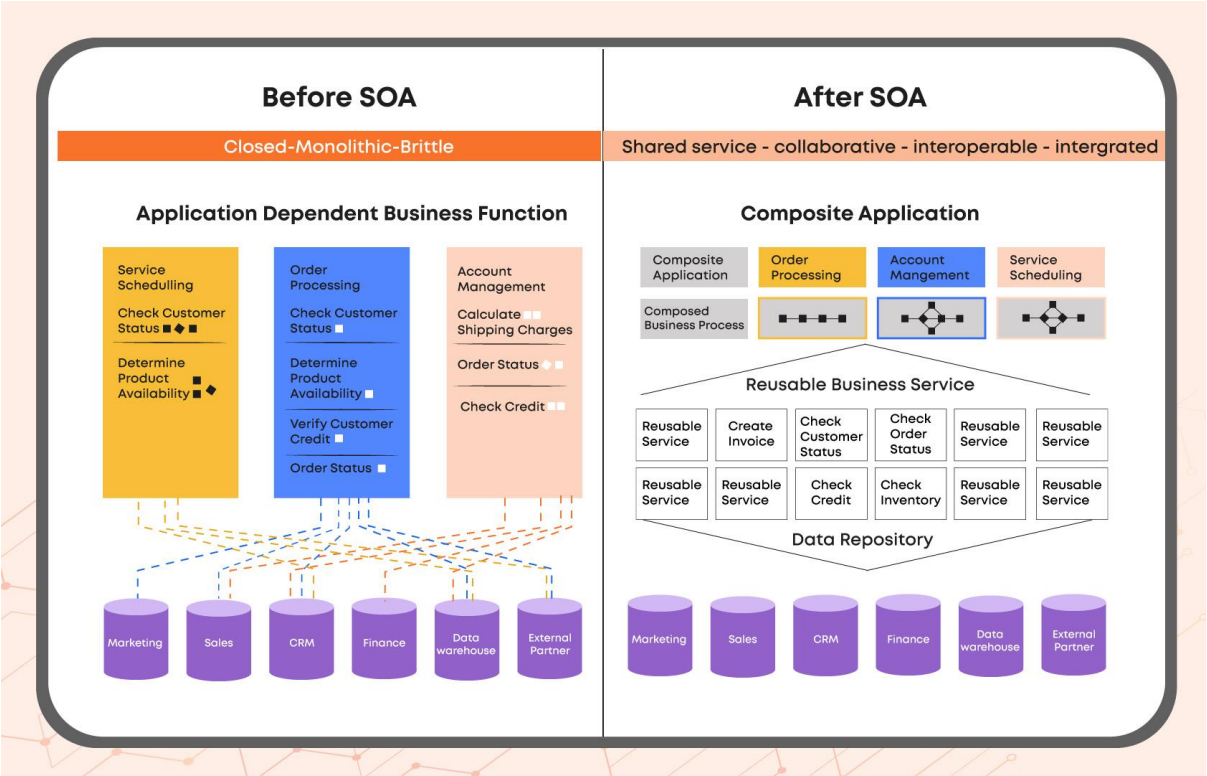
Service-Oriented Architecture

Definition:

Service-Oriented Architecture(SOA) is a style of architecture that promotes loose coupling and granular applications to make the components of the software reusable.



Example: In a ticket booking application, the main components would be

- the Booking service
- Authentication service
- Inventory management service
- Payment and confirmation notification service.

While booking a ticket, the booking service will make a request to the authentication service and therefore the booking service would act like a service requester and the authentication service as the service provider. Similarly, the booking service will make a request to the inventor service to check the availability of seats etc. All the services are separate components, separate code bases and are separately deployable. Therefore, this is an example of SOA architecture where different components are separate but communicate with each other to deliver common services.

Advantages of Service-Oriented Architecture:

1. Agile: All the components of Service-oriented architecture are separate so individual development on the components is feasible. Therefore, the time to go to production is less.
2. Selective scaling: Since the services are separated, the individual components scaling or selective scaling is possible.
3. Different tech stacks allowed: Different tech stacks and technologies can be used for the development of different components since all the services are separate components, separate code bases and are separately deployable.
4. Big size teams: Separate teams can be formed for different components of the same project without any conflict of management.
5. Loose Coupling: Leads to increased reusability of the software components.
6. Easier Debugging: Separation of concerns provides modularity and easier debuggability of individual applications
7. If properly designed, it can also lead to a contained blast radius in case a particular part of the system goes down

Disadvantages of Service-Oriented Architecture:

1. Higher Latency: Service-oriented architecture based applications require an additional network call between the service requester and the service provider leading to network delays/lags and therefore increasing latency.
2. Difficult testing: Testing is difficult because of moving boundaries since the software components are dynamic and changing.
3. Security: Security is complex because the software is highly granulated and loosely coupled so security must be ensured for each individual service.
4. Confused Developers: If the developers do not have knowledge of this architecture style this may lead to confusion leading to lower productivity.
5. Cascading Failures: Can lead to cascading failures in case of improperly planned interaction between multiple applications
6. Complex Understanding: This can lead to a difficult understanding of the whole system by a single person if all services developed in isolation