Problem Solving & Design - R2

Introduction

The discussion focuses on the 360° tech aspect of day-to-day tech work.

The round covers discussion around:

- Problem-solving
- Low-level design
- High-level design
- Architectural aspects
    - Availability
    - Scalability
    - Performance
    - Fault tolerance

Problem Statement

The discussion generally focuses on the problems that one faces in day-to-day life/work and has real-world scenarios, constraints, and trade-offs.

For example, *A friend group generally spends together for their day-to-day needs, can you help them in identifying who owns how much to whom*.
On purpose, the problem may be somewhat fuzzy, for you to explore and scope the engineering problem out of it.

Focus Areas

- Requirement Understanding
    - Ask the relevant questions to clarify the scope of the problem
    - For real-world problems, this is crucial to identify the right scope of the problem and the right set of constraints and trade-offs.
    - Clarify any assumptions that you make
- Problem-solving
    - Once the problem is scoped, identify multiple approaches & solutions with trade-offs.
    - Able to choose the appropriate approaches & solutions based on the clarity and trade-offs.
    - The solution would satisfy and solve for the functional requirements & scope.

    - Identify any corner cases/edge cases and scenarios (with minimal guidance & help)
- Low-level design
    - The solution should be adapted to changing requirements and assumptions
    - Identify core models/entities and their attributes/properties.
    - Identify appropriate domain boundaries and relationships b/w the proposed entities.
    - Apply the right use of the data structures relevant to the problem
- High-level design
    - Identify the right system boundaries (with separation of concerns) & carve out modules/components
    - Dry run through the functionality problem asks
    - Always callout the trade-off b/w multiple solutions
- Architectural aspects
    - Cover aspects on fault-tolerance, performance, availability, consistency, scale, security
    - Identify the right trade-offs, appropriate tech stack choices (which satisfy the functional & non-functional requirements)
    - Try to access the use-cases and access pattern for the selection of high-level tech choices for components for example: Push vs Pull, Asynchronous vs synchronous

Examples:
Problem solving & Design:

1. Design a Bulk file uploader
    a. You are building an application where users can upload really large files to cloud servers (greater than 100 MB)
        i. Should be an async uploader, where the user selects the file/files to be uploaded and upon completion, the user is notified of the successful upload.
    b. These files should be downloadable on demand
    c. Extension:
        i. How would your design change if this was a video uploader
            1. And while downloading you would want the video to be streaming

Links & Resources

- Books
    - Fundamentals of Software Architecture
    - Clean Architecture
    - 97 things every software architect should know

    - Patterns of enterprise application architecture
    - Head first design patterns
- Videos & Links
    - Stability Patterns & Anti-patterns
    - Turning the database inside out by Martin Kleppmann
    - Event Sourcing
    - Domain-driven design by Eric Evans
    - Principles of Microservices by Sam Newman
    - SOLID Go Design by Dave Cheney