# OOMD MINI PROJECT

## LABORATORY PRACTICE - IV

Group Members :

1. Kunal Desai (24134)
2. Mayur Kharmate (23131)
3. Aniket Uttekar (24130)
4. Shreyas Kulkarni (24129)

# UML Diagram for Project Work

## Title :

Mini-Project :

Draw all UML diagrams for your project work .

## Problem Definition :

Draw all UML diagrams like

1. Use Case Diagram
2. Sequence Diagram
3. Activity Diagram
4. Deployment Diagram
5. Component Diagram
6. Class Diagram

for your project work .

## Prerequisite :

Basic Concepts of Object Oriented Modelling and Designing and UML Diagrams, Access to draw.io website , etc.

## Software Requirements :

Web Browser like Google Chrome , Microsoft Edge , Access to internet , etc.

## Hardware Requirement :

2GB RAM, 500 GB HDD, i5 Processor , etc.

## Theory :

### What is UML?

The UML stands for Unified modeling language, is a standardized general-purpose visual modeling language in the field of Software Engineering. It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system. It helps in designing and characterizing, especially those software systems that incorporate the concept of Object orientation. It describes the working of both the software and hardware systems.

The UML was developed in 1994-95 by Grady Booch, Ivar Jacobson, and James

Rumbaugh at the Rational Software. In 1997, it got adopted as a standard by the Object Management Group (OMG).
The Object Management Group (OMG) is an association of several companies that controls the open standard UML. The OMG was established to build an open standard that mainly supports the interoperability of object-oriented systems. It is not restricted within the boundaries, but it can also be utilized for modeling the non-software systems. The OMG is best recognized for the Common Object Request Broker Architecture (CORBA) standards.

## Characteristics of UML :

The UML has the following features:

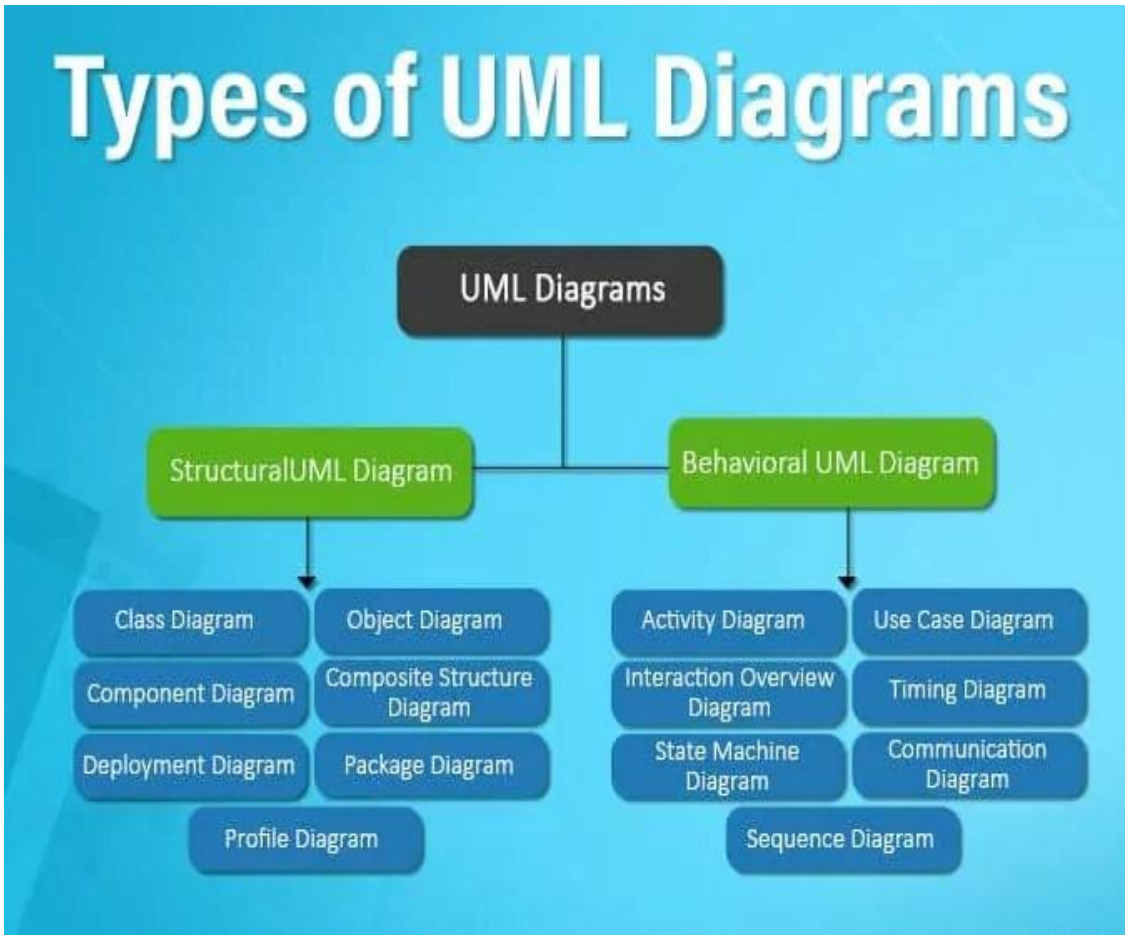
- It is a generalized modeling language.
- It is distinct from other programming languages like C++, Python, etc.
- It is interrelated to object-oriented analysis and design.
- It is used to visualize the workflow of the system.
- It is a pictorial language, used to generate powerful modeling artifacts.

