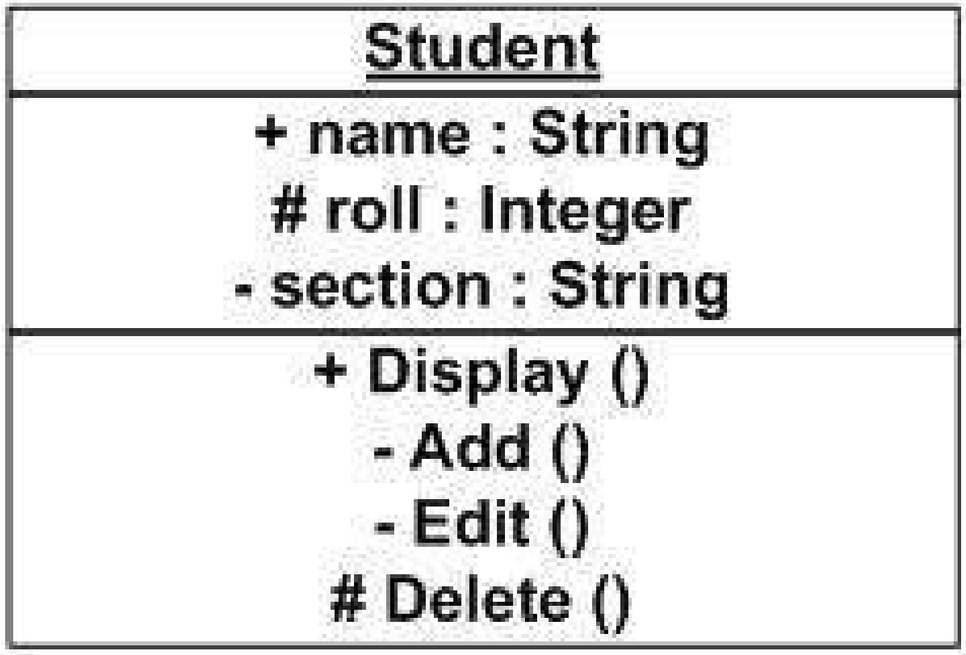
- **The top section is used to name the class.**
- **The second one is used to show the attributes of the class.**
- **The third section is used to describe the operations performed by the class.**
- **The fourth section is optional to show any additional components.**

Classes are used to represent objects. Objects can be anything having properties and responsibility.



Object Notation

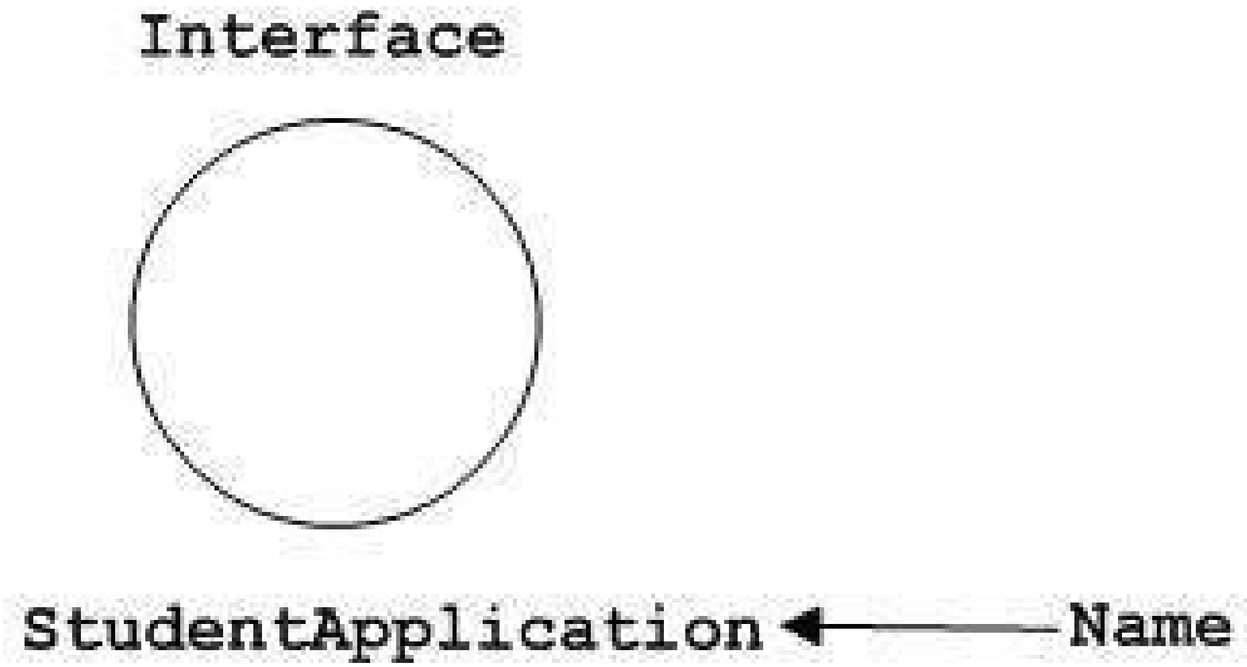
The object is represented in the same way as the class. The only difference is the name which is underlined as shown in the following figure.
As the object is an actual implementation of a class, which is known as the instance of a class. Hence, it has the same usage as the class.



Interface Notation

Interface is represented by a circle as shown in the following figure. It has a name which is generally written below the circle.

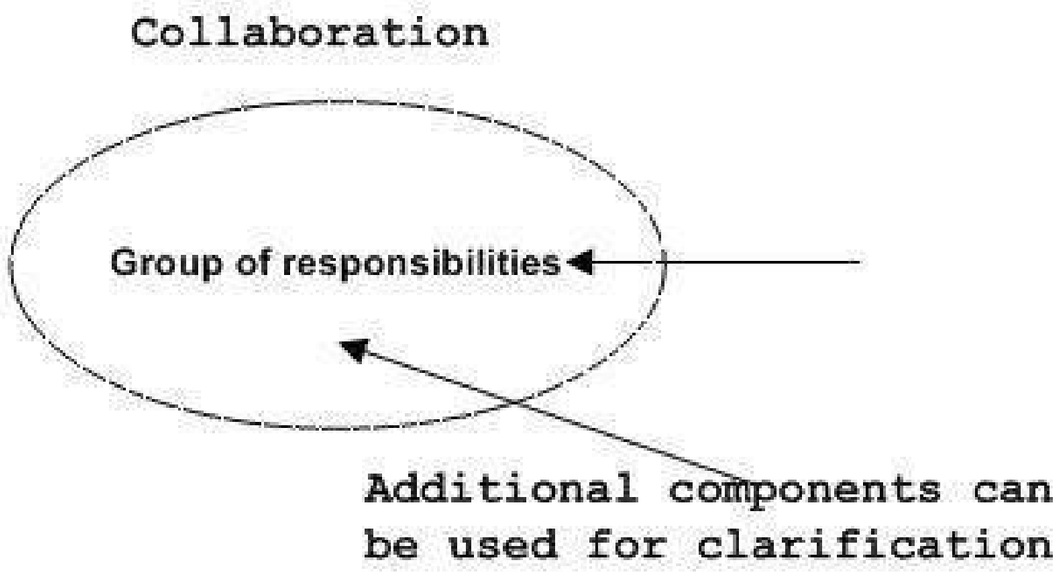
Interface is used to describe the functionality without implementation. Interface is just like a template where you define different functions, not the implementation. When a class implements the interface, it also implements the functionality as per requirement.



Collaboration Notation

Collaboration is represented by a dotted eclipse as shown in the following figure. It has a name written inside the eclipse.

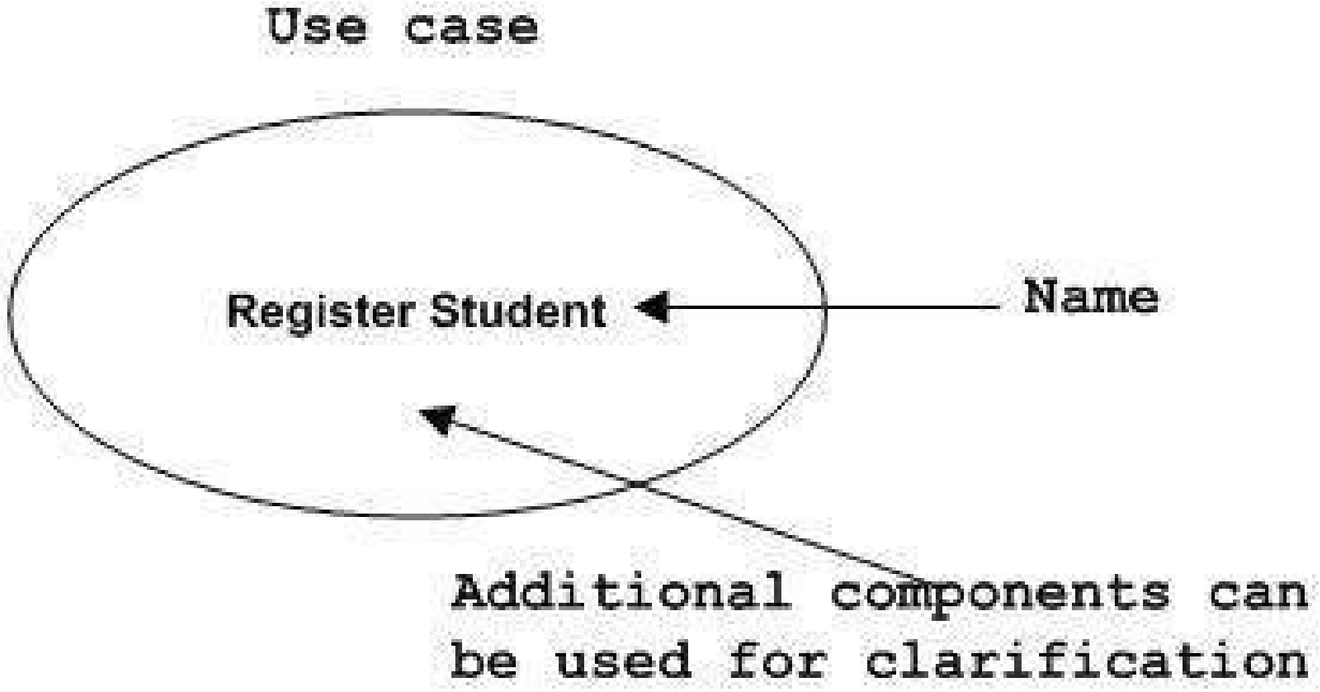
Collaboration represents responsibilities. Generally, responsibilities are in a group.



Use Case Notation

Use case is represented as an eclipse with a name inside it. It may contain additional responsibilities.

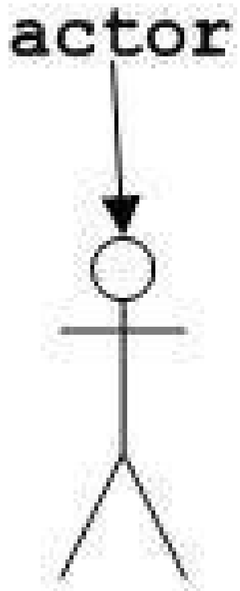
Use case is used to capture high level functionalities of a system.



Actor Notation

An actor can be defined as some internal or external entity that interacts with the system.

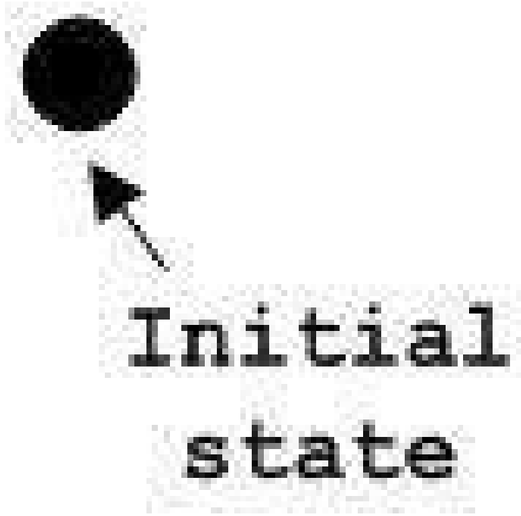
An actor is used in a use case diagram to describe the internal or external entities.



Initial State Notation

Initial state is defined to show the start of a process. This notation is used in almost all diagrams.

The usage of Initial State Notation is to show the starting point of a process.



Final State Notation

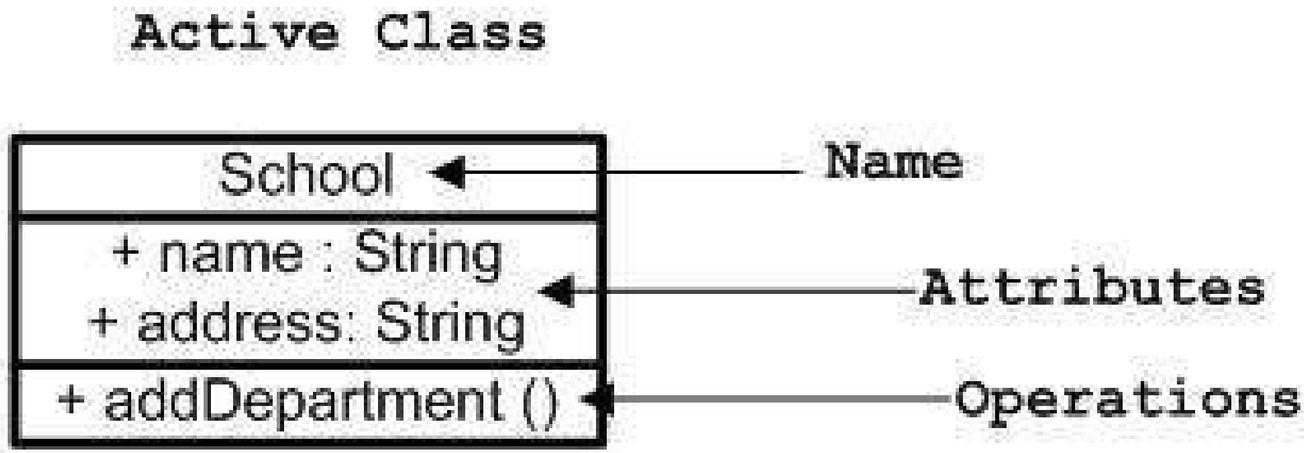
Final state is used to show the end of a process. This notation is also used in almost all diagrams to describe the end.

The usage of Final State Notation is to show the termination point of a process.

Active Class Notation

Active class looks similar to a class with a solid border. Active class is generally used to describe the concurrent behavior of a system.

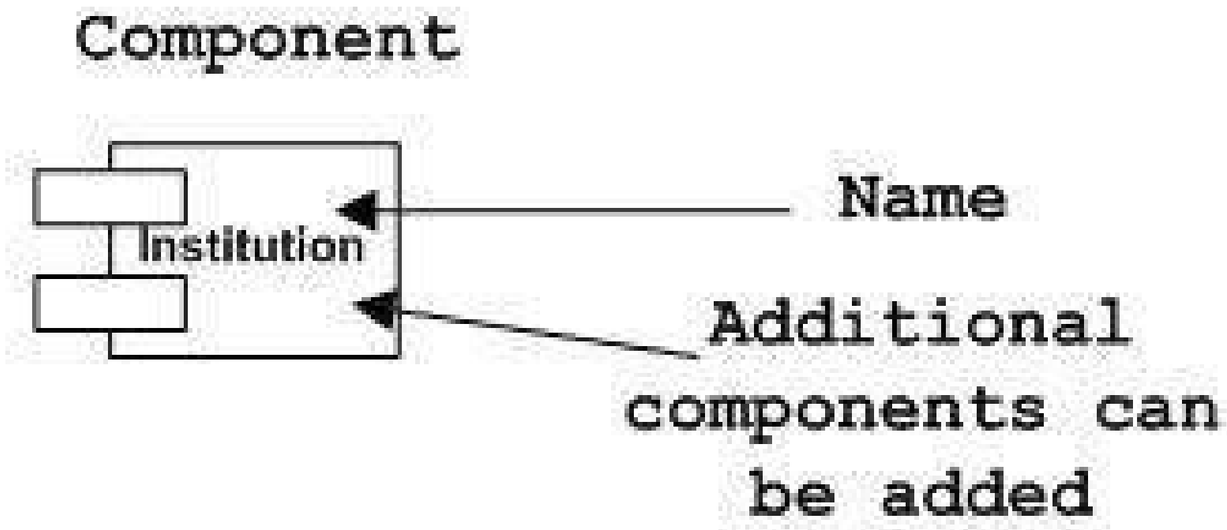
Active class is used to represent the concurrency in a system.



Component Notation

A component in UML is shown in the following figure with a name inside. Additional elements can be added wherever required.

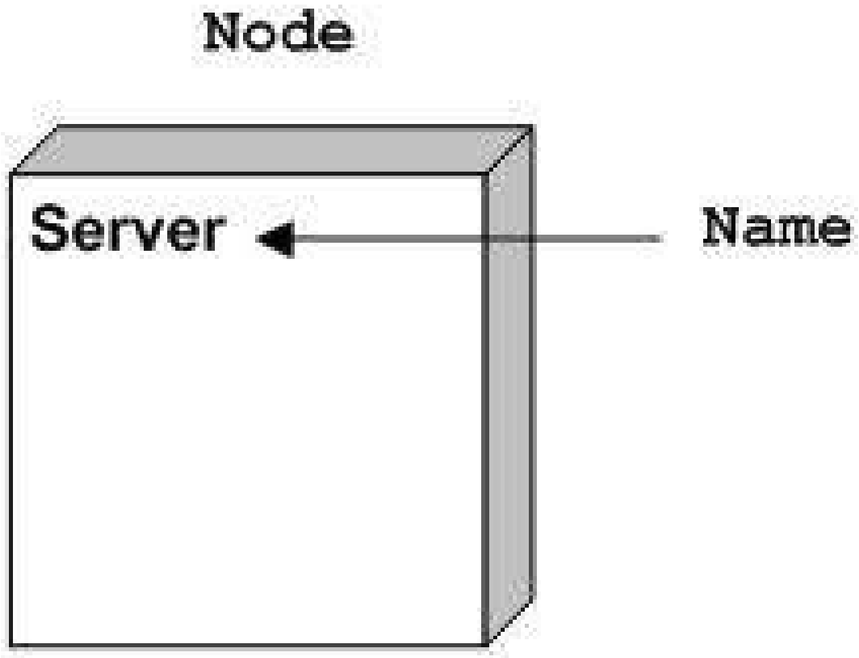
Component is used to represent any part of a system for which UML diagrams are made.



Node Notation

A node in UML is represented by a square box as shown in the following figure with a name. A node represents the physical component of the system.

Node is used to represent the physical part of a system such as the server, network, etc.



Behavioral Things

Dynamic parts are one of the most important elements in UML. UML has a set of powerful features to represent the dynamic part of software and non-software systems. These features include interactions and state machines.

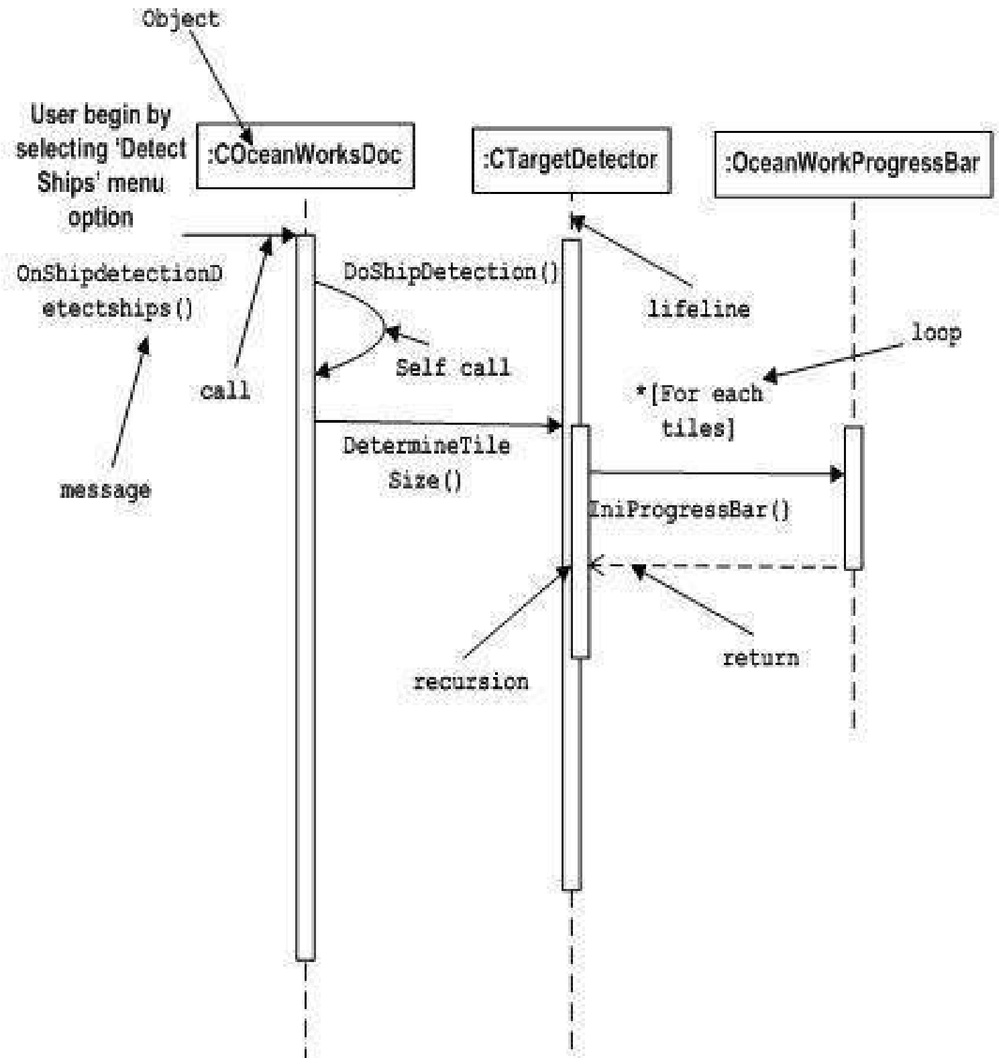
Interactions can be of two types −

- **Sequential (Represented by sequence diagram)**
- **Collaborative (Represented by collaboration diagram)**

Interaction Notation

Interaction is basically a message exchange between two UML components. The following diagram represents different notations used in an interaction.

