Course Code: CSL601	Course Name:
Class :	Batch :
Roll no :	Name :

Experiment : 05

Aim : Identify scenarios & develop UML Use case and Class Diagram for the project.

Theory :

What is UML diagram and why it used?
UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques.
Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficacy.

What is usecase digram?
In the Unified Modeling Language (UML), a use case diagram can summarize the details of our system's users (also known as actors) and their interactions with the system. To build one, we'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of our system

Explain the components of usecase diagram?

- Actors: The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data.
- System: A specific sequence of actions and interactions between actors and the system. A system may also be referred to as a scenario.
- Goals: The end result of most use cases. A successful diagram should describe the activities and variants used to reach the goal.

What is class diagram?
In software engineering, 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, operations (or methods), and the relationships among objects.
Purpose of class diagram are:

- Shows static structure of classifiers in a system
- Diagram provides basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from business perspective

Explain the components of class diagram?

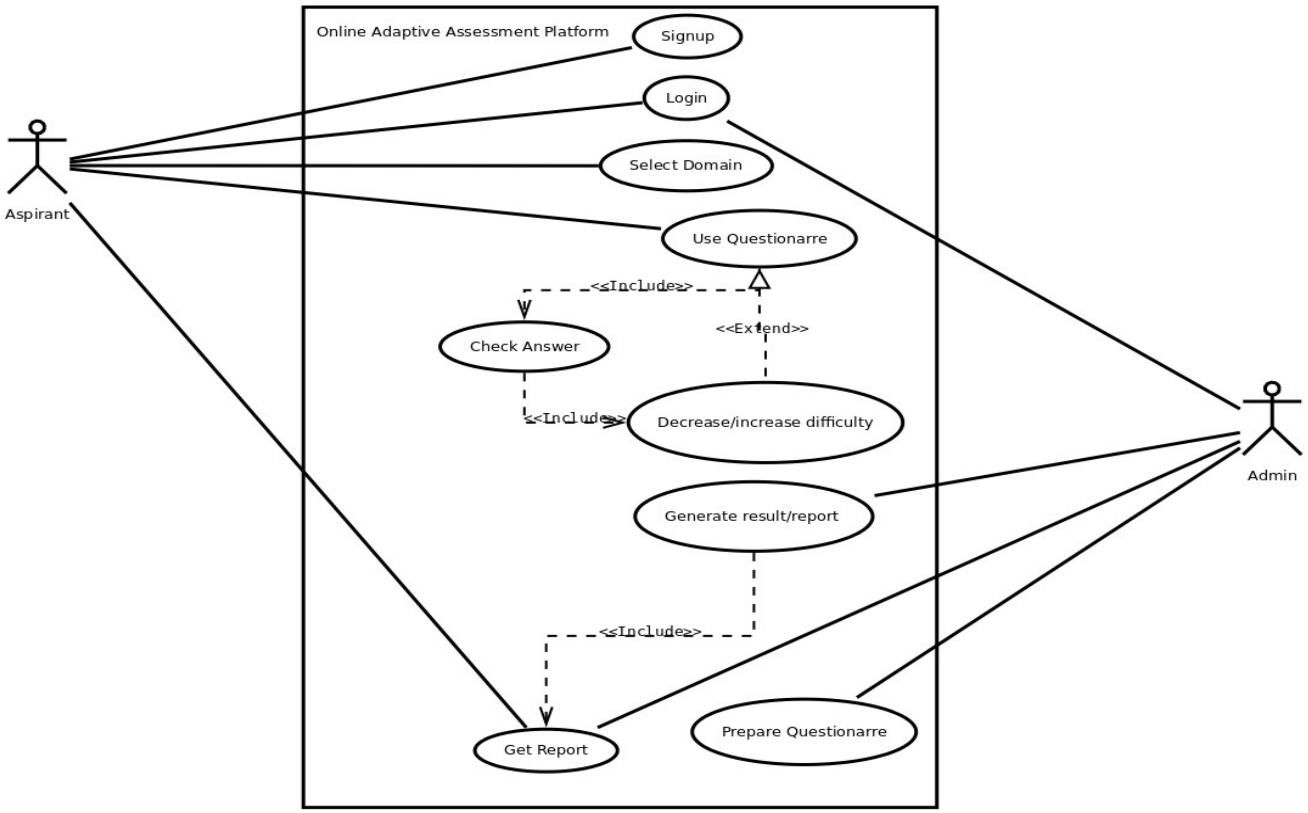
The standard class diagram is composed of three sections:

- Upper section: Contains the name of the class. This section is always required, whether you are talking about the classifier or an object.
- Middle section: Contains the attributes of the class. Use this section to describe the qualities of the class. This is only required when describing a specific instance of a class
- Bottom section: Includes class operations (methods). Displayed in list format, each operation takes up its own line. The operations describe how a class interacts with data.
- Member access modifiers - All classes have different access levels depending on the access modifier (visibility). Here are the access levels with their corresponding symbols:
  Public (+)
  Private (-)
  Protected (#)
  Package (~)
  Derived (/)
  Static (underlined)
- Member scopes There are two scopes for members: classifiers and instances.
  Classifiers are static members while instances are the specific instances of the class. If you are familiar with basic OO theory, this isn't anything groundbreaking.
- Classes: A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations. When you draw a class in a class diagram, you're only required to fill out the top row—the others are optional if you'd like to provide more detail.
  - ➔ Name: The first row in a class shape.
  - ➔ Attributes: The second row in a class shape. Each attribute of the class is displayed on a separate line.
  - ➔ Methods: The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.
- Signals: Symbols that represent one-way, asynchronous communications between active objects.
- Data types: Classifiers that define data values. Data types can model both primitive types and enumerations.
- Packages: Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
- Interfaces: A collection of operation signatures and/or attribute definitions that define a cohesive set of behaviors. Interfaces are similar to classes, except that a class can have an instance of its type, and an interface must have at least one class to implement it.
- Enumerations: Representations of user-defined data types. An enumeration includes groups of identifiers that represent values of the enumeration.
- Objects: Instances of a class or classes. Objects can be added to a class diagram to represent either concrete or prototypical instances.
- Artifacts: Model elements that represent the concrete entities in a software system, such as documents, databases, executable files, software components, etc.

Output :

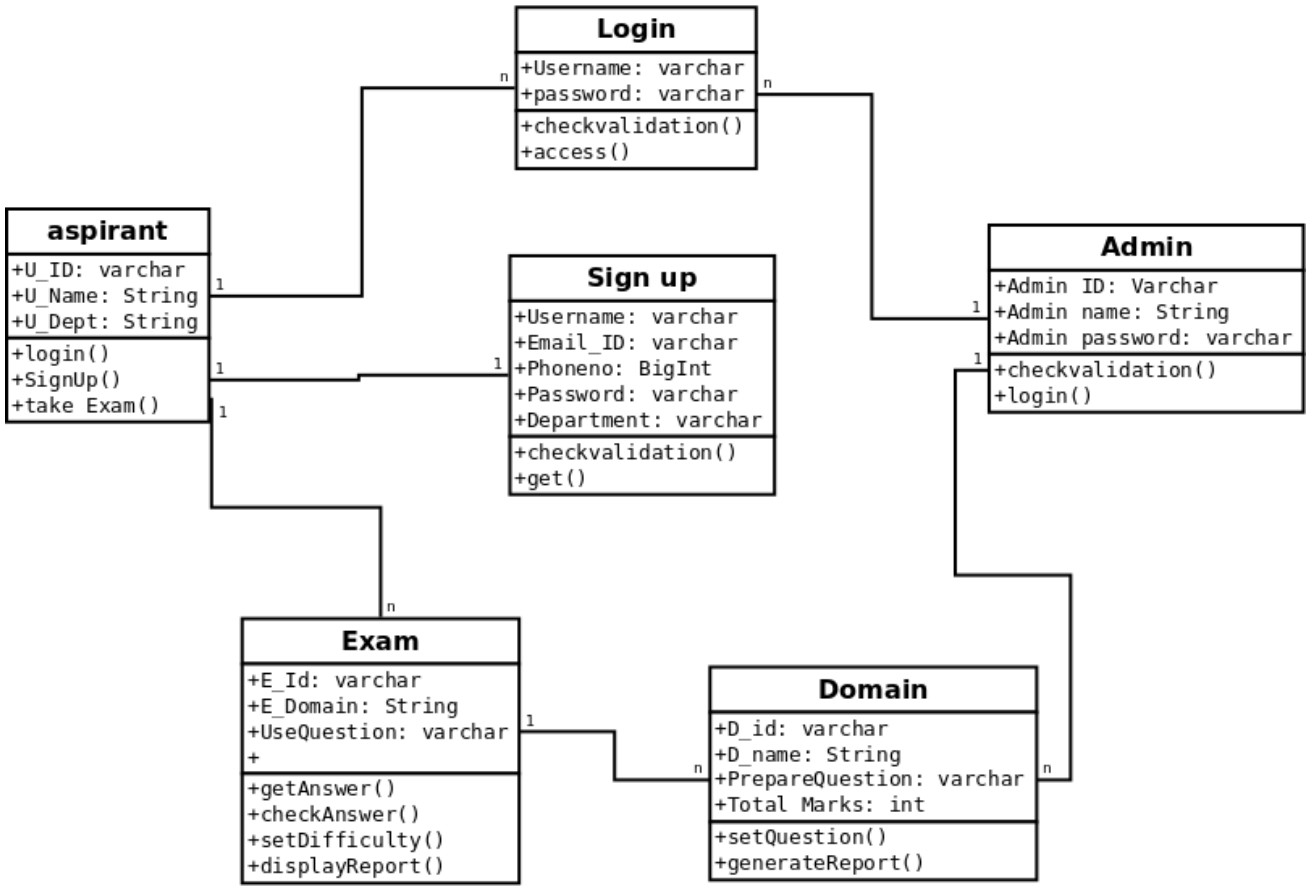
Use Case Diagram

Use Case Diagram for Online Adaptive Assessment Platform (OOAP)



Class Diagram

Class Based Diagram For Online Adaptive Assessment Platform



Reference:

https://www.uml-diagrams.org/use-case-diagrams.html
https://www.uml-diagrams.org/class-diagrams-overview.html
https://www.draw.io/
https://creately.com/

Conclusion:
