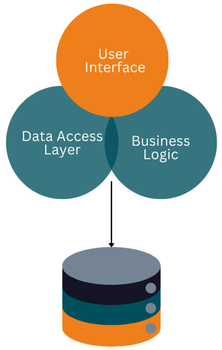
Monolithic vs Microservices

Application Architectures

1. Introduction
   1. It defines
      1. How the app is built
      2. How different modules talk to each other
      3. How data flows and operations are performed
   2. Structure & design of how different parts of an app (UI, logic, database) are organized and connected
   3. Defines how these components interact to perform tasks efficiently
   4. ☑ Types of Application Architectures
      1. Monolithic Architecture (Tightly Coupled)
      2. Microservices Architecture (Loosely Coupled)

2. Monolithic Architecture (Tightly Coupled) V/S Microservices Architecture (Loosely Coupled)

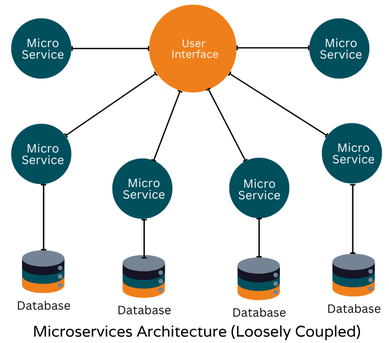
1



Monolithic Architecture (Tightly Coupled)

1	Structure	Single unified codebase and deployment unit
2	Coupling	Tightly coupled — components depend heavily on each other
3	Deployment	One deployment for the entire app
4	Scalability	Difficult to scale parts independently
5	Testing	Requires end-to-end testing
6	Updates	Updating one module may affect the whole system
7	Failure Impact	Failure in one part can crash the whole application
8	Technology Stack	Usually single tech stack
9	Database	One shared database
10	DevOps & CI/CD	Complex for large teams
11	Example	Traditional ERP systems, early web apps
12	Hosting Environment	Usually hosted on Virtual Machines

2



Microservices Architecture (Loosely Coupled)

1	Structure	Multiple small services with independent codebases
2	Coupling	Loosely coupled — services are independent
3	Deployment	Each service is deployed independently
4	Scalability	Easy to scale services individually
5	Testing	Individual service testing is possible
6	Updates	Update one service without impacting others
7	Failure Impact	Failures are isolated to individual services
8	Technology Stack	Different services can use different stacks/languages
9	Database	Each service can have its own database
10	DevOps & CI/CD	Enables faster development and CI/CD pipelines
11	Example	Netflix, Amazon, Uber, Spotify
12	Hosting Environment	Typically hosted in Containers using ECS, EKS, or Kubernetes