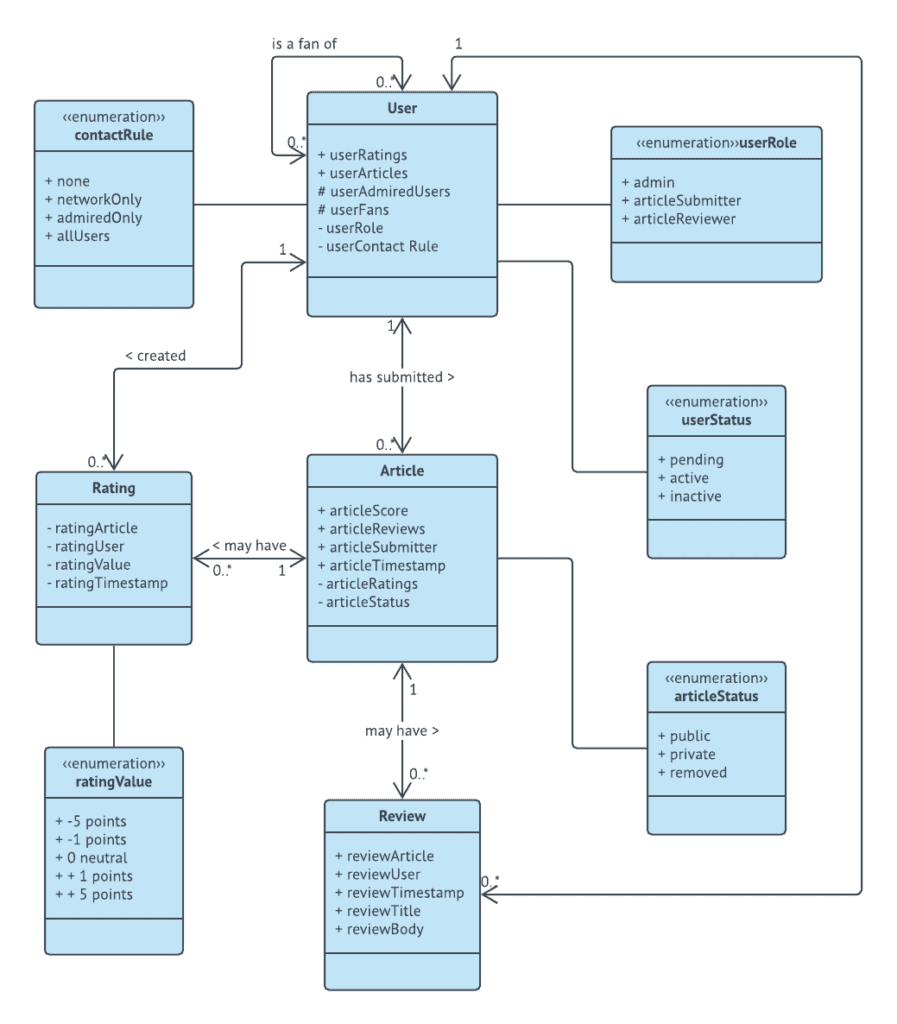
**Unit 6 Application of UML to Secure Software Design**

UML stands for Unified Modeling Language, and it is a modeling language that is most often used for software engineering but has extended its use to business processes and other project workflows. Essentially, UML is visualizing software through diagrams, specifically one of the thirteen UML diagrams. This modeling language was created by three software engineers at the company Rational Software for their projects, and it has become the standard with very few updates over the years.

A UML diagram is typically used in software engineering and other business processes where modeling is useful. There are two main ways UML diagrams are used as a part of these processes:

* **Forward design.** The modeling and design are all done before coding the software application. Usually, forward design is used to help developers better see the system they are trying to create.
* **Backward design.** The modeling is done after the code has been written, and the UML diagrams act as documentation for the workflow of the project. This can help developers see the project development as it was, in reality, to improve in the future.

Whether UML diagrams are used before or after the coding or project, they provide a way to visualize many aspects of a project and who is responsible for what activity.



**Types of UML Diagrams**

The original UML had nine types of diagram, and the second version raised that to thirteen types of diagram. Each diagram can be divided into two categories: structural diagrams and behavioral diagrams. A structural diagram will break down the project by the process or way the project is set up. A behavioral diagram will break down the project by how the system behaves and how its components and actors are involved. Here are each of the types of UML diagrams explained.

**Structural Diagrams**

* **Class diagram.**This is the most common type of documentation diagram for software. The diagram has three fields where the first one is the class, then its attributes right beneath it, and finally the behaviors are in the third field. Then different types of lines connect each class to show the relationships between the classes.
  + **Object diagram.**This diagram is, in a way, a subset of the class diagrams. Object diagrams display a particular instance of a class. For example, if a class diagram had the class of “Customers,” the object diagram might have “Customer 1” to show the attributes and behaviors of that particular customer. These diagrams allow developers to see if the system they created with a class diagram is viable when put into practice.
  + **Composite structure diagram.**As another further subset of class diagrams, composite structure diagrams show the internal structure of a class. Each class is broken down into its components, and the connecting lines are drawn between the components to show how each relates and communicates. It’s an extremely specific diagram, so it’s used less often, especially in a business environment.
* **Package diagram.**This type of diagram helps show the big picture by organizing elements of the system into groups called packages. The packages are then sorted into related groups, and those groups are connected to show relationships or interaction. Package diagrams help show complex systems and complex behaviors that might be hard to visualize in detail.
* **Component diagram.**It can be difficult to visualize complex systems, but that is what a component diagram is designed to do. The system is broken down into its components, which communicate with each other through interfaces. The interfaces are connected to show the path of component communication. These diagrams provide a bird’s eye overview of a complex system or process.
* **Deployment diagram.**This diagram shows the relationship between software and the physical hardware it’s deployed through. Deployment diagrams are a physical model of how each software component (called artifacts) is deployed through its hardware node and the connections between all of the software and hardware elements. This diagram is typically only useful for software development because of its focus on hardware.

**Behavioral Diagrams**

* **Activity diagram.**Activity diagrams are some of the most useful diagrams, especially for business processes. These diagrams show the workflow of a project or operational process, and it can show the relationship between different activities. The flow between activities can be both parallel and sequential. The diagram will usually display the materials used, objects produced by activities, and the relationship between each.
  + **Interaction** **overview diagram.**While activity diagrams show the sequence of processes, interaction overview diagrams show the sequence of object interactions and in the correct order. Arrows are used to show the flow between the interactions. These are some of the most complex UML diagrams, and it uses the same annotations as an activity diagram.
* **Sequence diagram.**As the same suggests, this diagram shows the sequence of interactions in a system chronologically during a particular process. All communication between actors or objects is shown with a variety of connecting lines, and actors don’t need to be active until another actor communicates with it. Visually these diagrams are structured vertically with chronological, top-down organization. These diagrams are important for the development of any kind of application, not just software.
  + **Communication diagram.**These diagrams are similar to sequence diagrams, but the main focus here is on the communication between objects, not all types of interaction. Communication diagrams follow a numbering system to show chronology and arrows to show the flow of messages. On the diagram, the objects don’t have to be physically close to each other to communicate; they only need to be included in the numbering and have an arrow connecting them to another object.
  + **Timing diagram.**If the focus of a project is the timing of events, this diagram is the best to use. Like a sequence diagram, this shows the sequence of events, but instead of connecting by a relationship, each event is shown to flow along a linear axis of time, usually from left to right. The main focus is the duration of time, and the diagram often shows this with a staircase and arrows displaying the time constraint.
* **Use case diagram.**These diagrams give an overview of all of the actors, functions each actor needs to perform, and how the functions interact within a system. The goal is to visualize the functions of each actor involved and how all of the actors and functions are related to each other. Usually, actors are connected to their functions with arrows, and related functions are connected to each other with different arrows.
* **State diagram.**Also known as state chart diagrams, these show the state of a system when it’s acted on by an external or internal stimulus. The diagram visualizes the many states of a system and how it could change if certain external or internal events take place and all of the actions in between.