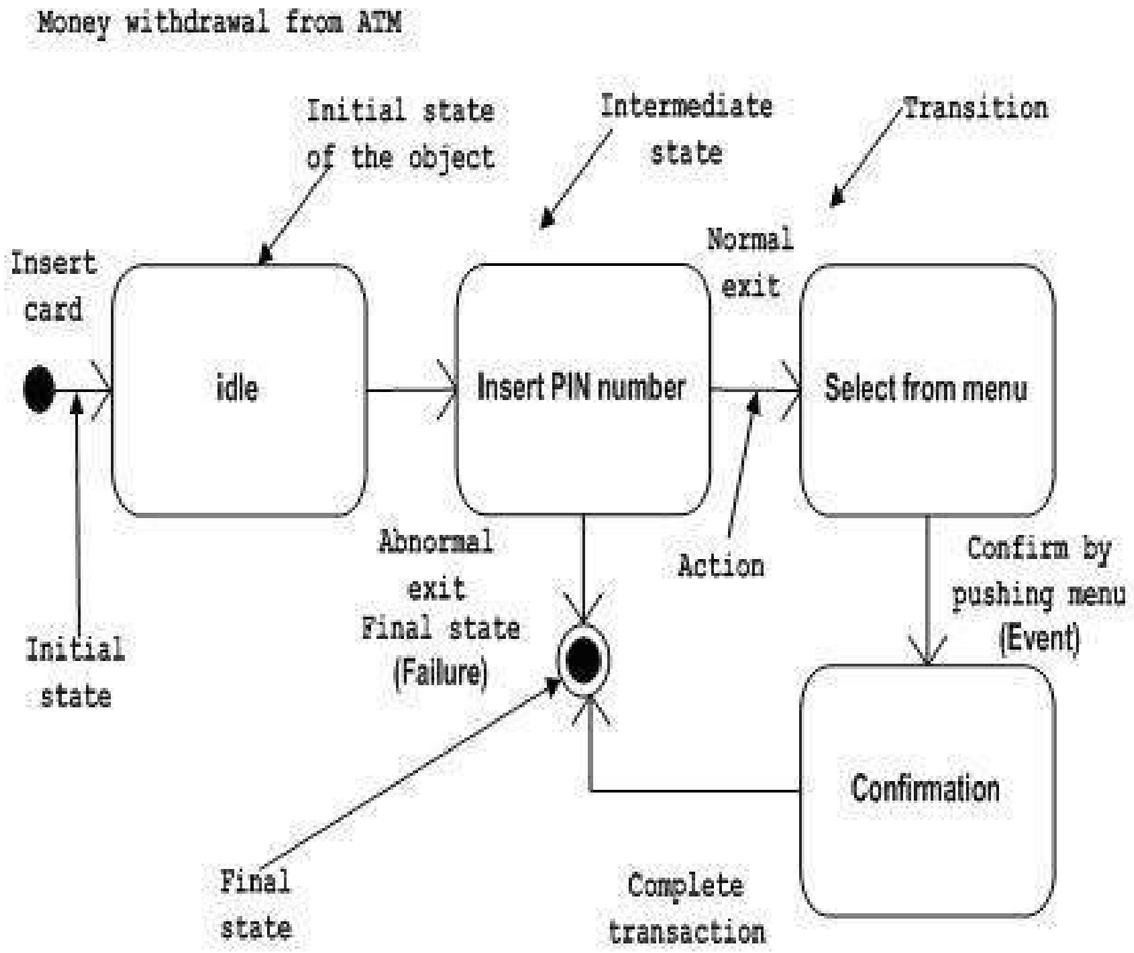
Interaction is used to represent the communication among the components of a system.



State Machine Notation

State machine describes the different states of a component in its life cycle. The notations are described in the following diagram.

State machine is used to describe different states of a system component. The state can be active, idle, or any other depending upon the situation.

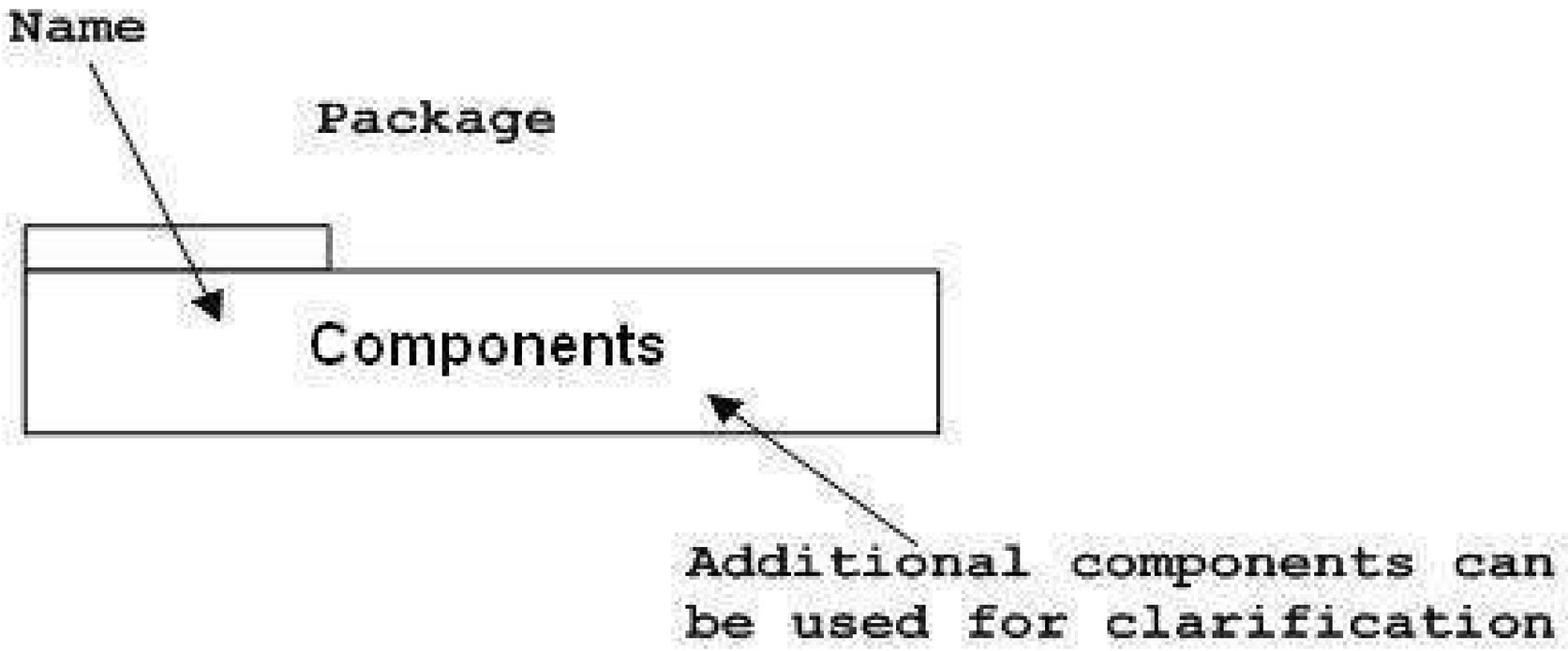


Package Notation

Package notation is shown in the following figure and is used to wrap the components of a system.

Annotational Things

In any diagram, explanation of different elements and their functionalities are very important. Hence, UML has notes notation to support this requirement.



Relationships

A model is not complete unless the relationships between elements are described properly. The Relationship gives a proper meaning to a UML model. Following are the different types of relationships available in UML.

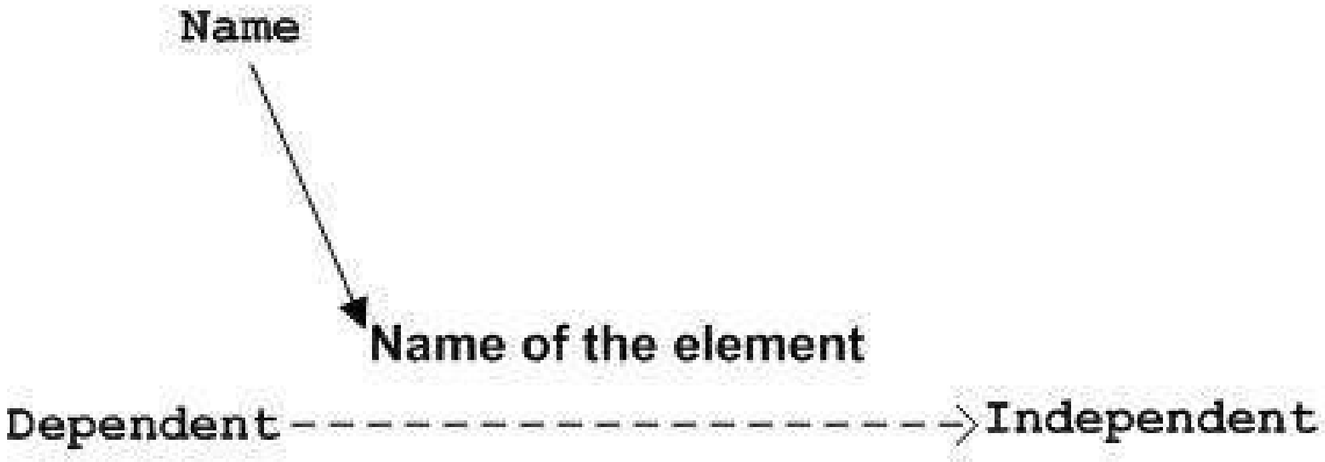
- **Dependency**
- **Association**
- **Generalization**
- **Extensibility**

Dependency Notation

Dependency is an important aspect in UML elements. It describes the dependent elements and the direction of dependency.

Dependency is represented by a dotted arrow as shown in the following figure. The arrow head represents the independent element and the other end represents the dependent element.

Dependency is used to represent the dependency between two elements of a system

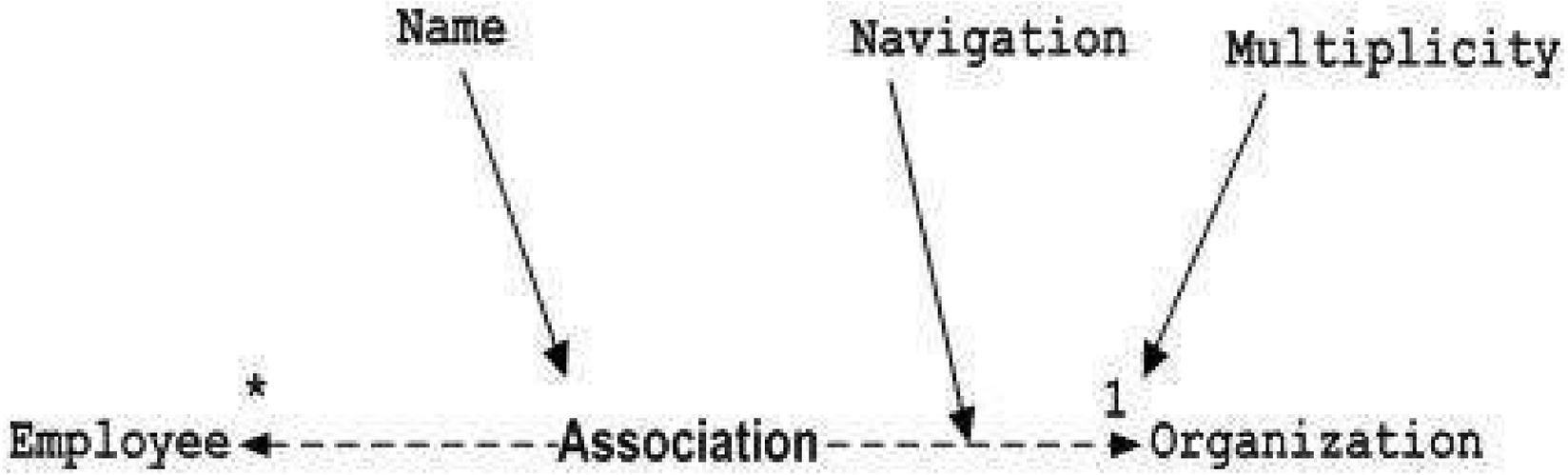


Association Notation

Association describes how the elements in a UML diagram are associated. In simple words, it describes how many elements are taking part in an interaction.

Association is represented by a dotted line with (without) arrows on both sides. The two ends represent two associated elements as shown in the following figure. The multiplicity is also mentioned at the ends (1, *, etc.) to show how many objects are associated.

Association is used to represent the relationship between two elements of a system.



Generalization Notation

Generalization describes the inheritance relationship of the object-oriented world. It is a parent and child relationship.

Generalization is represented by an arrow with a hollow arrow head as shown in the following figure. One end represents the parent element and the other end represents the child element.

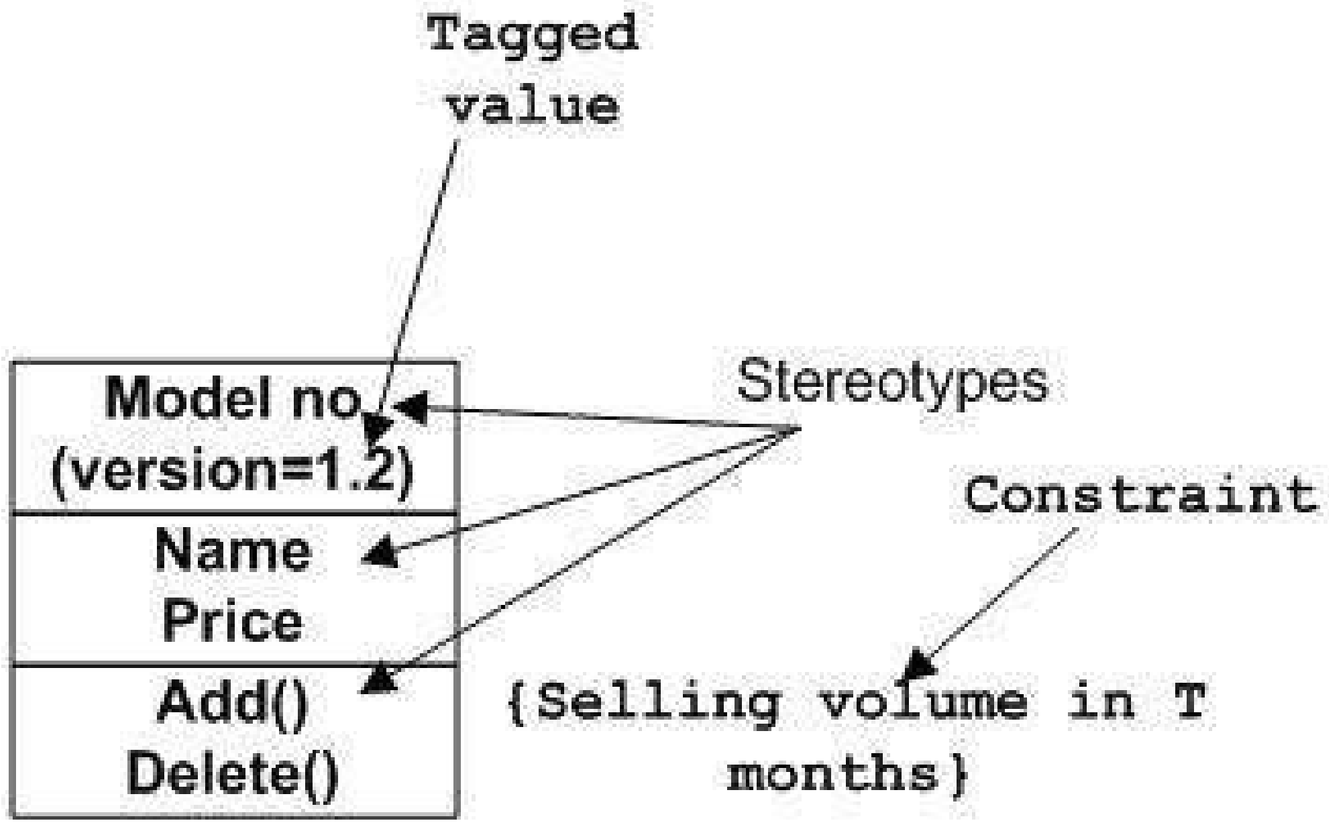
Generalization is used to describe parent-child relationship of two elements of a system.

Extensibility Notation

All the languages (programming or modeling) have some mechanism to extend its capabilities such as syntax, semantics, etc. UML also has the following mechanisms to provide extensibility features.

- **Stereotypes (Represents new elements)**
- **Tagged values (Represents new attributes)**
- **Constraints (Represents the boundaries)**

Extensibility notations are used to enhance the power of the language. It is basically additional elements used to represent some extra behavior of the system. These extra behaviors are not covered by the standard available notations.



- **UML includes the following nine diagrams, the details of which are described in the subsequent chapters.**
- **Class diagram**
- **Object diagram**
- **Use case diagram**
- **Sequence diagram**
- **Collaboration diagram**
- **Activity diagram**
- **Statechart diagram**
- **Deployment diagram**
- **Component diagram**

UML architecture

- Design
- Implementation
- Process
- Deployment

UML modelling

Structural Modeling

Structural modeling captures the static features of a system. They consist of the following −

- Classes diagrams
- Objects diagrams
- Deployment diagrams
- Package diagrams
- Composite structure diagram
- Component diagram

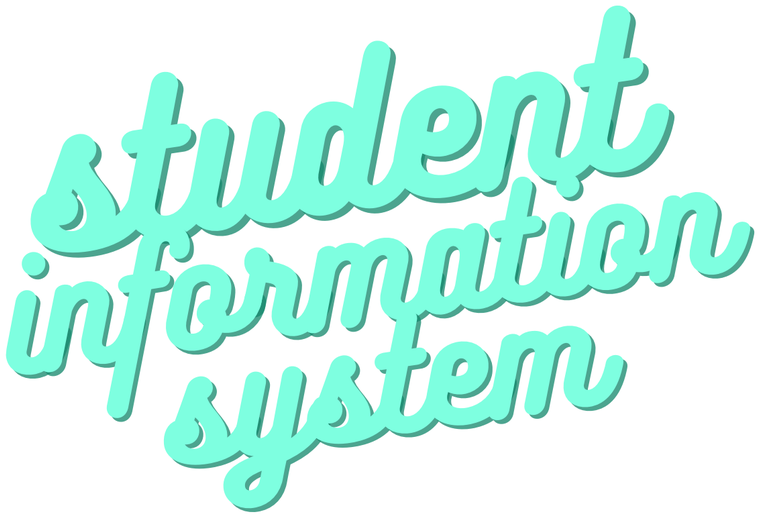
Behavioral Modeling

Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system. They consist of the following –

- **Activity diagrams**
- **Interaction diagrams**
- **Use case diagrams**

Architectural Modeling

Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system. Architectural model can be defined as the blueprint of the entire system. Package diagram comes under architectural modeling.



project

PROBLEM ANALYSIS AND PROJECT PLANNING

A Student information system (SIS) is a software application for educational establishments to manage student data. Student information systems provide capabilities for entering student test and other assessment scores, building student schedules, tracking student attendance, and managing many other student-related data needs in a school, college or university

PROBLEM STATEMENT

a. Effective for Administration Purpose
b. Cheap
c. Better Service

UML DIAGRAMS:

The following UML diagrams describe the process involved in the student information system

a. Use case diagram

b. Class diagram

c. Sequence diagram

d. Collaboration diagram

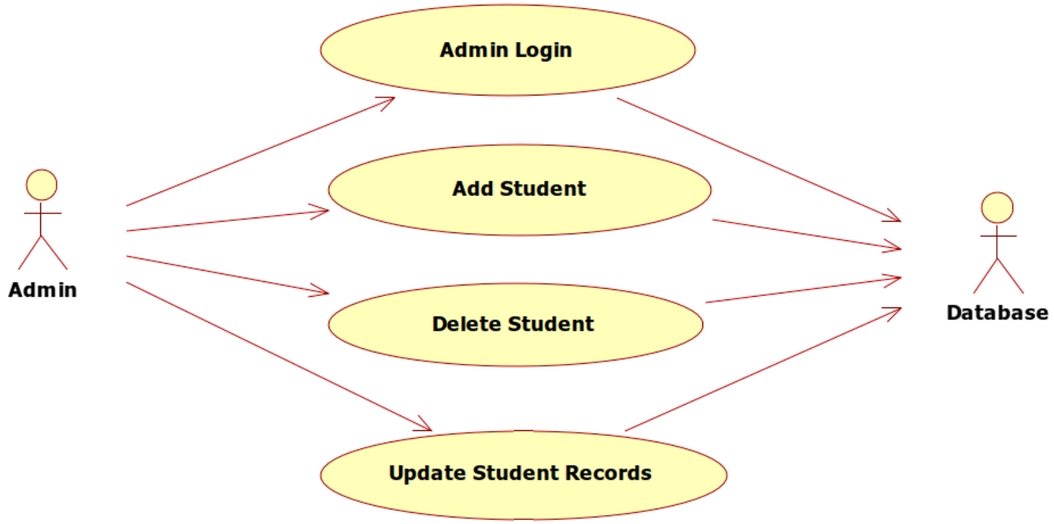
e. Activity diagram

f. Component diagram

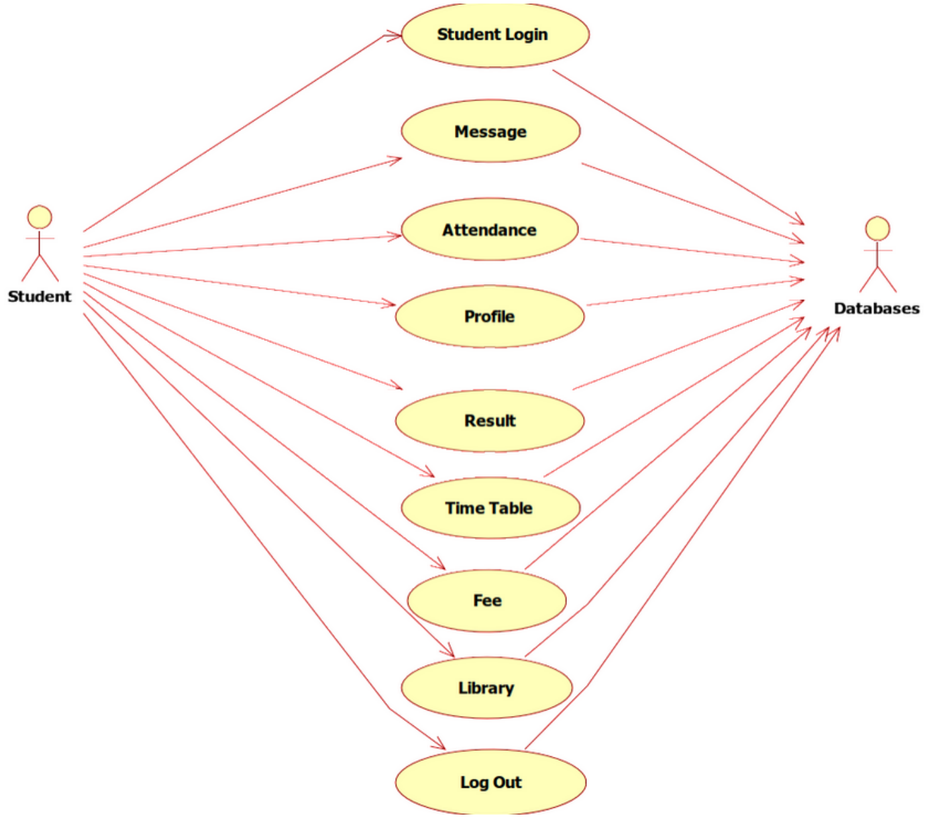
USE CASE DIAGRAM:

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It is represented using ellipse. Actor is any external entity that makes use of the system being modelled. Its represented using stick figure

For Administrator:



For Student:



DOCUMENTATION OF USE CASE DIAGRAM

The actors in this use case diagram are Admin, Student, Database. The use cases are the activities performed by actors.

a. Admin register login, and store the student records details in database.

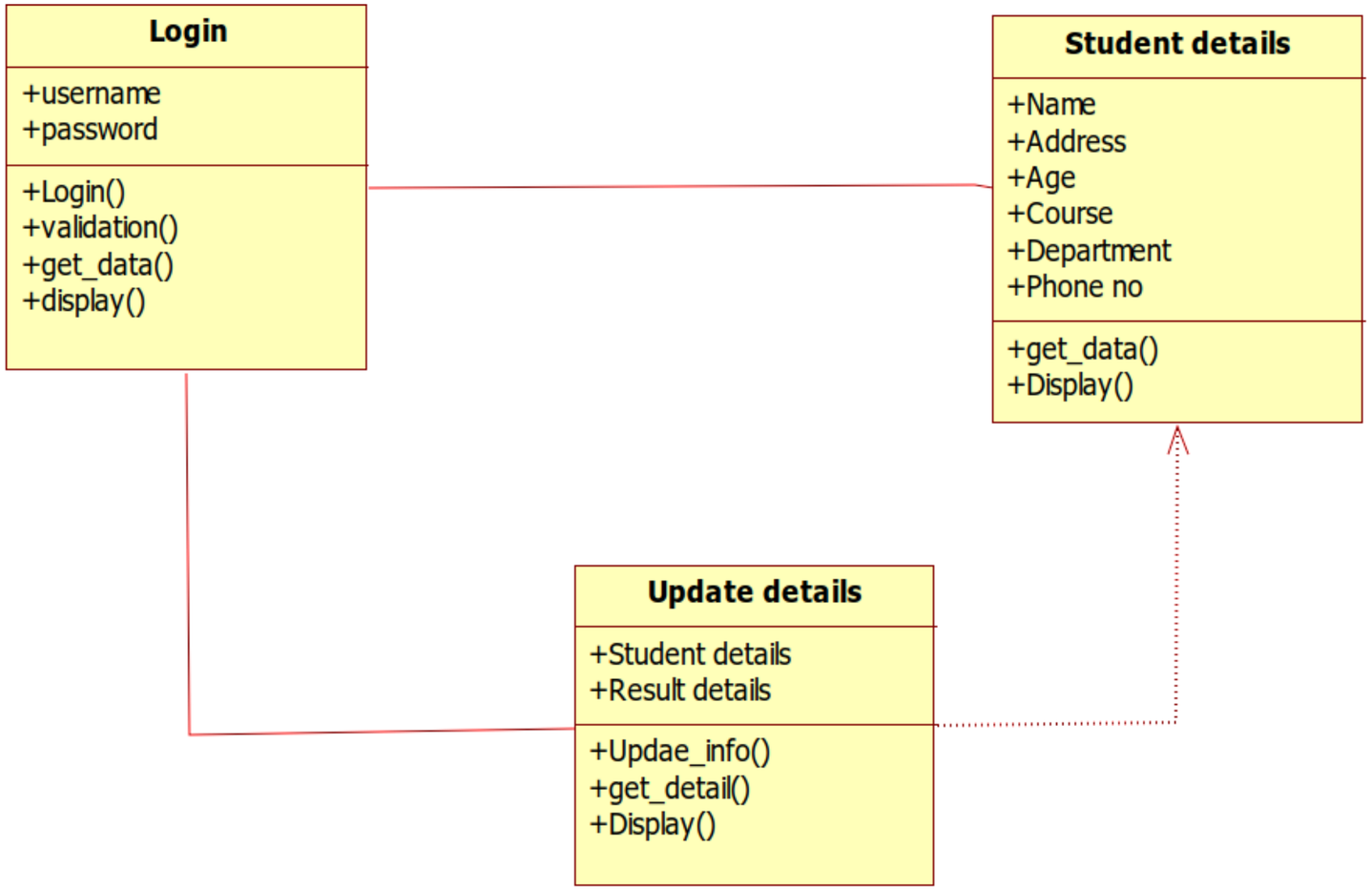
b. Student Register from the Student Login process.

c. Then the database is searched for details and verified.

d. Database stores the details and returns acknowledgement

CLASS DIAGRAM:

A class diagram in the unified modeling language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. It s represented using a rectangle with three compartments. Top compartment have the class name, middle compartment the attributes and the bottom compartment with operations



DOCUMENTATION OF CLASS DIAGRAM

This class diagram has three classes Login, Student details and Update details in database.

a. Students – is the class name. Its attributes are name, Address, DOB, Gender, College, Subjects, Semester, Year, Degree, Branch. The operations Performed in the students class, Store database and Update.

b. Administration– is the class name. Its attributes are Login, Password and database. The operations performed are Student Details store in database and send acknowledgement.

c. Database – is the class name. The operations performed are storing Search and storing the values.

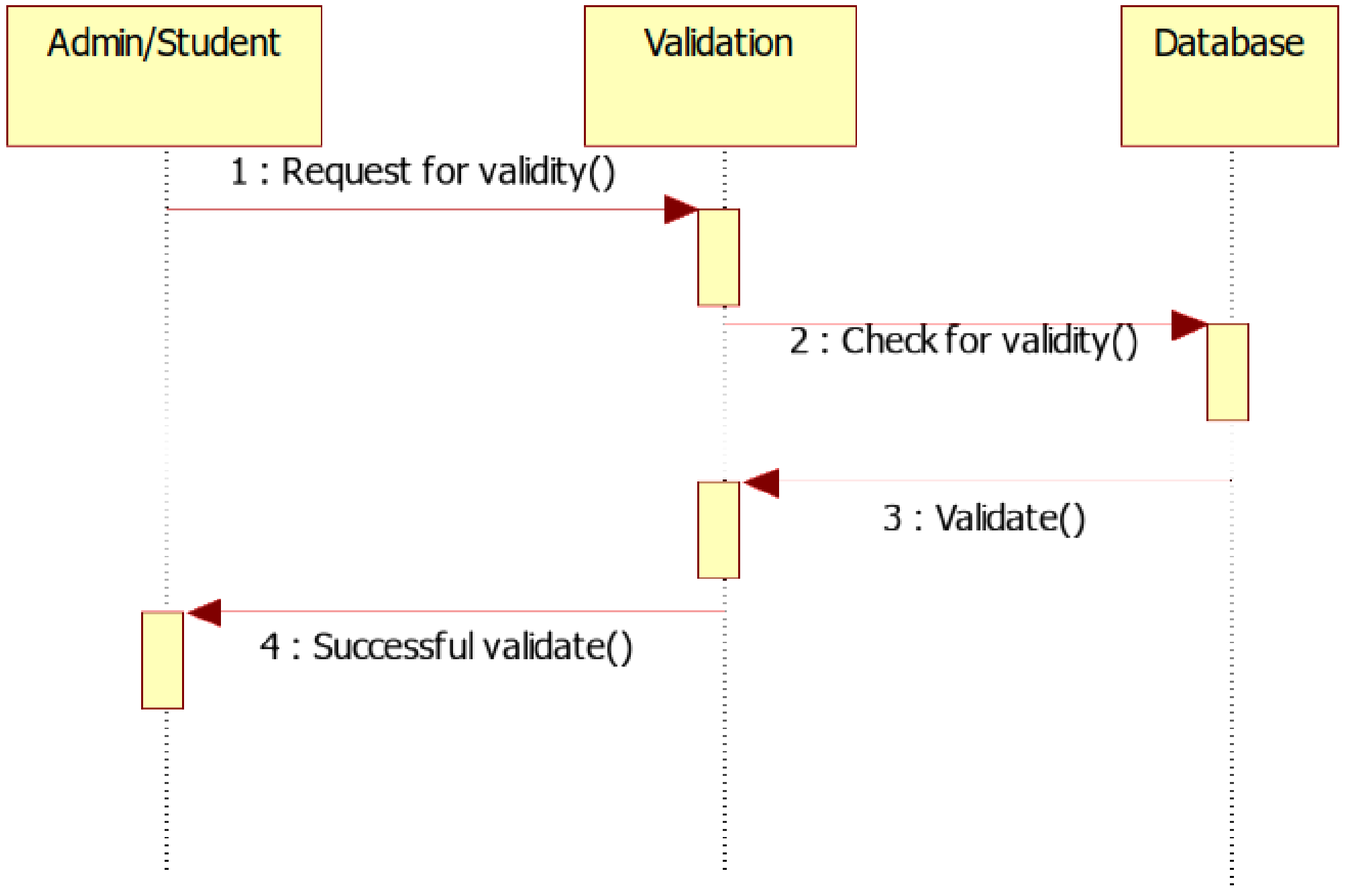
SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. There are two dimensions.

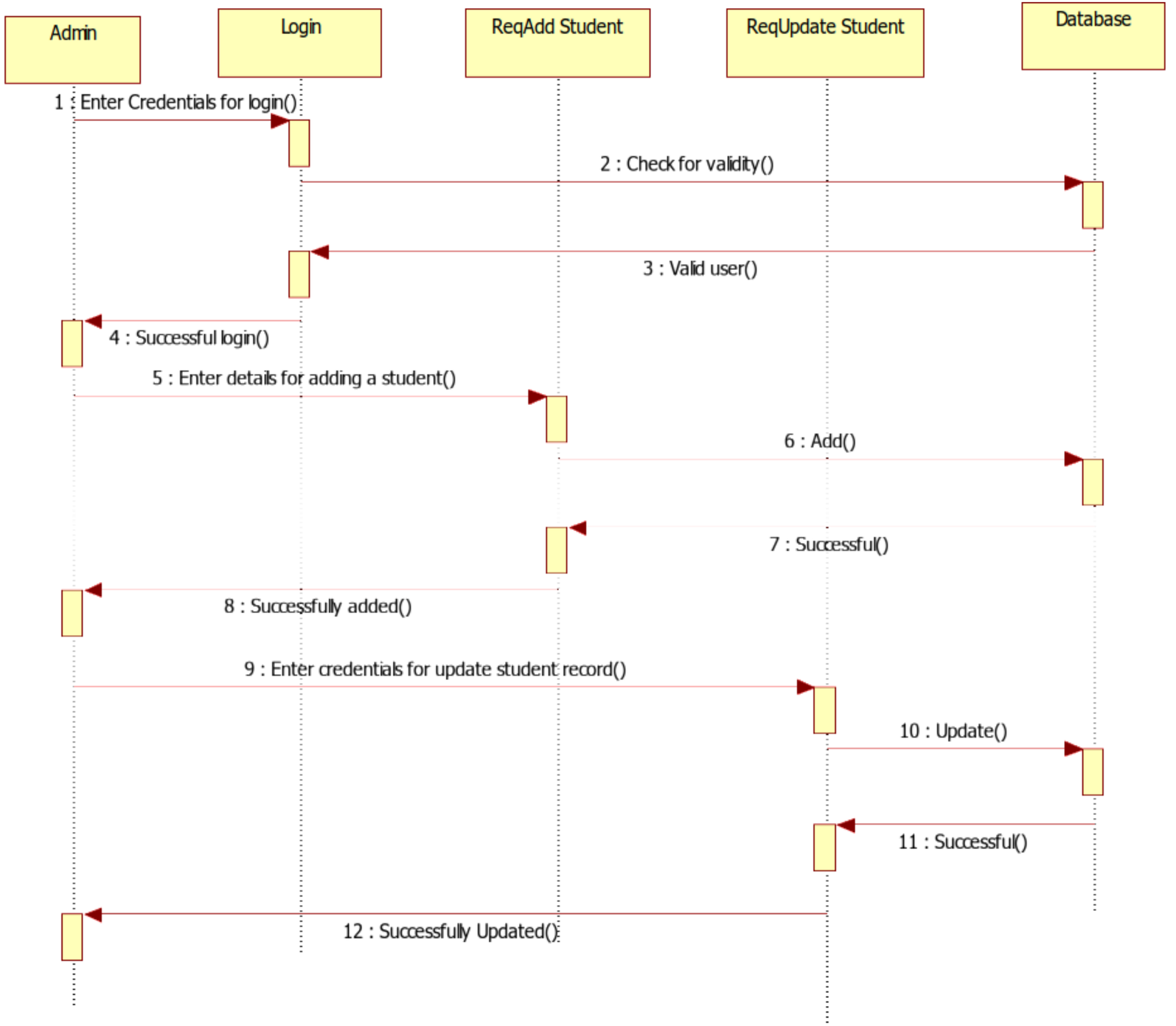
1. Veritcal dimension-represent time.

2. Horizontal dimension-represent different objects.

For validation:



For Administrator:



For Student:

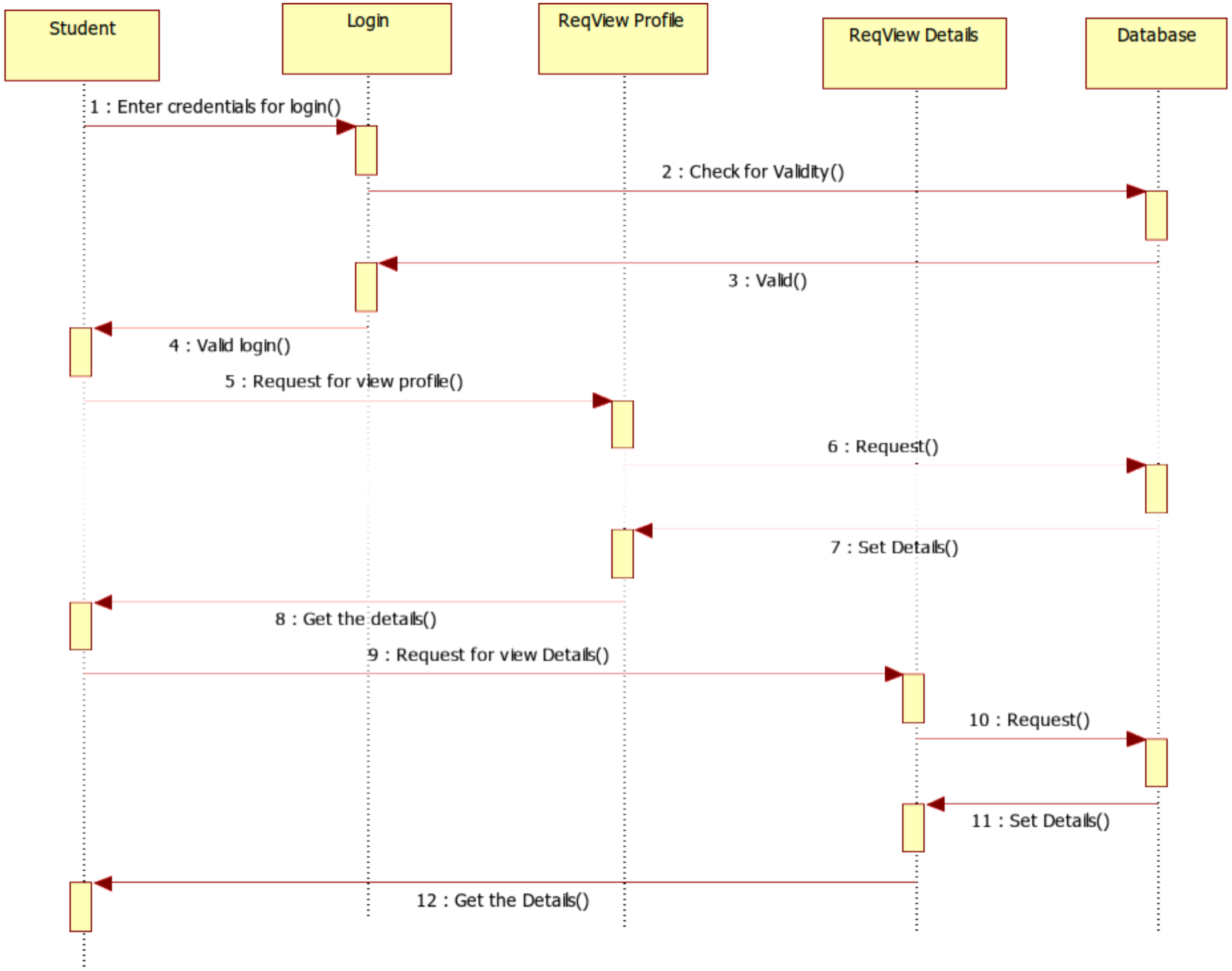
DOCUMENTATION OF SEQUENCE DIAGRAM

The sequence diagram describes the sequence of steps to show

a.The Admin login and registering for Add Stduent Details.

b. The verification done by the interface and sending acknowledgement for registration.