## UML Diagrams :

The diagrams are the graphical implementation of the models that incorporate symbols and text. Each symbol has a different meaning in the context of the UML diagram. There are thirteen different types of UML diagrams that are available in UML 2.0, such that each diagram has its own set of a symbol. And each diagram manifests a different dimension, perspective, and view of the system.

UML diagrams are classified into three categories that are given below:

1. Structural Diagram
2. Behavioral Diagram
3. Interaction Diagram




1. Structural Diagram :

It represents the static view of a system by portraying the structure of a system. It shows several objects residing in the system. Structural diagrams depict a static view or structure of a system. It is widely used in the documentation of software architecture. It embraces class diagrams, composite structure diagrams, component diagrams, deployment diagrams, object diagrams, and package diagrams. It presents an outline for the system. It stresses the elements to be present that are to be modeled. Following are the structural diagrams given below:

- Class diagram :
  Class diagrams are one of the most widely used diagrams. It is the backbone of all the object-oriented software systems. It depicts the static structure of the system. It displays the system's class, attributes, and methods. It is helpful in recognizing the relation between different objects as well as classes.

- Object diagram :
  It describes the static structure of a system at a particular point in time. It can be used to test the accuracy of class diagrams. It represents distinct instances of classes and the relationship between them at a time.

- Package diagram :
  It is used to illustrate how the packages and their elements are organized. It shows the dependencies between distinct packages. It manages UML diagrams by making it easily understandable. It is used for organizing the class and use case diagrams.

- Component diagram :
  It portrays the organization of the physical components within the system. It is used for modeling execution details. It determines whether the desired functional requirements have been considered by the planned development or not, as it depicts the structural relationships between the elements of a software system.

- Deployment diagram :
  It presents the system's software and its hardware by telling what the existing physical components are and what software components are running on them. It produces information about system software. It is incorporated whenever software is used, distributed, or deployed across multiple machines with dissimilar configurations.

## 2. Behavioral Diagram :

It depicts the behavioral features of a system. It deals with dynamic parts of the system. Behavioral diagrams portray a dynamic view of a system or the behavior of a system, which describes the functioning of the system. It includes use case diagrams, state diagrams, and activity diagrams. It defines the interaction within the system. It encompasses the following diagrams:

- Activity diagram :
  It models the flow of control from one activity to the other. With the help of an activity diagram, we can model sequential and concurrent activities. It visually depicts the workflow as well as what causes an event to occur.

- State machine diagram :
  It is a behavioral diagram. it portrays the system's behavior utilizing finite state

transitions. It is also known as the State-charts diagram. It models the dynamic behavior of a class in response to external stimuli.

- Use case diagram :
  It represents the functionality of a system by utilizing actors and use cases. It encapsulates the functional requirement of a system and its association with actors. It portrays the use case view of a system.

3. Interaction diagram :

Interaction diagrams are a subclass of behavioral diagrams that give emphasis to object interactions and also depicts the flow between various use case elements of a system. In simple words, it shows how objects interact with each other and how the data flows within them. It consists of communication, interaction overview, sequence, and timing diagrams. It depicts the interaction between two objects and the data flow between them. Following are the several interaction diagrams in UML:

- Timing diagram :
  It is a special kind of sequence diagram used to depict the object's behavior over a specific period of time. It governs the change in state and object behavior by showing the time and duration constraints.
  Interaction Overview diagram: It is a mixture of activity and sequence diagram that depicts a sequence of actions to simplify the complex interactions into simple interactions.

- Sequence diagram :
  It shows the interactions between the objects in terms of messages exchanged over time. It delineates in what order and how the object functions are in a system.

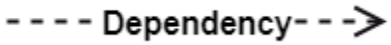
- Communication diagram :
  It shows the interchange of sequence messages between the objects. It focuses on objects and their relations. It describes the static and dynamic behavior of a system.

## UML-Relationship :

Relationships depict a connection between several things, such as structural, behavioral, or grouping things in the unified modeling language. Since it is termed as a link, it demonstrates how things are interrelated to each other at the time of system execution. It constitutes four types of relationships, i.e., dependency, association, generalization, and realization.

### 1. Dependency :

Whenever there is a change in either the structure or the behavior of the class that affects the other class, such a relationship is termed as a dependency. Or, simply, we can say a class contained in other class is known as dependency. It is a unidirectional relationship.




### 2. Association :

Association is a structural relationship that represents how two entities are linked or connected to each other within a system. It can form several types of associations, such as one-to-one, one-to-many, many-to-one, and many-to-many. A ternary association is one that constitutes three links. It portrays the static relationship between the entities of two classes.

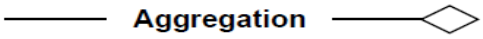
An association can be categorized into four types of associations, i.e., bi-directional, unidirectional, aggregation (composition aggregation), and reflexive, such that an aggregation is a special form of association and composition is a special form of aggregation. The mostly used associations are unidirectional and bi-directional.

It is represented by a line between the classes followed by an arrow that navigates the direction, and when the arrow is on both sides, it is then called a bidirectional association. We can specify the multiplicity of an association by adding the adornments on the line that will denote the association.


### 3. Aggregation :

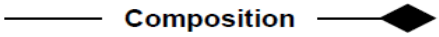
An aggregation is a special form of association. It portrays a part-of

relationship. It forms a binary relationship, which means it cannot include more than two classes. It is also known as Has-a relationship. It specifies the direction of an object contained in another object. In aggregation, a child can exist independent of the parent.



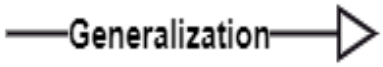

## 4. Composition :

In a composition relationship, the child depends on the parent. It forms a two-way relationship. It is a special case of aggregation. It is known as Part-of relationship.




## 5. Generalization :

The generalization relationship implements the object-oriented concept called inheritance or is-a relationship. It exists between two objects (things or entities), such that one entity is a parent (superclass or base class), and the other one is a child (subclass or derived class). These are represented in terms of inheritance. Any child can access, update, or inherit the functionality, structure, and behavior of the parent.




## 6. Realization :

It is a kind of relationship in which one thing specifies the behavior or a responsibility to be carried out, and the other thing carries out that behavior. It can be represented on a class diagram or component diagrams. The realization relationship is constituted between interfaces, classes, packages, and components to link a client element to the supplier element.

## Advantages :

- **Helps in communication :**

UML provides a standard language that communicates design information among software developers. This saves time and money, as it eliminates the requirement to train developers in a new, proprietary software language. It also facilitates communication with other teams or individuals who review the design.

- **Saves time :**

UML can help developers save time by automating some of the design processes. In addition, using this language can help developers avoid potential errors in their programs, which can save time in the development and testing processes. UML can also save time in the coding process because it allows developers to reuse code.

- **Enhances collaboration :**

UML allows different software developers to work on the same project by providing a common language. This enhances collaboration and provides for a more efficient design process. It also helps identify potential problems early in the design process.

- **Provides a better understanding of a system :**

UML allows developers to see the big picture of a system. This can help them understand the system better and identify potential challenges. It also allows them to see how different system parts interact with each other.

- **Unifies design :**

UML provides a standard way to design software and systems. This allows for a more unified design process and helps to ensure compliance with standards. It also offers easier design reviews and enhanced productivity.

## Disadvantages :

- Too Much Emphasis on Design :

UML places much emphasis on design, which can be problematic for some developers and companies. Looking at a software scope in a UML diagram can lead to software project stakeholders over-analyzing problems, as well as cause

people to lose focus by spending too much time and attention on software features. Companies cannot solve every problem with a software tool using a UML diagram -- eventually, they just have to start coding and testing. Brody Gooch, a co-creator of UML, said that the original vision for UML was a "graphical language to help reason about the design of a system as it unfolds." If people get hung up using a diagram to identify and solve issues, it can delay the actual work that needs to be done to fix the issues.

- Formal Notation is Not Necessary :

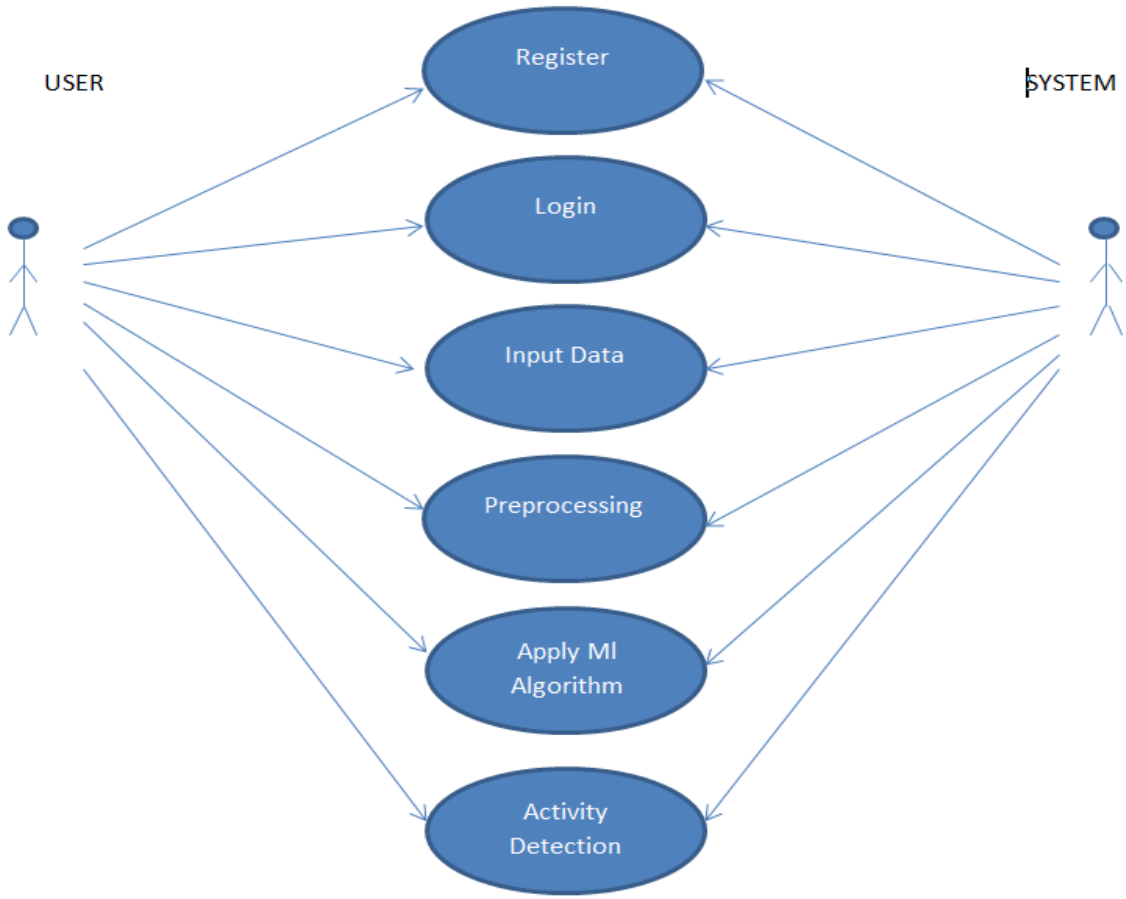
The strongest argument against UML is that you don't really need a UML diagram to communicate your designs. You can have the same impact and effect with informal, box-and-line diagrams created in PowerPoint, Visio, or a whiteboard. As coding is a formal language by itself, a lot of developers don't prefer the complexity and the formality at the architectural level, which discourages the use of UML and has become one of its disadvantages.

- Ascending Degree of Complexity :

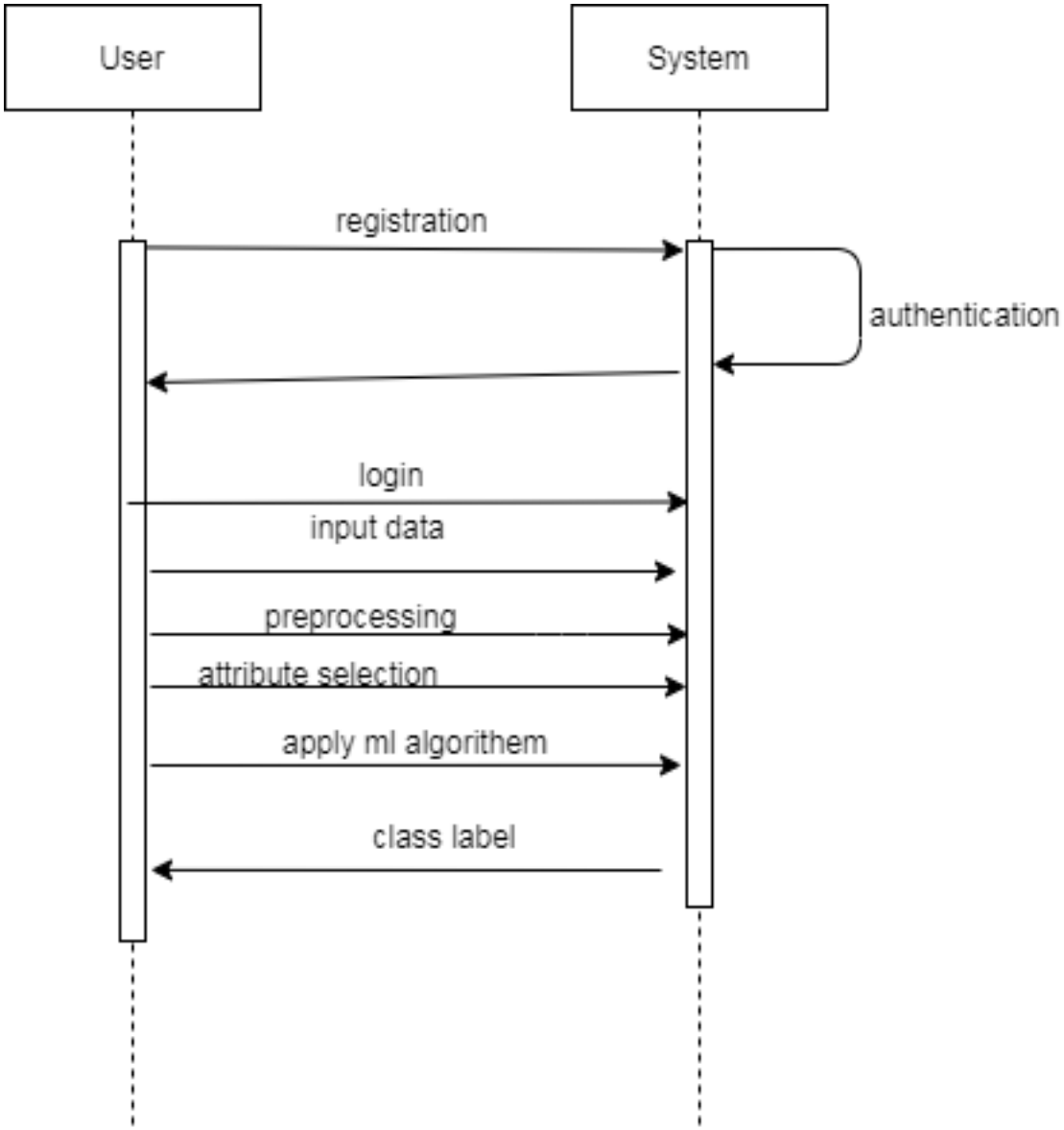
Since its initiation until now, UML has grown in complexity and size. The sheer size of UML makes a lot of people nervous right at the onset, and they feel like they won't be able to learn it, and are better off without it.

