# System requirements

## functional

define system **behavior**
what a system is supposed to do

example:
the system must allow applications to exchange messages

## non-functional

define **qualities** of a system
how a system is supposed to be

example:
scalable, highly available, fast

# System requirements

# System requirements

scale writes

scale reads

hash-based partitioning

bounded queue

append-only log

embedded database

B-tree database

LSM-tree database

push vs pull

long polling

leader-based replication

leaderless replication

Hmmm… Let me think…
Here are some fundamental concepts
that help design scalable, highly
available, and fast systems.

software
engineer

leader election

coordination service

strong consistency

load shedding

rate limiting

shuffle sharding

reverse proxy

service discovery

request routing

batching

compression

network protocols

# System requirements
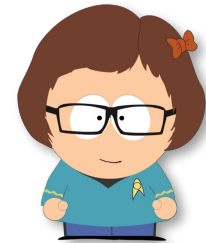
# System requirements



software
engineer

- Define functional requirements and scope.

- Define non-functional requirements.

- Order non-functional requirements according to their importance.

- Use this list as a tiebreaker when considering multiple design options and trade-offs.