Experiment-1.

Name: Vishesh Garg

Roll no: 12113066

Section: IT-A-04

Software Development Lab

Ques-1. What is UML and why it is required?

UML stands for Unified Modeling Language. It is a standardized visual modeling language used in software engineering to represent, design, and communicate various aspects of a system or software application. UML provides a set of graphical notations that allow developers, analysts, and stakeholders to visualize and document different components, relationships, behaviors, and interactions within a system.

UML is required because it offers a common and standardized way to visually communicate complex software systems, regardless of the specific programming language or technology being used. It enhances communication and understanding among team members, helps in capturing requirements, designing architectures, and documenting software projects. UML diagrams such as class diagrams, use case diagrams, sequence diagrams, and more, provide a clear and consistent way to express ideas and concepts, facilitating better collaboration and more effective software development.

Ques-2. What type of diagrams is available in UML, with their functionality?

1. Class Diagram: This diagram shows the different classes in a software system and how they are connected. It's like a blueprint that defines the structure of the program, including the attributes (variables) and methods (functions) each class has.

2. Use Case Diagram: This diagram is like a storybook. It displays the interactions between users (actors) and the system, showcasing the various tasks or actions the system can perform to meet the users' needs.

3. Sequence Diagram: Imagine it's like recording the conversation between different parts of a program. This diagram illustrates how various components of the system interact and communicate with each other over time, displaying the order of messages exchanged.

4. Activity Diagram: Think of it as a flowchart for the software. This diagram displays the workflow and logical steps of a process or a feature, showing how different activities connect and decisions are made.

5. State Machine Diagram: This diagram is like watching how a program reacts to different situations. It shows the different states a system or object can be in and how transitions between states occur based on events or conditions.

6. Component Diagram: Imagine it's like arranging puzzle pieces. This diagram illustrates the physical components of a system and how they are connected or grouped together to form a working software system.

7. Deployment Diagram: Think of it as placing things on a map. This diagram displays how the software components are deployed on hardware, such as servers, computers, or devices, showing the physical arrangement of the system.

8. Communication Diagram: It's like drawing lines to show how people talk to each other. This diagram visualizes the interactions between objects or components, emphasizing the connections and messages passed among them.

9. Package Diagram: Imagine it's like organizing files in folders. This diagram shows how classes and other components are organized into packages or groups, helping in managing and understanding the system's structure.

10. Object Diagram: It's like taking a snapshot of objects at a certain moment. This diagram captures a specific instance of the system, showing the objects and their relationships as they exist during a particular scenario.

Ques-3. Difference between OOPs and procedural oriented programming languages?

Object-Oriented Programming (OOP) and Procedural Programming are two distinct approaches to writing computer programs. Procedural Programming centers around creating functions that manipulate data, treating code as a sequence of steps. In contrast, Object-Oriented Programming revolves around objects, which combine data and the functions that operate on it into cohesive units. This encapsulation enables clearer data management and promotes code modularity.

Procedural Programming tends to separate data and functions, resulting in a less organized structure. On the other hand, OOP integrates data and behavior within objects, fostering better encapsulation and data hiding. Inheritance, a key OOP concept, facilitates the creation of new classes by reusing attributes and methods from existing ones, while polymorphism enables objects of different classes to be treated interchangeably through shared interfaces. Moreover, Procedural Programming lacks built-in support for these advanced features, it may be suitable for simpler tasks. However, Object-Oriented Programming excels in managing complex projects by offering better code organization, reusability, and adaptability. Its ability to model real-world scenarios more naturally makes it well-suited for projects that require a close alignment with real-world entities and interactions.

Ques-4. Different platform for UML diagrams?

Various platforms for UML diagrams are:

Lucidchart, Visual Paradigm, Draw.io, Gliffy, Astah, PlantUML, Creately, Enterprise Architect, yEd, SmartDraw