Task 6: Design pattern and architecture:

Design Patterns: Design patterns are reusable solutions to common software design problems. They are not specific implementations but rather general guidelines and templates that can be applied to various situations. Design patterns provide proven solutions to recurring issues in software development, and they ensure that the resulting code is efficient, flexible, and easy to understand.

Design patterns should be used when you encounter specific design problems or challenges in your software development process. They provide proven and reusable solutions to common design issues, making your code more maintainable, flexible, and efficient.

Software Architecture: Software architecture is the high-level structure and organization of a software system. It defines the key components of the system, their relationships, and how they interact with each other to achieve the desired functionality and performance. Architecture provides the blueprint for the software solution and is the foundation on which the system is built.

Software architecture should be considered and designed early in the software development process, typically during the initial stages of a project. It provides a high-level blueprint for the entire system and lays the foundation for the software solution.

Both design patterns and software architecture are essential for creating robust and maintainable software systems. They complement each other and help developers build software that is easy to understand, extend, and maintain over time.

Scope: Design patterns address specific design issues within the system, while software architecture deals with the overall structure and organization of the entire system.

Level of Abstraction: Design patterns are low-level, fine-grained guidelines for solving specific design problems, whereas software architecture is a high-level, coarse-grained blueprint for the entire system.

Focus: Design patterns focus on improving the design and implementation of individual components or interactions, while software architecture focuses on defining the relationships and interactions between all components and how they fit together to form a coherent system.