Technical Concepts Pt. 2

Agenda

1. System Design & Scalability
2. Technical Concepts
3. Object-Oriented Approach
4. Break-Out Rooms

1.
System Design & Scalability

How to Approach

What is System Design?

- Process of designing elements of a system
  - Architecture
  - Modules
  - Components
  - Interfaces
  - Data

4

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

Elements of a System

Architecture - This is the conceptual model that defines the structure, behavior and more views of a system. We can use flowcharts to represent and illustrate the architecture.

Modules - This are components that handle one specific tasks in a system. A combination of the modules make up the system.

Components - This provides a particular function or group of related functions. They are made up of modules.

Interfaces - This is the shared boundary across which the components of a the system exchange information and relate.

Data - This the management of the information and data flow.

https://medium.com/the-andela-way/system-design-in-software-development-f360ce6fcbb9

Actual experience with a wide range of tools and systems is an advantage, but being able to identify a need and suggest a common solution for it would get you a long way, even if you've never used it yourself.

- Pramp

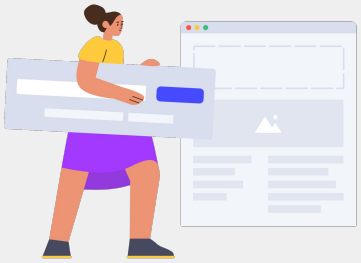
System Design Question

- Similar to technical system design questions
- Brainstorming session
  - Open-ended questions
- Don’t have to know everything!
  - Thought process behind design choices is more important
- There is no optimal solution
  - Everything is a tradeoff

- Technical, but not too technical
- The interviewer may grill you and dig deeper for the technical details on how certain components work, but you don’t have to know everything
- As long as you can identify a need and a well-known solution (that you don’t need to know everything about), you’re fine
- The interviewers are looking for you to have an understanding of how systems are built, since as a PM you need to have these conversations with engineers and software architects

Step-by-Step

1. **Go to the whiteboard!**
   a. Write/draw all your thoughts
2. **Ask clarifying questions**
   a. Understand the goal
   b. Narrow the scope
   c. Share your assumptions
3. **List out features you will be designing for**
   a. Lay out which areas to focus on
   b. Use cases



Clarifying questions:
What is the goal of the system?
Who are the users of the system?
What do they need it for?
How are they going to use it?
What are the inputs and outputs of the system?
Features:
What clients do we want to support (mobile, web, etc)?

Step-by-Step

4. **Draw out high-level approach**
    a. End-to-end user flow based on established goals
5. **Drill down into which data structures, algorithms, software solutions**
    a. Don't spend too much time on the details
6. **Discuss tradeoffs and edge cases**
    a. Pros and cons of different approaches

High Level Approach:
Keep it simple, and then iterate based on feedback from interviewer
Draw out major components like DB, servers
Drilling down:
Account for scaling vertically or horizontally -> will explain more in upcoming slides
Trade Offs:
What type of database would you use and why?
What caching solutions are out there?

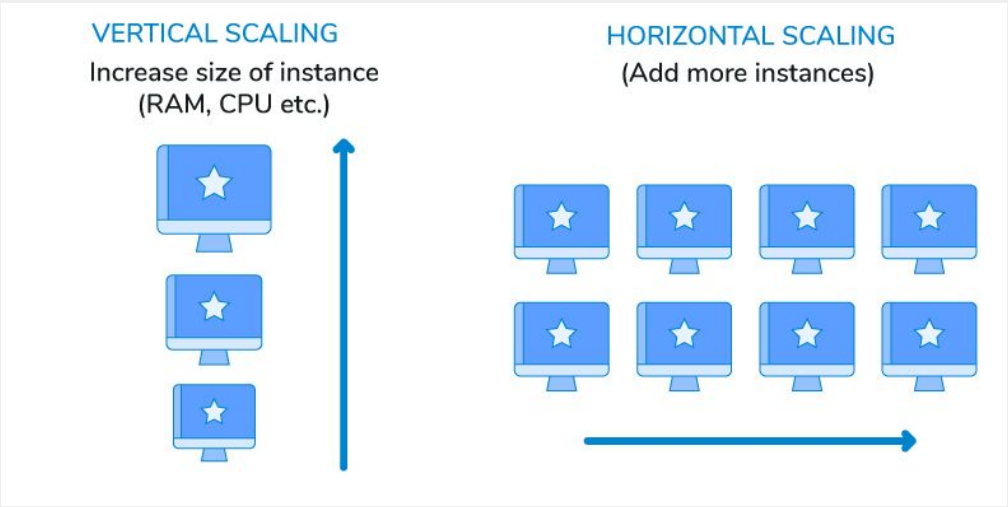
2.
Concepts

Be familiar with a broad range

And when it comes to distributed systems, it turns out, size really does *matter.*

- Vaidehi Joshi, Medium

Scalability



11

Scalability -> the ability to handle more input or increased load of the application
For example, if the number of users of the system increases by factors of 100 or more, the software will continue to function with comparable response times. And because of larger data size, we may add more servers or databases to handle the increased resources as a result.
Vertical scaling -> bigger machine
Horizontal scaling -> more machines

Pros & Cons

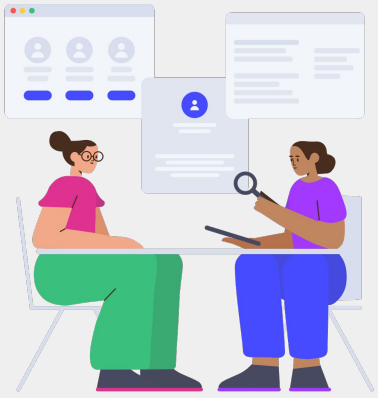
Horizontal Scaling	Vertical Scaling
Load Balancing Required	N/A
Resilient	Single point of failure
Network Calls	Inter-process communication
Data inconsistency	Consistent
Scales well as users increase	Hardware limitations

Load balancing - the process of distributing network or application traffic efficiently across multiple servers. This ensures no single server bears too much demand. By spreading the work evenly, load balancing improves application responsiveness. It also increases availability of applications and websites for users. Modern applications cannot run without load balancers.
Resilient - if one machine fails, can redirect input to another one
Network calls - all communication will be through servers for multiple machines which are slow
Inter-process communication inside one machine is fast - data is all in one machine
Data is harder to maintain between multiple machines - if one machine sends data to another and to another, lose transactional guarantee

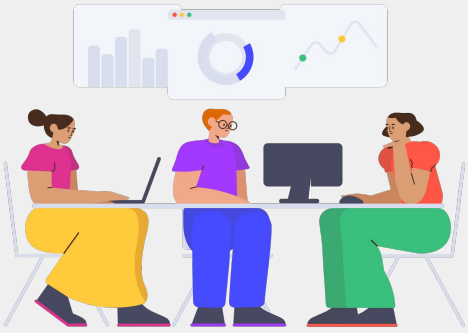
https://gist.github.com/vasanthk/485d1c25737e8e72759f

Main Takeaways

- Communicate clearly with your interviewer, but you're the driver
- Explore different directions
- Ask the right questions
- Examine the tradeoffs
- Don't drill down on details
- Be explicit about your assumptions



Technical Limitations



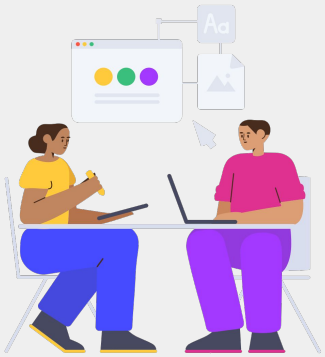
- Memory (RAM)
  - Data storage costs
- Execution time
  - Complexity of implementation

3.
Object-Oriented Approach

Java Time!

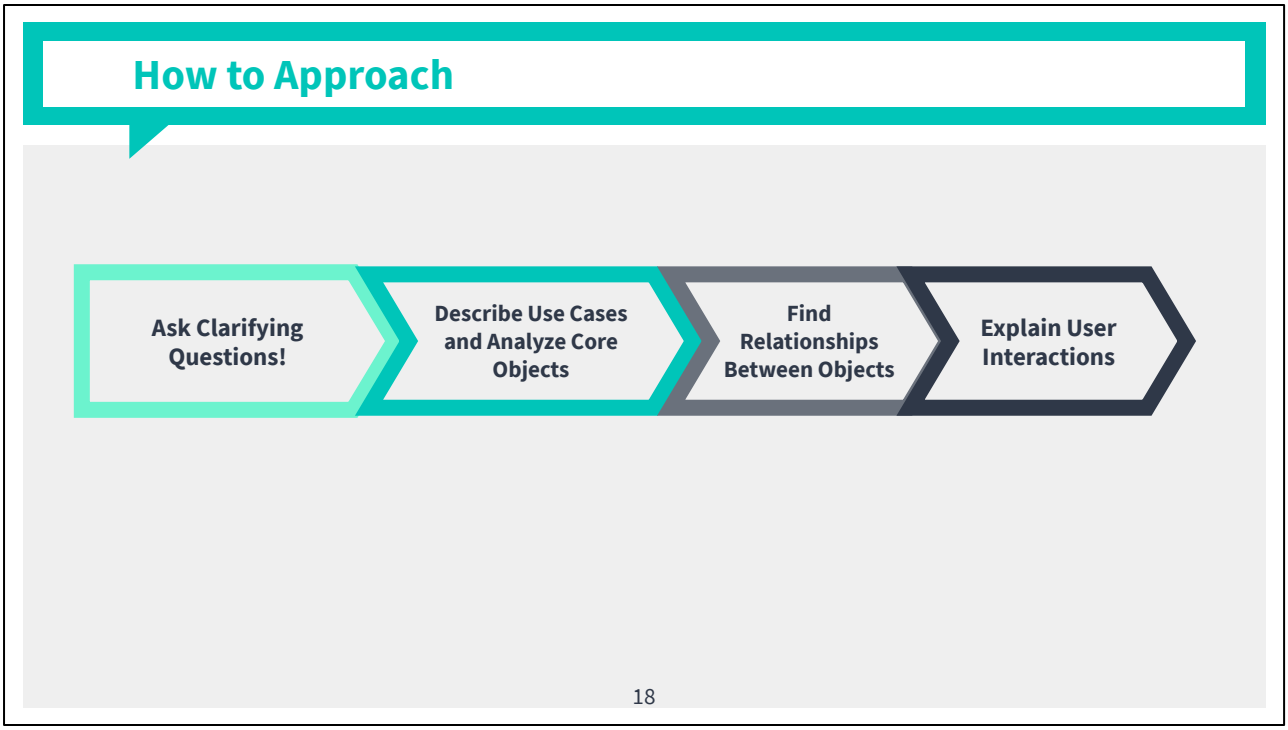
How does this relate to PM?

- Object-Oriented Design Questions
  - **Vision** for big picture
  - Show off your **technical** expertise
  - Ability to understand **end-user**
- Very similar to product design questions



Jukebox Example

- **System Components**
  - **Jukebox**
    - Plays songs and playlists
  - **CD**
    - Contains songs and artists
  - **Playlist**
    - Can add, delete, or queue songs in playlist
  - **Song**
    - Contains song name and artist name
  - **Artist**
    - Component of song



Find the right level

- The very first thing you need to do as a candidate is figuring out what output the interviewer is looking for. If it wasn't made clear in the question itself, don't be afraid to ask!
- "Are we going to be implementing the design here, or are you just looking for an overall class structure?" Anything to do with implementation and it's probably worth spending brain cycles on details like specific data structures. If they seem to be looking for the high level relationships, don't get sucked into the details!

Use cases

- Here is the key part: don't just think about your use cases, physically write them on the white board. This does two things. First, it sets a contract between you and the interviewer on what the important parts of the design are. As new use cases are introduced we will be updating this list. Later we can use it to double check that our design meets the requirements. Secondly, it's an incredibly easy way to identify the right objects.
- In each use case, the noun can be thought of as a core object and the verb can be the behaviors you need to model

Thanks!

Any questions?