Design pattern vs Architecture pattern

What is Design pattern and Architecture pattern?

Design Patterns:

Design patterns are specific solutions to recurring design problems in software development. They focus on the design of individual classes, objects, and their interactions. These patterns provide templates or guidelines for solving common coding challenges and improving code quality, reusability, and maintainability.

examples Singleton pattern, Factory Method pattern.

Architecture Patterns:

Architecture patterns, on the other hand, deal with the higher-level structure and organization of entire software systems. They guide the overall layout, components, and interactions of the system. These patterns address concerns such as scalability, maintainability, separation of concerns, and system performance.

examples Model-View-Controller (MVC) pattern, Microservices architecture.

What is the differences?

Scope:

Design patterns focus on solving specific coding and design issues within individual classes or objects.

Architecture patterns focus on the high-level structure and organization of entire systems.

Granularity:

Design patterns provide solutions at a finer granularity, addressing class-level design concerns.

Architecture patterns provide solutions at a coarser granularity, addressing system-level concerns.

Purpose:

Design patterns aim to improve code quality, reusability, and maintainability within smaller components.

Architecture patterns aim to create a scalable, maintainable, and well-organized system as a whole.

Level of Abstraction:

Design patterns deal with the specifics of coding and object interactions.

Architecture patterns deal with the overall system structure, components, and their interactions.