

2. Technical Interview

General

- "What is your passion, what do you enjoy working on the most?"
- "What does your current/previous work-day like?"
- "Describe the project you've worked on that you're most proud of. What did you do that worked out particularly well?" (Tells us a lot about what they know, value, and have done on teams)
- "Describe the project you've worked on that you're least proud of. What would you do differently?" (Tells us a lot about how good they are at learning from their mistakes and how difficult it is for them to admit they made a mistake)

Pair Programming

White Board

reference: <https://www.tryexponent.com/blog/how-to-whiteboard-for-system-design-interviews> (in particular video on facebook messenger app design)

Text Messaging App (like Facebook Messenger)

• Requirements

- Two users need to be able to send messages to each other back and forth

• Other details

- date and time of each message
- who sent each message

How would you design the database to store required data?

- in Postgres / relational database?
- would it look and different in CosmosDB / NoSql db?

Requirements to add and see how db design changes (perhaps design it out in sql with all requirements and then see difference in design for no sql)

- Name a conversation
- show if a user is online or not
- add more than 2 users to a conversation

Architecture and Technical Goals - Interviewers can draw a simple architecture of users => load balancer => api servers => db - they can add to the architecture if needed

Low Latency - what are some strategies you could use to make sure users get new messages quickly? - try and get them to give pros and cons of each strategy - short polling (pulling data periodically) - long polling (server holds request until response is available) - web sockets (signal r, for server to client communication)

- High Volume
 - What are some strategies you could implement to handle really high volume? and how will each help with handling that high volume?
 - db?
 - no sql, how?
 - partitioning data correctly so load is distributed across many partitions
 - caching?
 - less traffic to the db
 - api
 - serverless functions
 - better scalability
- Reliability
 - What could you do to make the system more reliable / robust?
 - queueing system (azure storage queues, rabbit mq)
 - more responsive api server
 - throttle database load
 - publisher / subscriber
 - push model to allow services / users receive updates in real time and not add load to system to see if anything is available
 - event grid

Revision #3

Created 27 August 2024 03:19:41 by Quinn Godfrey

Updated 27 August 2024 03:21:22 by Quinn Godfrey