Monolithic Architecture/Centralised system: Monolithic architecture is a software development approach where an application is developed as a single, self-contained unit. In this approach, all the modules of an application are tightly coupled and run as a single process, which means that any changes to one module can affect the entire system.

In a monolithic architecture, all the components of an application, including the user interface, business logic, and data access layers, are integrated into a single codebase. This approach makes it easier to develop and test the application as a whole, but it can also make it harder to scale and maintain as the application grows.

With the increasing complexity of modern applications and the need for scalability and flexibility, many developers are now adopting more modern architecture patterns such as microservices and serverless architectures.

Pros:

1. Simplicity: Monolithic architecture is relatively easy to develop, test, and deploy because all the modules are in one codebase.
2. Performance: A monolithic architecture can offer better performance than other architectures because all the components of the application are running in the same process.
3. Easier maintenance
4. Easier debugging

Cons:

1. Scalability: It can be challenging to scale a monolithic application as it grows because all the components of the application must scale together.
2. Flexibility: Monolithic applications can be less flexible than other architectures because all the components are tightly coupled.
3. Longer development cycles: Developing a monolithic application can take longer because all the components must be developed and tested together.
4. High risk: A monolithic application can be high risk because any change to one module can affect the entire system.

Overall, monolithic architecture can be a good choice for smaller applications with simpler requirements. However, for larger and more complex applications, other architectures such as microservices and serverless architectures may be a better fit.

Link

Microservices Architecture: Microservices architecture is an approach to software development where an application is built as a collection of small, independent services that can be developed, deployed, and scaled separately. Each service runs as a separate process and communicates with other services through APIs.

In a microservices architecture, each service focuses on a specific business function, such as user authentication, product catalog management, or order processing. These services are developed, deployed, and managed independently of each other, allowing for greater flexibility and scalability. If a specific service needs more resources to handle increased demand, it can be scaled up without affecting the other services.

The benefits of a microservices architecture include:

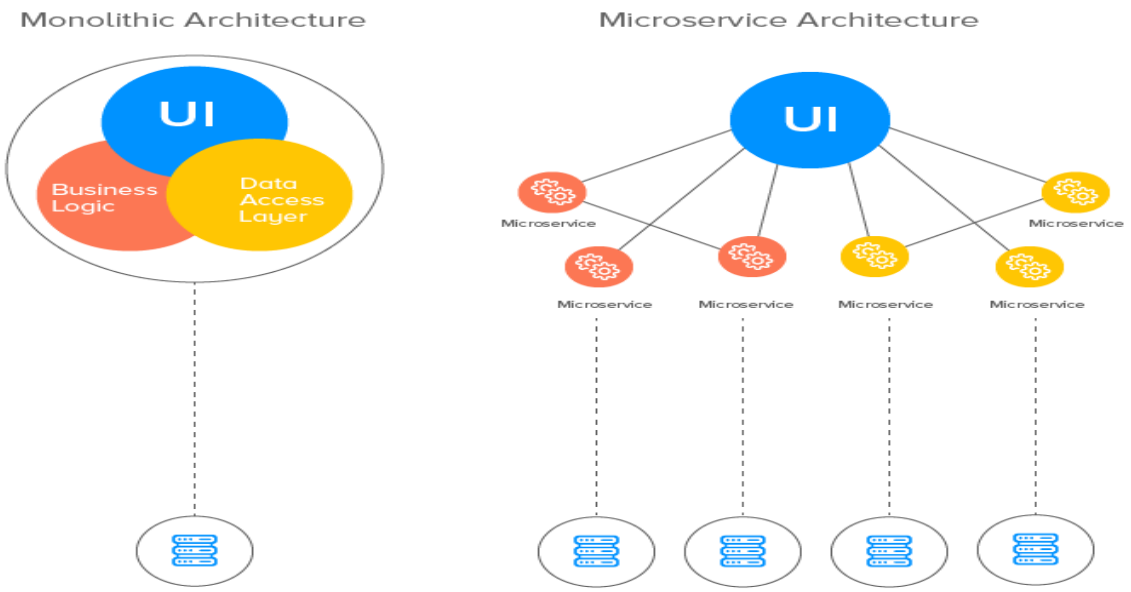
1. Scalability: Since each service can be scaled independently.
2. Flexibility: Microservices architecture allows developers to choose the most appropriate technology stack for each service
3. Resilience: If one service fails, it does not necessarily affect the other services.
4. Easy deployment: Because each service is independent, it can be deployed without affecting the rest of the system.
5. Agility: Microservices architecture enables developers to release new features and updates more quickly, as they can be developed and deployed independently.

Cons:

1. Complexity: Implementing a microservices architecture can be more complex as it requires careful management of the APIs and communication between services.
2. Distributed system issues: As microservices architecture is a distributed system, it can be more susceptible to issues such as network latency and failures.
3. Testing: Testing a microservices architecture can be more challenging, as each service needs to be tested independently.
4. Operational overhead: Microservices architecture requires more operational overhead, as each service needs to be monitored and managed independently.
5. Integration challenges: Integrating data and services across multiple services can be more challenging in a microservices architecture than in a monolithic architecture.

However, implementing a microservices architecture can be more complex than a monolithic architecture, as it requires careful management of the APIs and communication between services. Additionally, because each service is independent, it can be more difficult to ensure consistency across the entire application.

Video Sources:
Bytebytego: Link
Arpit Bhayani: Link
Gaurav Sen: Link Link
Nana Janashia: Link
Yogita Sharma: Link Link
Anshul P Tiwari: Link



Source: appinventiv.com

Service Oriented Architecture: Service-oriented architecture (SOA) is a method of software development that uses software components called services to create business applications. Each service provides a business capability, and services can also communicate with each other across platforms and languages. Developers use SOA to reuse services in different systems or combine several independent services to perform complex tasks.

For example, multiple business processes in an organization require the user authentication functionality. Instead of rewriting the authentication code for all business processes, you can create a single authentication service and reuse it for all applications. Similarly, almost all systems

across a healthcare organization, such as patient management systems and electronic health record (EHR) systems, need to register patients. These systems can call a single, common service to perform the patient registration task.

Link

Serverless Architecture: Done in a separate page

Service Mesh:

Twelve Factor Apps: