Monolith vs Microservice

**Monolith architecture:**

* The technical/ developer team is small and cannot afford to break down the architecture into microservices.
* The service is simple to think of as a whole.
* The service requires very high efficiency, where network calls are avoided as much as possible.

**Disadvantages:**

* All developers must have context of all services.
* Deployment is complex since any change to the service requires the entire system to be deployed every time.
* Single point of failure.

**Microservice: Client interacts with the gateway and gateways in turn pass the requests to dedicated microservices. Each microservice can have their own database servers which serve data for that microservice.**

* Microservices are easier to reason about/design for a complicated system. They are easier to scale.
* They allow new members to train for shorter periods and have less context before touching a system.
* Parallel development is easy since each service has lesser dependency with each other.
* Deployments are fluid and continuous for each service.
* They allow decoupling service logic on the basis of business responsibility.
* They are more available, as a single service having a bug does not bring down the entire system. This is called a single point of failure.
* Individual services can be written in different languages.
* New services can be tested easily and individually. The testing structure is close to unit testing compared to a monolith.

**Disadvantages:**

* A service that does not need to be broken could have been broken into different parts. A good indicator for a microservice that shouldn’t be a microservice is when a service communicates with only one other service. The RPC network call could have been converted into a normal function call by including both in same service.