What is System Design ?

Lets take an example of a pizza shop and check how can we scale it. Shop is having a single chef at initial.

- **Vertical Scaling**

How do we Handle More Orders ?
Optimise processes and increase throughput using a single resource.

- **PreProcessing**

We can do pre orders preparations.We can cook the bases at our free time or in the time when the load is less.

- **Resilience** [Recovering from failure quickly]

In case the single chef is sick, for that scenario, we have to have a backup chef. Here we keep backups and keep out single point of failure. Example :- Master Slave Architecture.

- **Horizontal Scaling**

Hiring more chefs/resources.

- **MicroServices Architecture**

As now we have more resources, we can expand. Example :- We have 3 chefs, 2 have specialities in preparing pizza and third can prepare Garlic breads. Now we can route the pizza orders to chef 1 and 2 & Garlic bread orders to Chef 3. Now we can increase the chefs that are specialist in Pizza and in Breads. Thus Pizza team is non dependent on another team and can be scalable irrespective of another team.

- **Distributed Systems**

If there is a power shortage at the shop / the license goes off ,at this point our business will shut down. To overcome this we can distribute all our resources to different place which can take more time or have less chefs but it helps in not stoping our business.
Here we can also distribute our load to different places.

- **Load Balancing**

A central system that takes the decision that the incoming request will be sent to S1 or S2 totally depending on the load and less delivery timing.

- **Decoupling**

Separating out the systems that they are not directly dependent on the different systems that provide the services.

- **Logging and Metrics**

In case of any breakdown or any activity we need to log it to our books for carrying out metrics. Logs can be used for Analytics, Auditing, Reporting and Machine Learning purposes.

- **Extensibility**

We need to decouple each system and make them robust. We doesn't want to re-write our code to add new features every time. Ex : Amazon using its delivery as a packages delivery , package can have anything, delivery guys doesn't want to know about it.