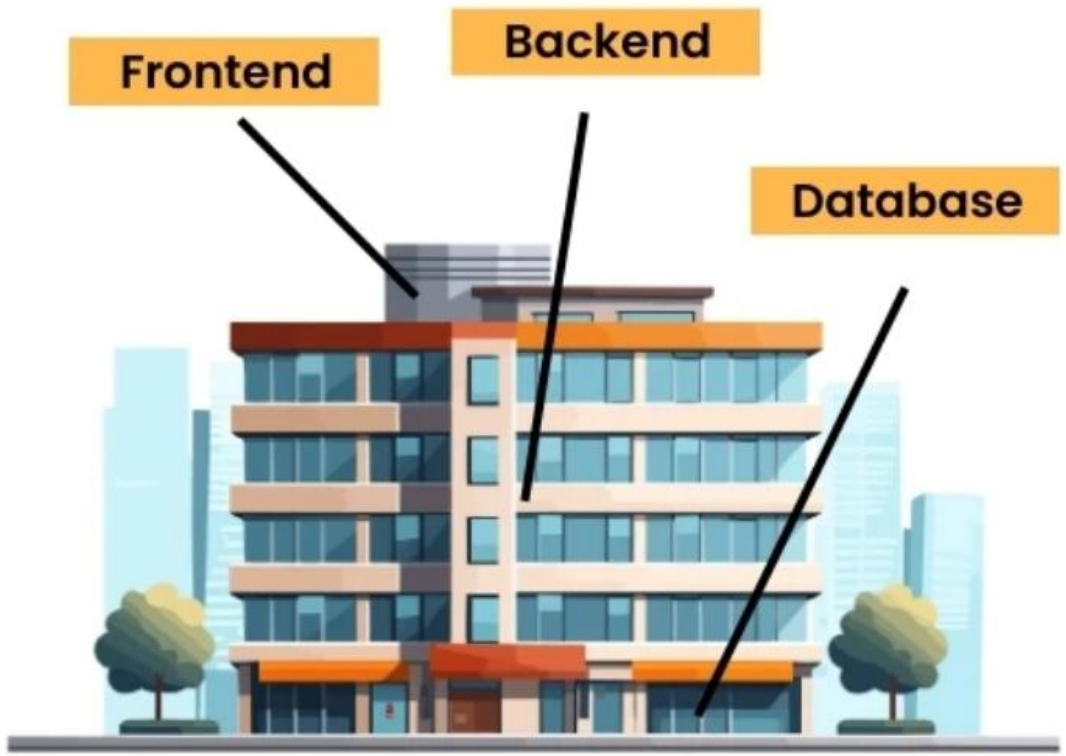

Microservices

What are Microservices

Monolithic vs Microservices



Customer Support

User Interface

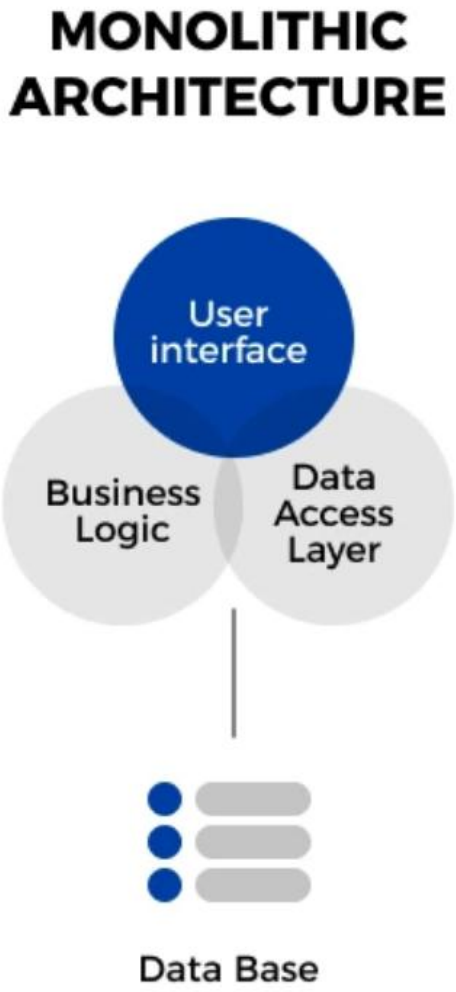
Platform

Frameworks

Monolithic architecture

Monolithic architecture is a traditional software development approach where an application is built as a single, unified unit.

All components, such as the user interface, business logic, and data access layers, are tightly coupled and run as a single process.



Example of Monolithic Architecture

Imagine an **e-commerce application** with the following modules:

- **User Management** (Login, Registration)
- **Product Catalog** (Display, Search)
- **Order Processing** (Cart, Checkout, Payments)
- **Inventory Management** (Stock Updates)

In a monolithic structure, all these modules reside in a **single project**, making deployment and maintenance straightforward at the beginning. However, as the application grows, managing dependencies and updates becomes complex.

Characteristics of Monolithic Architecture

- **Single Codebase** – The entire application is managed in a single repository.
- **Tightly Coupled Components** – Modules (UI, business logic, and database access) are interdependent.
- **Single Deployment Unit** – The whole application is deployed together, even for minor changes.
- **Shared Database** – Usually, a single database is used for all application functions.
- **Simple Development & Debugging** – Easy to develop and debug due to a unified structure.
- **Scalability Challenges** – Cannot scale individual components efficiently; the entire application must be scaled.

Advantages of Monolithic Architecture

✔ **Easy Development & Deployment**

– Ideal for small projects, as everything is in one place.

✔ **Performance**

– Faster communication between components since they run in the same memory space.

✔ **Simplified Testing**

– Easier to test since all components are integrated.

Disadvantages of Monolithic Architecture

- **Hard to Scale**
  – Scaling requires deploying the entire application rather than specific components.

- **Difficult to Maintain**
  – As the codebase grows, modifying or debugging becomes complex.

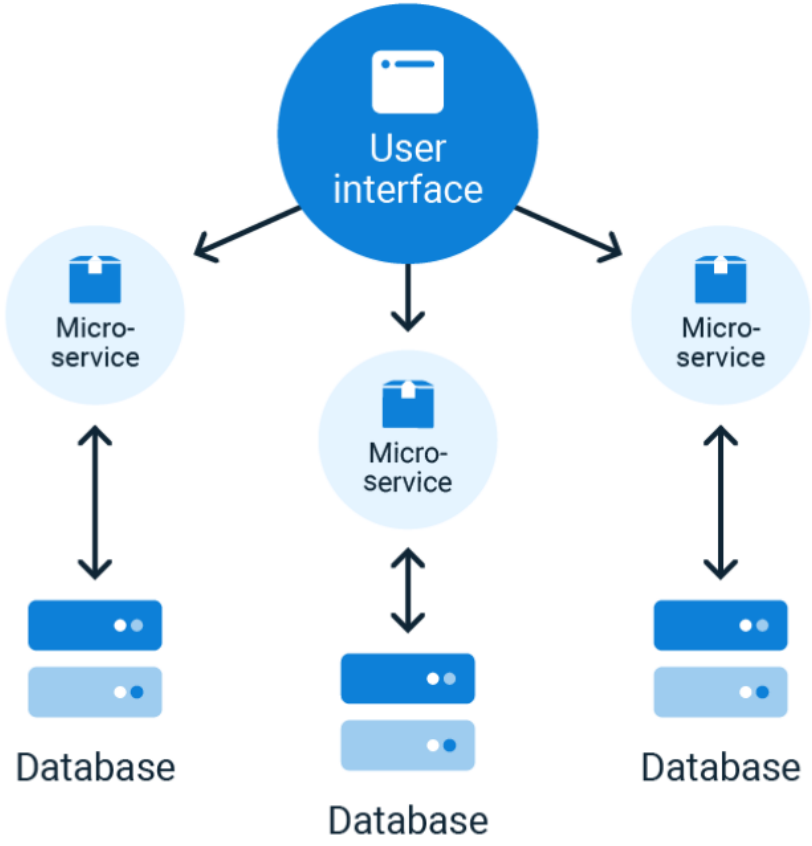
- **Technology Lock-in**
  – Hard to introduce new technologies without affecting the entire system.