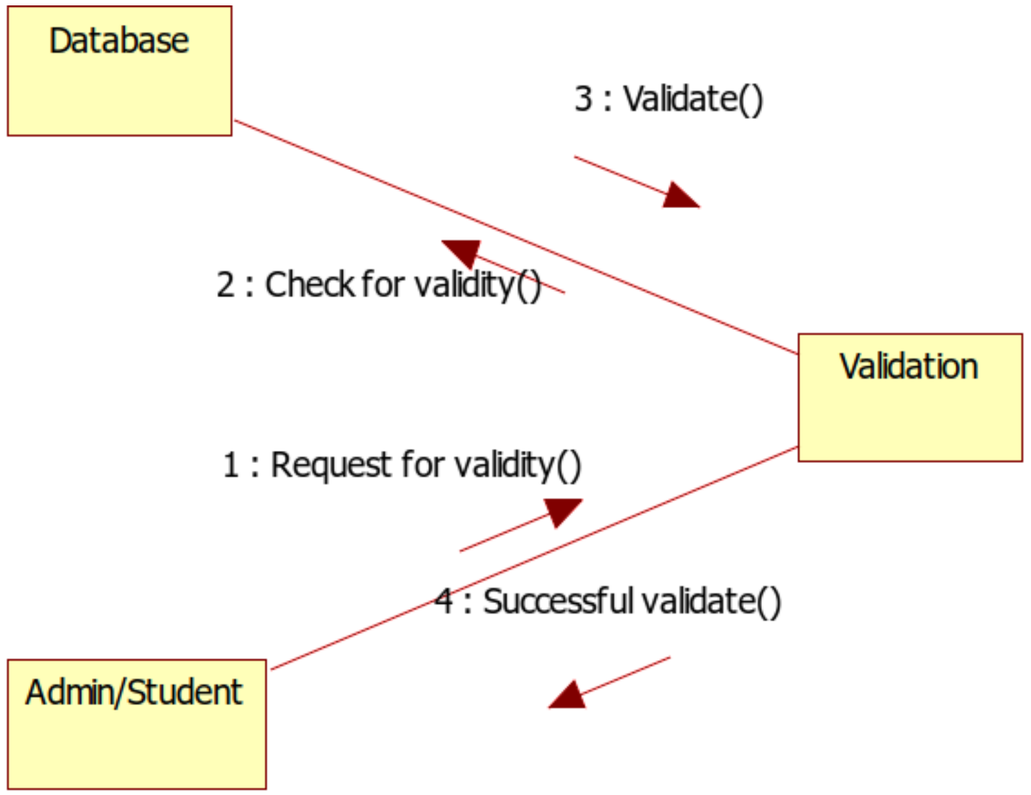
c. Searching the database with login and displaying it for maintenance.



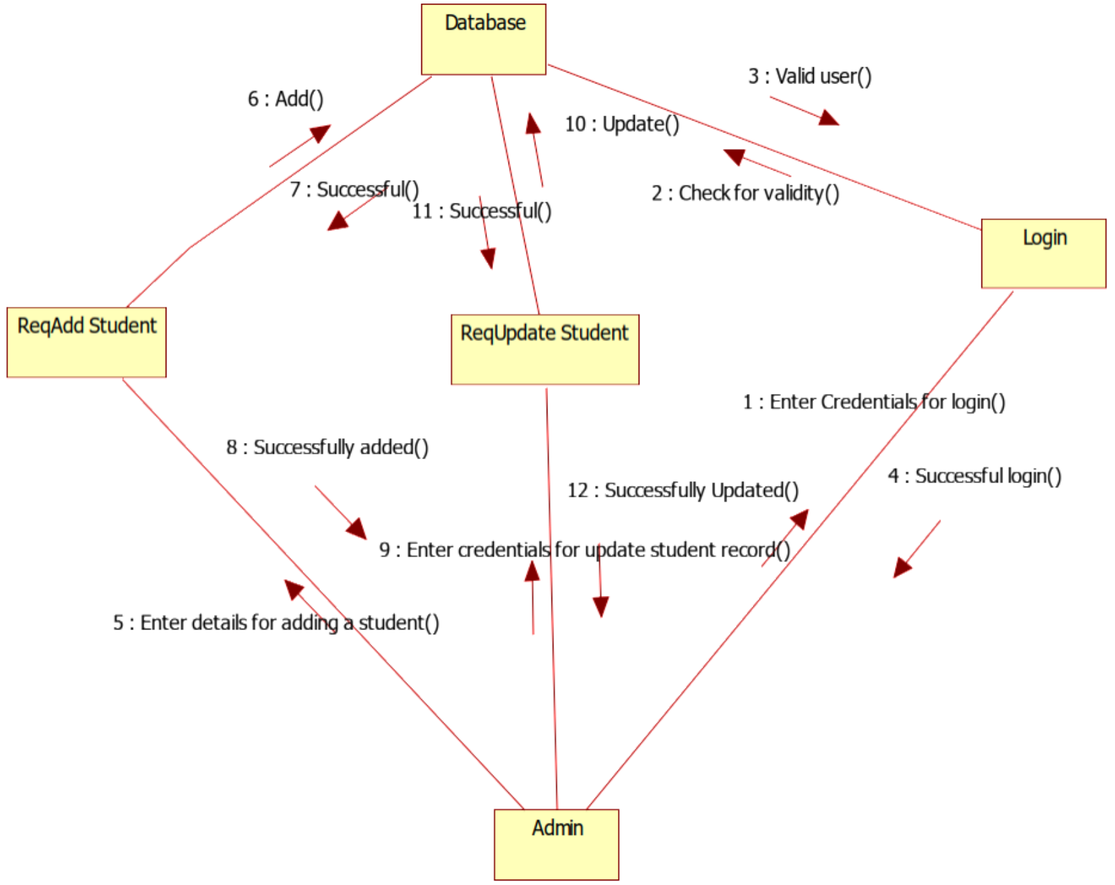
COLLABRATION DIAGRAM:

A collaboration diagram, also called a communication diagram or interaction diagram,. A sophisticated modeling tool can easily convert a collaboration diagram into a sequence diagram and the vice. A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time

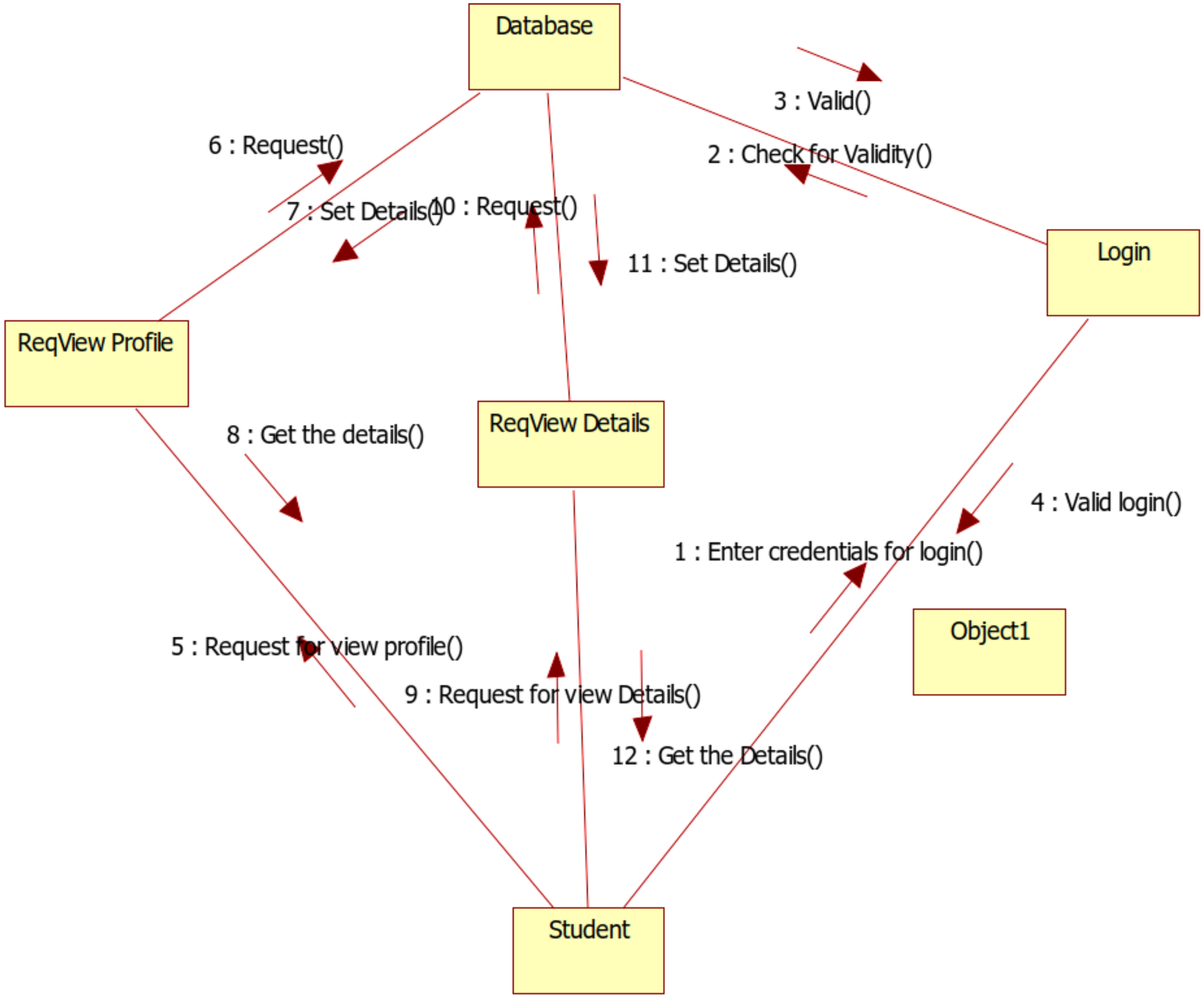
For Validity:



For Administrator:



For Student:



DOCUMENTATION OF COLLABRATION DIAGRAM

The collaboration diagram is to show how the Student registers and the authorities maintains the details of the registered students in the Information system. Here the sequence is numbered according to the flow of execution

ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control. An activity is shown as an rounded box containing the name of the operation.

DOCUMENTATION OF ACTIVITY DIAGRAM

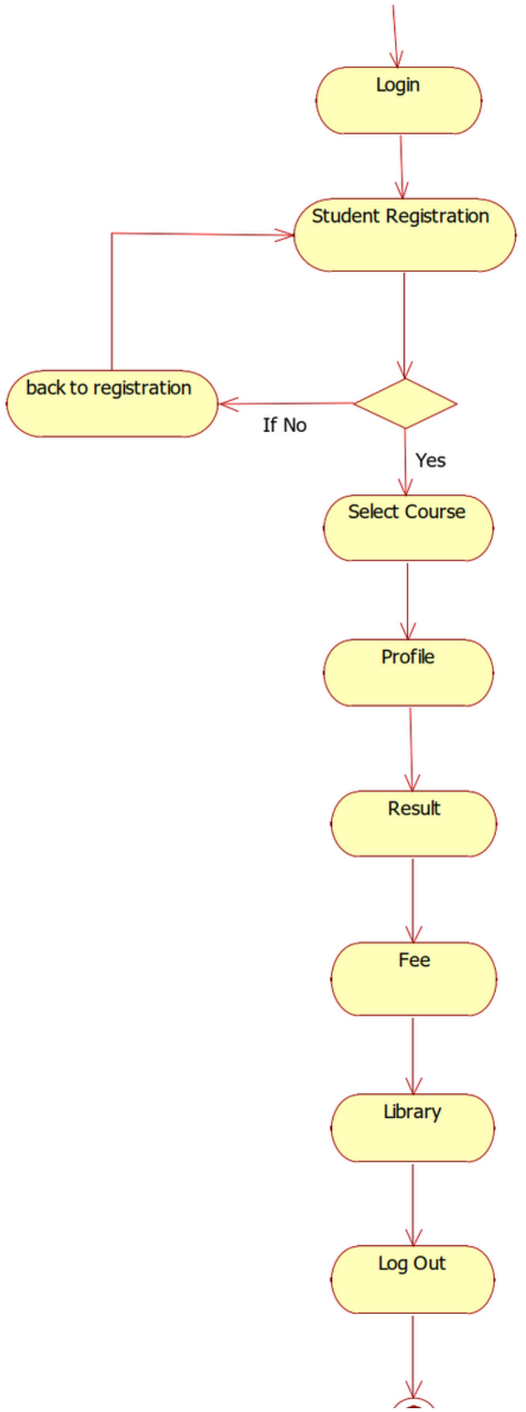
This activity diagram flow of stepwise activities performed in recruitment system.

a. The student details are Add and stored in database.

b. Select the course from the given Course by student.

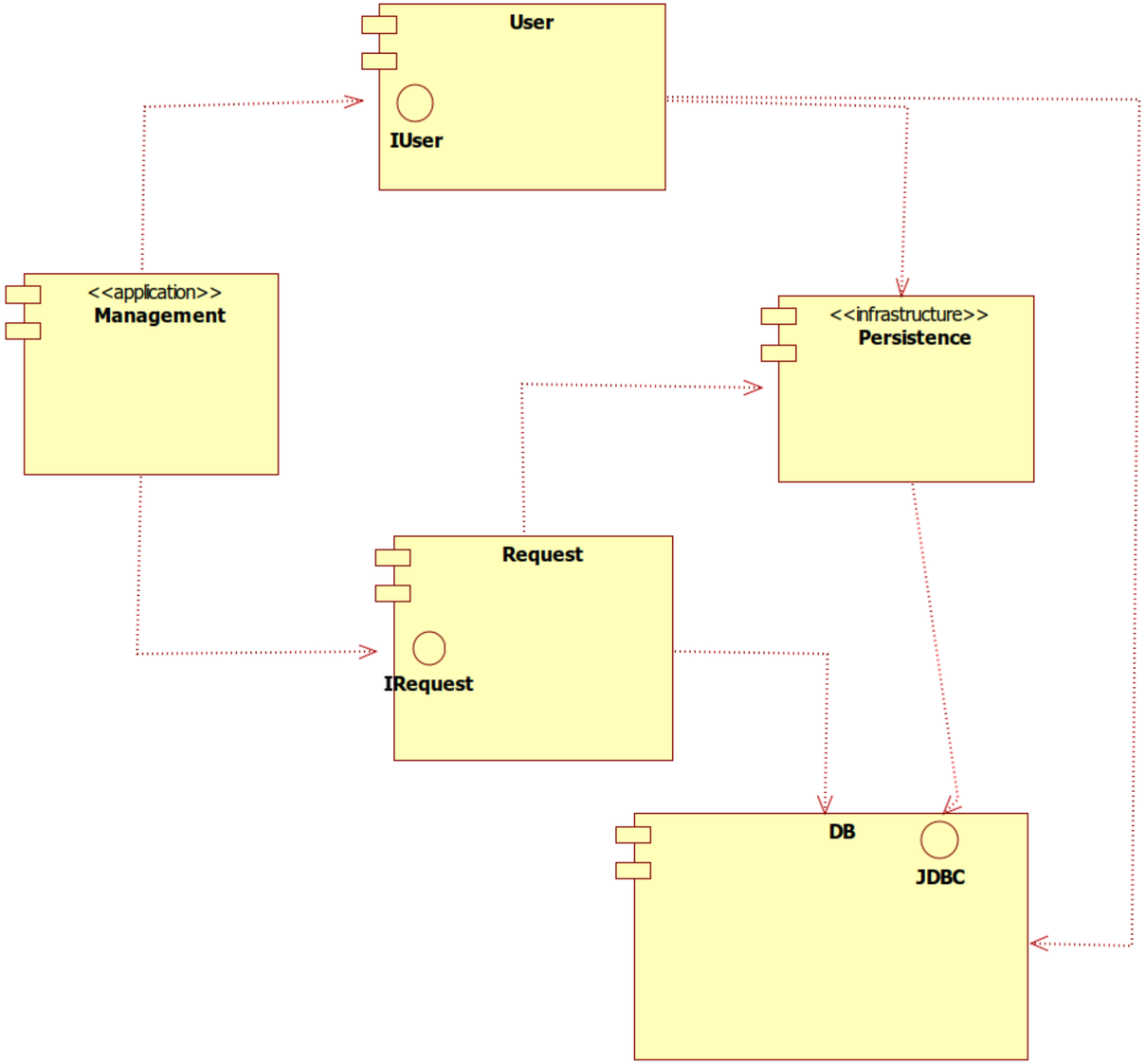
c. Search Profile and Result with login and if data present in the database.

d. The searched data is displayed if available and then Log Out.



COMPONENT DIAGRAM:

The component diagram's main purpose is to show the structural relationships between the components of a system. It is represented by boxed figure. Dependencies are represented by communication association.

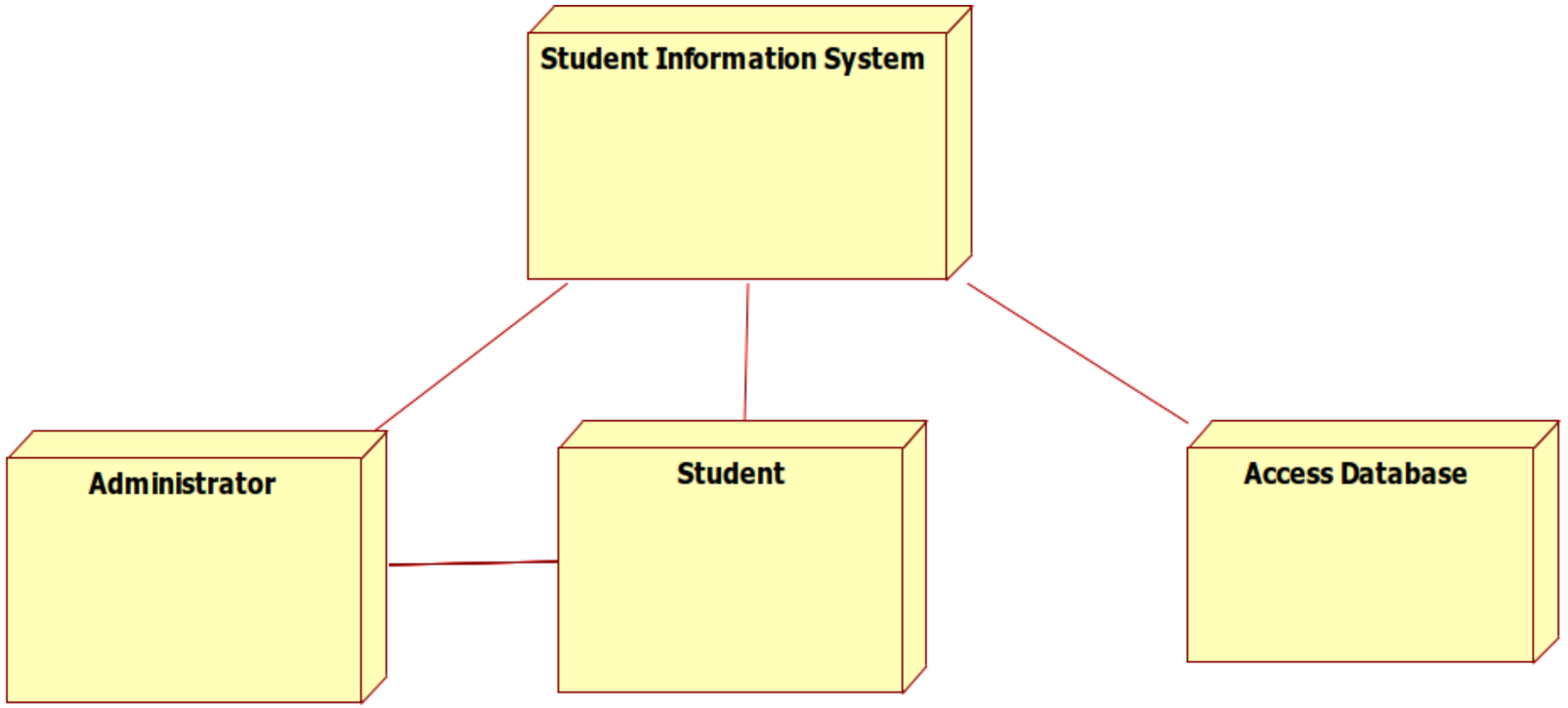


DOCUMENTATION OF COMPONENT DIAGRAM

The main component in this component diagram is Student Information system. And register, User and Manage, Request details are the components comes under the main component.

DEPLOYMENT DIAGRAM:

A deployment diagram in the unified modeling language serves to model the physical deployment of artifacts on deployment targets. Deployment Diagrams show "the allocation of artifacts to nodes according to the Deployments defined between them. It is represented by 3-dimentional box. Dependencies are represented by communication association



DOCUMENTATION OF DEPLOYMENT DIAGRAM

The processor in this deployment diagram is the Student Information System which is the main part and the Student are the Admin, verify and search which are the some of the main activities performed in the system.