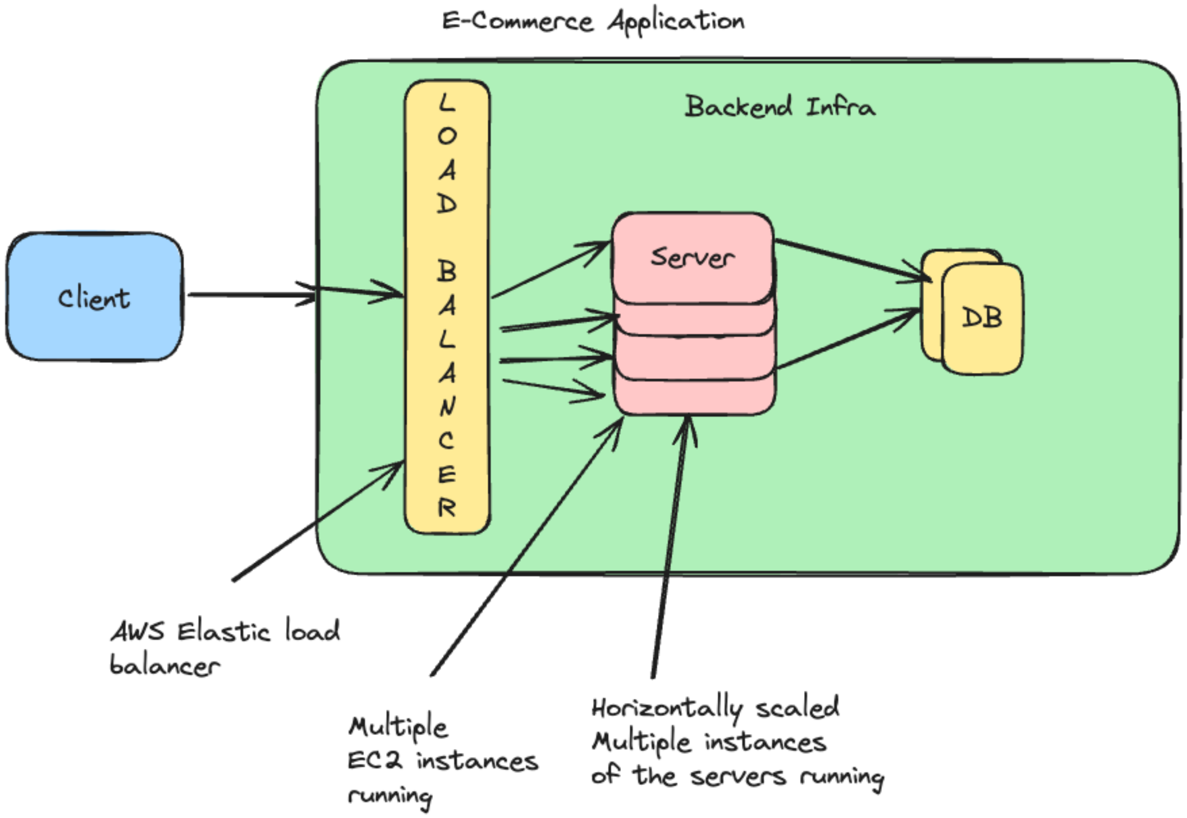
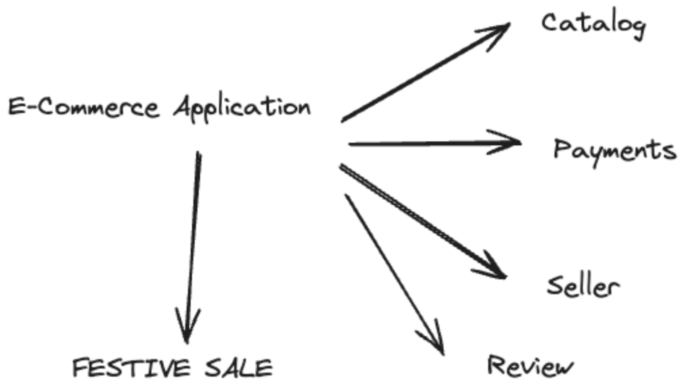
Architectures For Backend


E-Commerce Application
Backend Infra
Client
LOAD BALANCER
Server
DB
AWS Elastic load balancer
Multiple EC2 instances running
Horizontally scaled Multiple instances of the servers running



5 machines → Scale UP → 25 machines

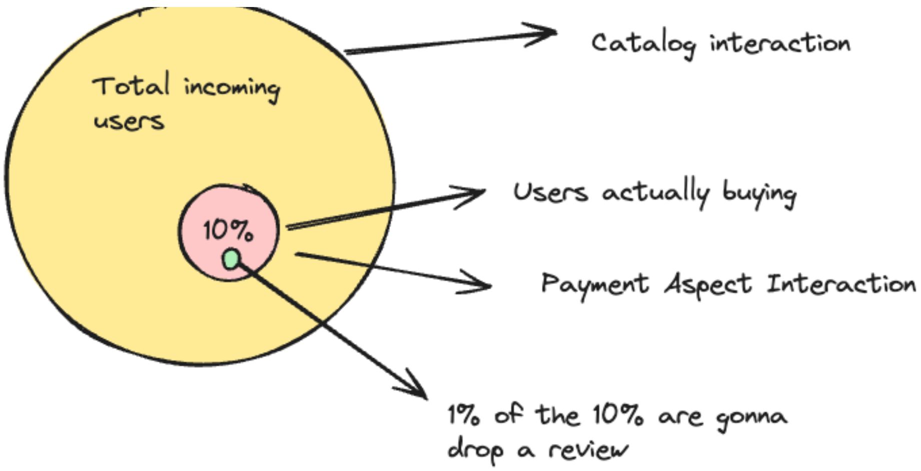
Normal Days → Sale Days

1 million users
a day

10 million users

90% are window shoppers

90% are window shoppers



Total incoming users

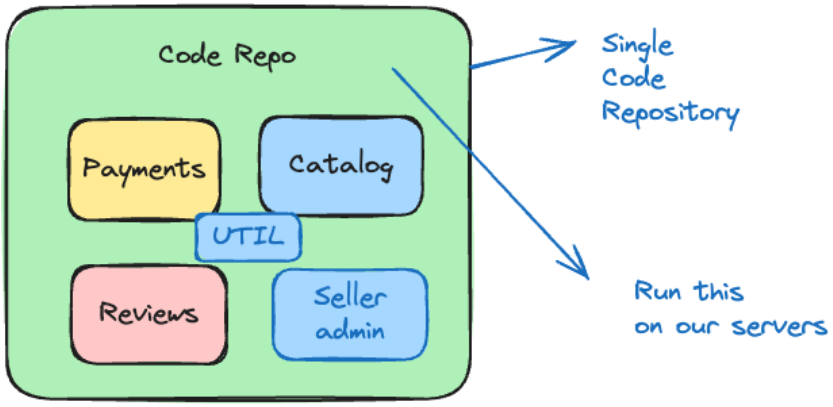
10%

Catalog interaction

Users actually buying

Payment Aspect Interaction

1% of the 10% are gonna drop a review



Code Repo

Payments

Catalog

UTIL

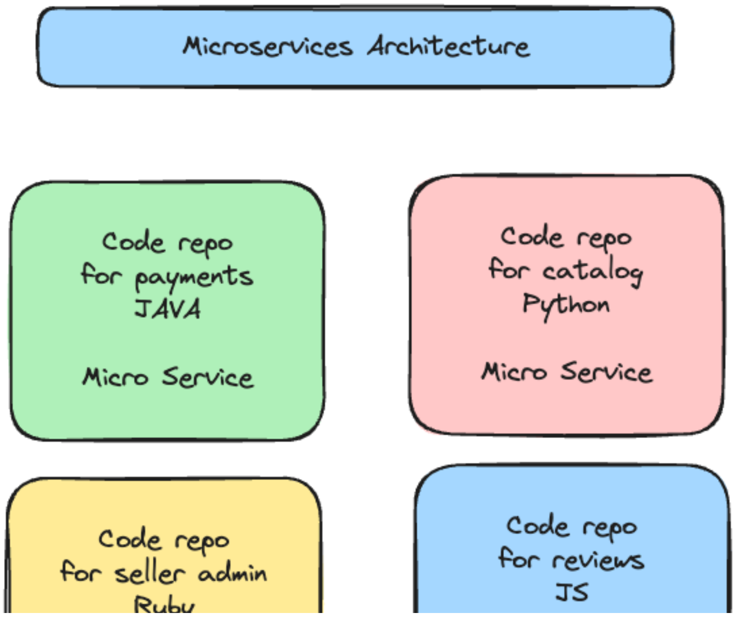
Reviews

Seller admin

Single Code Repository

Run this on our servers

MONOLITH Architecture



Microservices Architecture

Code repo for payments JAVA

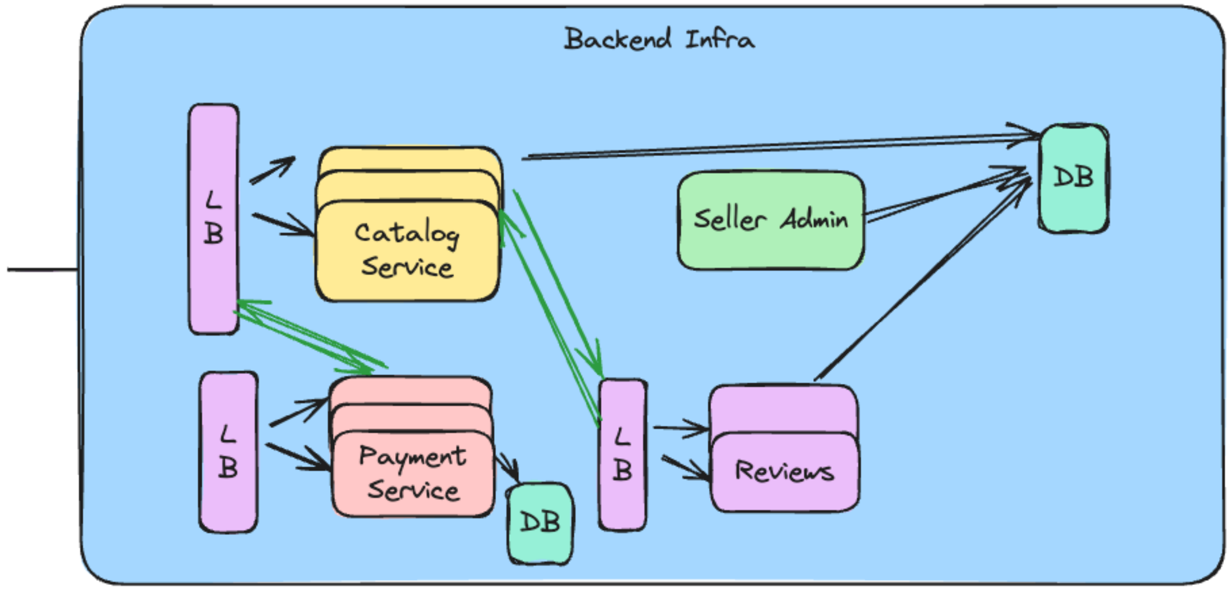
Micro Service

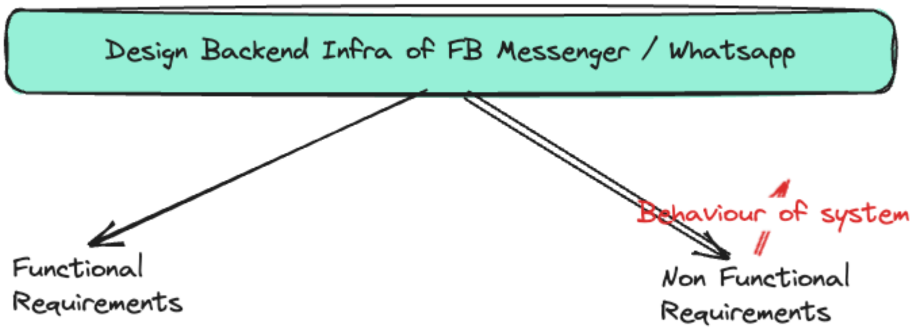
Code repo for catalog Python

Micro Service

Code repo for seller admin Ruby

Code repo for reviews JS





Design Backend Infra of FB Messenger / Whatsapp

Behaviour of system

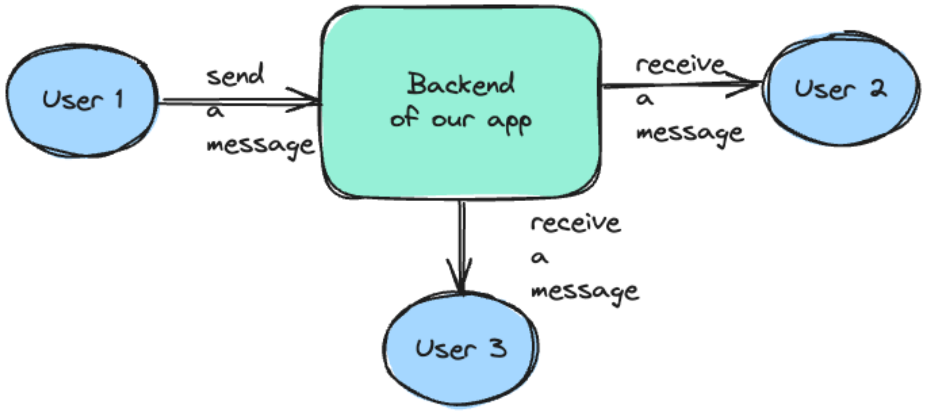
Functional Requirements

Non Functional Requirements

- Application should support 1:1 and group chats
- Applications should support mobile and web client both
- No video call support
- Only text message support
- Not support E2E
- Do we need to store chat history ? Yes
- One message can be of maximum 10^5 chars
- Limit on group size also (configurable)
- Online / Offline tag
- If a user is offline and later comes back online the messages should be then delivered

- Application should be scalable
- No loss messages - i.e. highly consistent

Simple Architecture (Abstracted)

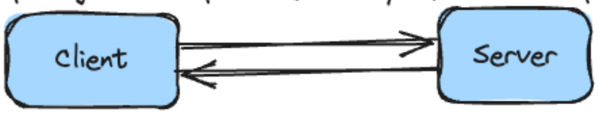


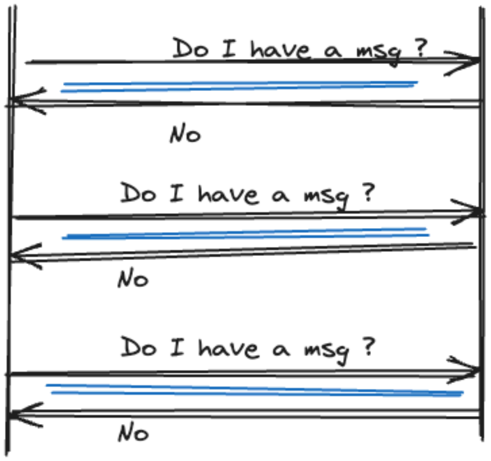
The network communications need to happen in a reliable fashion.
Then the transport layer protocol we need is TCP.

We need to think of a mechanism where recipients of the message don't have to refresh their page/app again and again to receive the message sent to them.

- Short Polling

- This polling is a simple req, res cycle, so can happen via HTTP



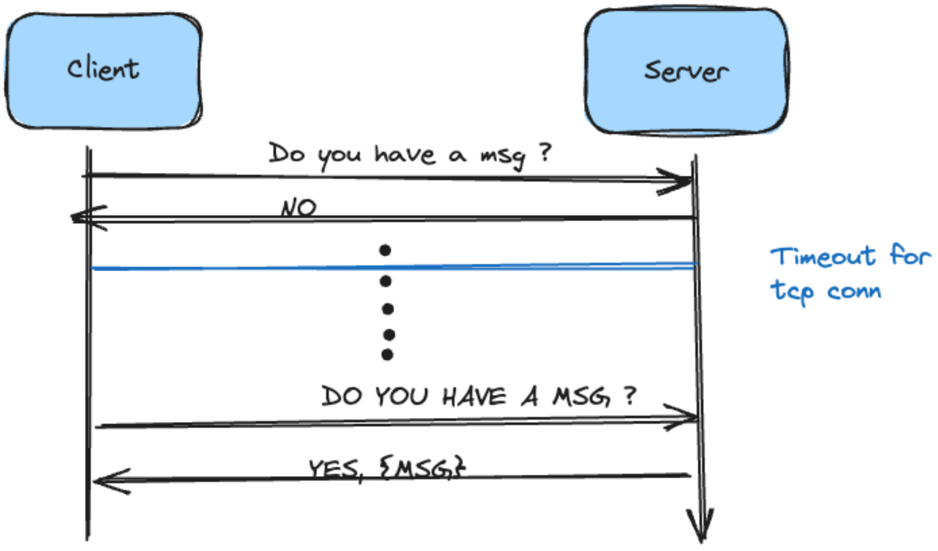


- Server is unnecessarily bombarded with a lot of http reqs, even when there is no response to send
- It increases the load on the server

So increases the load on the serve

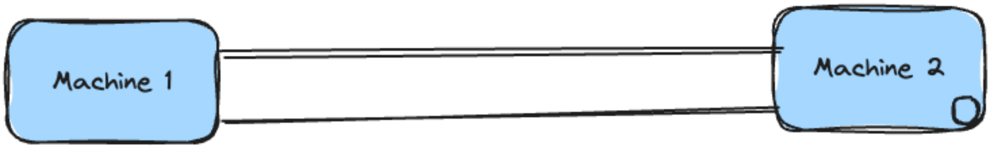
- Long Polling

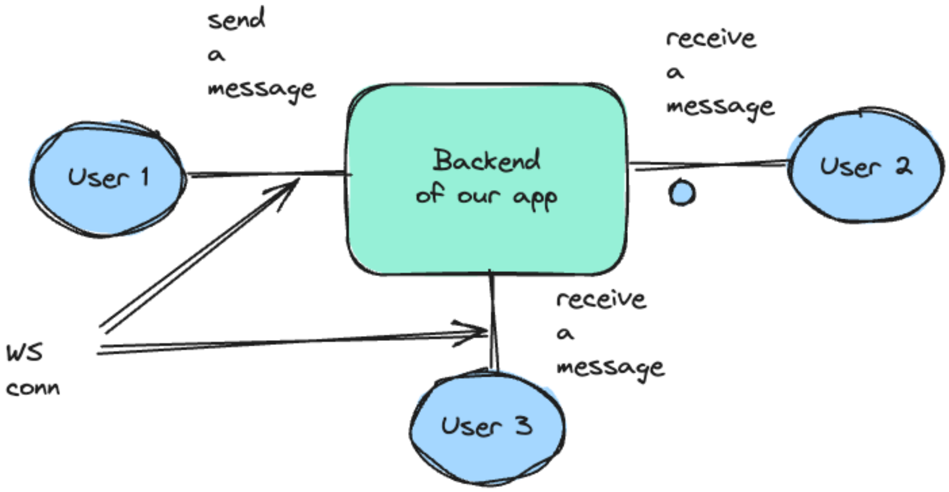
Instead of a burst of consecutive requests to server for checking for a message
you do a slow and long poll, i.e. after a poll you wait for sometime and then
send a req again.

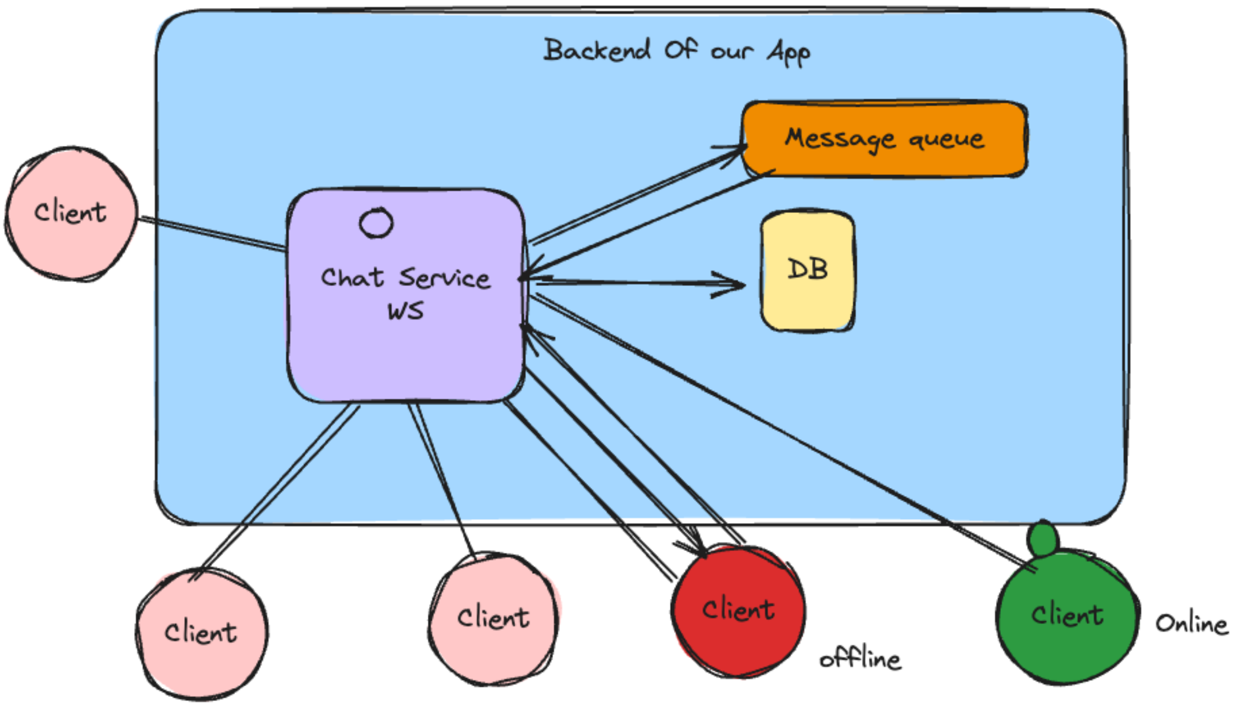


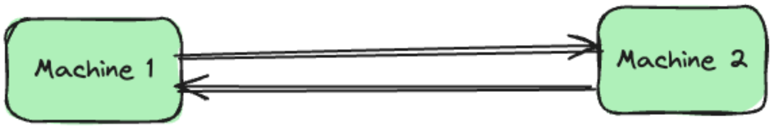
WebSockets

Protocol Built on top of TCP
It helps you to setup a bidirectional - persistent connection between two machines
To setup a ws conn, we first need to send an HTTP req from one machine to
other and then when the http req is received the other machine sends an Acknowledgement
that it is ready for a ws, and then this HTTP conn is upgraded to ws









1. Sync Comm
2. Async comm