- **Longer Deployment Times**
  – Even small changes require redeploying the entire application.

Microservices Architecture

Microservices architecture is a software development approach where an application is built as a collection of small, independent services.

Each service is responsible for a specific functionality and communicates with other services through APIs.



Example of Microservices Architecture

Consider an **e-commerce application** broken into the following services:

- **User Service** – Handles authentication, user profiles, and accounts.
- **Product Service** – Manages product catalog, descriptions, and pricing.
- **Order Service** – Processes orders, payments, and invoices.
- **Inventory Service** – Tracks stock availability.
- **Notification Service** – Sends emails, SMS, and app notifications.

Each service has **its own database** and communicates via APIs, ensuring flexibility and fault isolation.

Characteristics of Microservices Architecture

- **Decoupled Services** – Each service operates independently with its own business logic and database.
- **Independent Deployment** – Services can be updated, deployed, and scaled individually.
- **Technology Diversity** – Each service can be developed using different programming languages or databases.
- **Lightweight Communication** – Services communicate through REST APIs, gRPC, or messaging queues.
- **Scalability** – Services can be scaled individually based on demand.

Advantages of Microservices Architecture

✔ **Scalability** – Services can be scaled independently based on usage.
✔ **Faster Deployment** – Teams can deploy changes to a single service without affecting others.
✔ **Technology Flexibility** – Different services can use different technologies (e.g., Java for User Service, Python for AI-based Recommendation Service).
✔ **Resilience** – If one service fails, the entire application does not go down.
✔ **Improved Development Speed** – Different teams can work on separate services in parallel.

Disadvantages of Microservices Architecture

- **Increased Complexity** – Managing multiple services, databases, and APIs is more complex.
- **Higher Operational Overhead** – Requires infrastructure for API management, logging, and monitoring.
- **Latency in Communication** – API calls between services add network latency compared to monolithic applications.
- **Data Management Complexity** – Each service has its own database, leading to challenges in data consistency.

Monolithic vs. Microservices

	Monolithic Architecture	Microservices Architecture
Structure	Single, unified app	Multiple independent services
Deployment	Entire app redeployed	Services deployed independently
Scalability	Hard to scale	Easy to scale individual components
Maintenance	Becomes complex over time	Easier with smaller codebases
Technology	Single tech stack	Allows multiple technologies
Fault Isolation	Failure affects entire app	Failure is limited to the affected service

Monolithic Architecture

- **When to Use?**
  - **For Small & Medium Applications**
  - **When Quick Development is Needed**
  - **For Startups & MVPs** – Easier to build, test, and launch without worrying about distributed systems.

- **When to Avoid?**
  - **For Large-Scale Applications** – Harder to scale and maintain.
  - **If You Need Frequent Updates**.
  - **For High Availability & Resilience** – A single failure can bring down the entire system.

Microservices Architecture?

- **When to Use?**
  - **For Large-Scale Applications**
  - **When You Need Independent Scaling**
  - **For Faster Deployments & CI/CD –** When frequent updates and releases are required.
  - **For Teams Working in Parallel** – When multiple teams work on different parts of the system.

- **When to Avoid?**
  - **For Small Applications**
  - **If You Lack DevOps Expertise** – Requires strong DevOps skills
  - **If Latency is a Concern** – Network calls between services can introduce delays.

Thank you