# UML Diagrams For Suspicious Activity Detection in Hospital

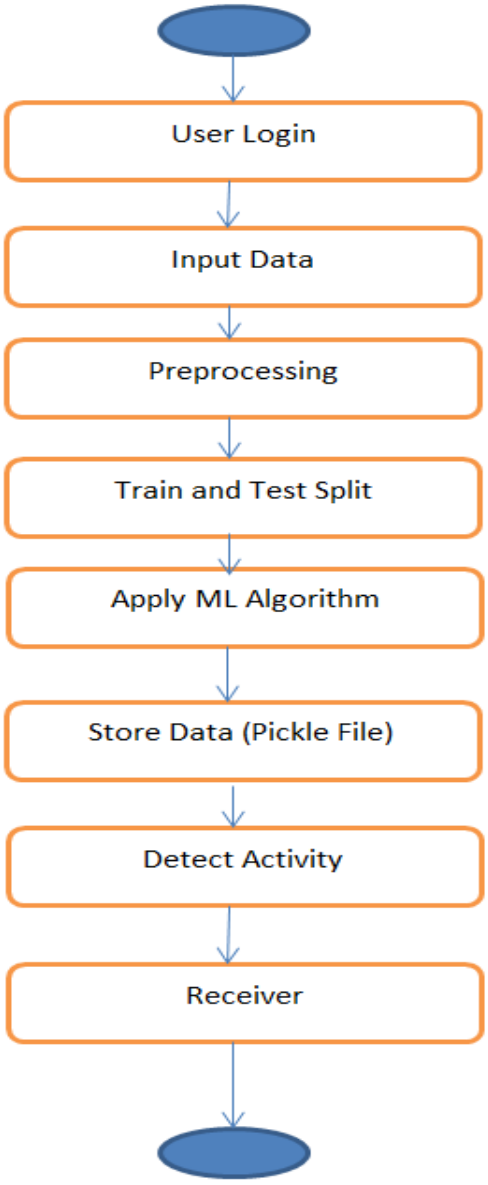
## Use Case Diagram :



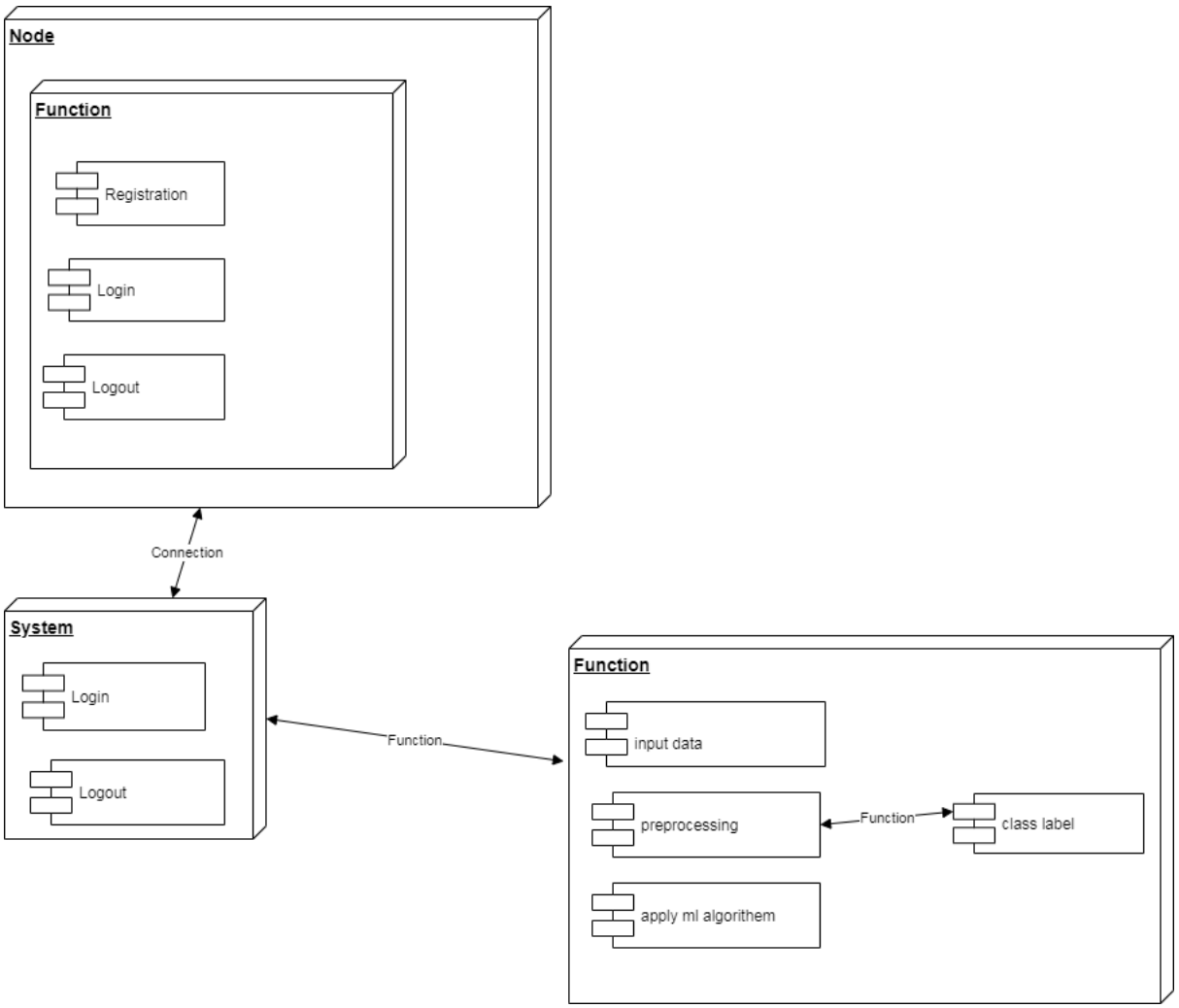
## Sequence Diagram :



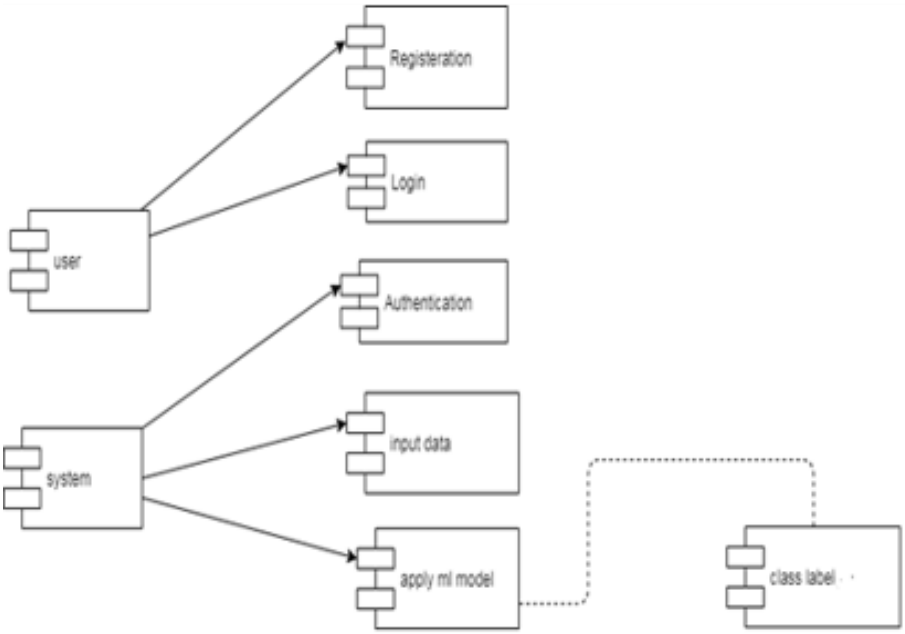
## Activity Diagram :



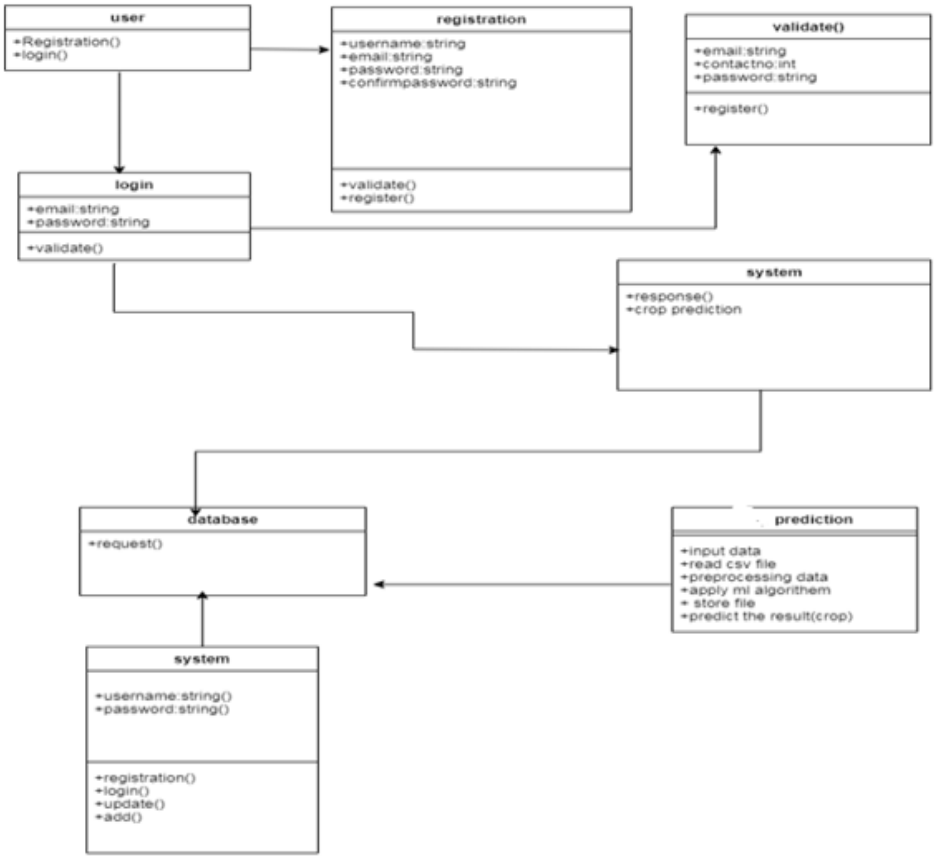
## Deployment Diagram :



## Component Diagram :



## Class Diagram :




## Conclusion :

Thus , We drew the UML diagrams for the project Suspicious Activity Detection in Hospital .