System Design
Skills Evaluation Framework

Technical Brief

CodeSignal

Introduction

Engineers who design large enterprise systems (e.g. Software Architects) are responsible for designing and planning different types of systems and choosing appropriate tools and techniques to meet the constraints and requirements for such systems. Some typical responsibilities for such engineers include:

- Analyzing system challenges and identifying the best solution to meet project needs and constraints
- Taking a project from conception to completion
- Optimizing systems while maintaining quality

With such responsibilities, hiring engineers or software architects skilled in system design practices is critical to the success of many organizations, and engineering and talent teams must be able to accurately evaluate the technical expertise of candidates for such roles. Unfortunately, due to a lack of industry standards, many hiring teams struggle to evaluate the technical skills of candidates properly, often using proxies (i.e., previous experience or pedigree) or poorly-designed evaluations that fail to effectively capture candidates' skills and knowledge.

This paper describes a framework for developing simulation-based evaluations which accurately capture signals about the technical skills of System Design candidates at scale. The current version of the framework will concentrate on fundamental building blocks of system design, excluding solutions provided by specific cloud providers (AWS, GCP, Azure, etc.).

Although Software Architects share many of the core skills and knowledge with Software Engineers such as problem-solving and code-writing, the System Design Framework is designed specifically to assess skills that are unique to system design, such as understanding storage types and being able to design secure, scalable, failure-tolerant and cost-effective applications.

This paper goes into detail about our guidelines for creating the framework based on consultation with subject matter experts with an emphasis on common core skills.

Framework Specifications

The framework is designed to closely model what the engineer would be expected to perform on the job. It can be utilized across different methods of delivery, assessment, or interview while preserving its objectivity by automatically calculating the final score.

The maximum allowed completion time for the framework is **60 minutes** and it consists of **4 progressive levels** that are designed to mimic an interactive system design interview. The scenario and assessment targets should be common applications and/or features that will not create any unfair advantages or disadvantages for any candidates. Possible scores range from 200 to 600.

Level 1 – Calculations and Estimations for the System

Completing requirements in the 1st level will result in a score of up to **25%** of the total. The average time for solving this level should be **15 minutes**.

Expected Knowledge

- Ability to gather relevant information about the system being designed
- Ability to estimate system and hardware requirements based on the functional requirements

Can Include

- Calculations and capacity estimations

Should Exclude

- Choosing data storage
- Basic or advanced design questions

Level 2 – Basic Design

Completing requirements in this level will result in a score between **25–50%** of the total. The average time for solving this level should be **15 minutes.**

Expected Knowledge

- Everything from the prior levels
- Understanding of storage types
- Choosing database solutions
- Estimating average and min/max loads
- Basic scaling

Can Include

- Calculations for average load
- Best storage types for the given data scheme
- Scaling strategies to handle min-max loads

Should Exclude

- Advanced design questions
- Replication techniques
- Data synchronization techniques

Level 3 – Advanced Design Strategies

Completing requirements in this level will result in a score between **50–75%** of the total. The average time for solving this level should be **15 minutes**.

Expected Knowledge

- Everything from the prior levels
- Redundancy and fault tolerance
- Database consistency and caching
- Consistency, availability, and performance

Can Include

- Load balancers
- Replication
- Message queues
- Communication between services

Should Exclude

- Cache coherency techniques
- Disaster recovery and availability on different geographical locations

Level 4 – Fine Tuning of the System to

Handle Additional Constraints and Edge Cases

Completing requirements in this level will result in a score between **75-100%** of the total. The average time for solving this level should be **15** minutes.

Expected Knowledge

- Everything from the prior levels
- Cache coherence
- Disaster recovery and availability
- zones
- CDNs
- Optimization of tools and techniques based on statistical data

Can Include

- Cost optimizations
- Consistency, availability, and/or performance improvements

Should Exclude

- Basic design related questions

Framework Example Content

Below is an example of a question that is established based on the structure of the framework. Similar questions are also created and monitored on an ongoing basis to ensure overall consistency as well as to prevent widespread cheating and plagiarism.

Scenario: Design a parking lot management system.

Imagine that you are designing an automated system for managing a parking lot. The system should consider:

- The lot has multiple floors, with each floor having several parking spots
- Each parking spot can be occupied by only one vehicle at a time
- Current capacity (remaining parking spots) should be monitored and displayed at the entrance
- Customers can:
  - Request to park by collecting a parking ticket upon entry, and paying at the exit
  - Buy a subscription to reserve a spot in the parking lot
- Average usage based on historical trends:
  - 2 new subscriptions per day
  - 15 000 parking requests per day

Functional requirements:

- Should be able to handle up to 30 000 parking requests per day
- Should be able to handle the capacity of 6 000 parking spots total across 4 floors
- Should be able to store all information for up to 10 years

Non-functional requirements:

- Should have high consistency and availability

Details about service costs:

- Database storage costs
  - Hot – 0.002 $/GB

- Cool – 0.001 $/GB
- Archive – 0.0005 $/GB

Other notes:

- 1 year = 365 days
- 1 KB = 1000 B

You are given the following data schemas for requests and subscriptions of the parking lot management system.

Requests (72B)

```
id: 8B
spotId: 8B
carPlateId: 8B
startTime: 16B
endTime: 16B
status: 16B
```

Subscription (72B)

```
id: 8B
userId: 8B
carPlateIds[3]: 24B
startDate: 16B
endDate: 16B
```

Note: subscriptions can be used for up to 3 different license plates.

Level 1 – Calculations and estimations for the system

Before diving into the design, determine the scope of system resources. For this level, your task is to calculate data requirements based on the expected usage of the parking lot.

Questions

1. What is the amount of requests data written per year in bytes?
2. What is the amount of subscriptions data written per year in bytes?

Level 2 – Advanced Design Strategies

Your task is to consider the basic design of the system.

Questions

1. What type of database should be used to ensure a high level of consistency?
2. You are developing a service to run some algorithms on old data – the data of the previous week will be run very frequently, however, the older data will be run a few times a year. What type of storage should be used to ensure that the architecture is cost-effective?

Level 3 – Advanced Design Strategies

For this level, your task is to extend the basic design so the system can handle corner cases and advanced con-

straints.

Questions
1. You have a car plate scanning service that verifies that either the car has a subscription or a ticket is booked upon entry. The service frequently fails due to camera failures. How would you improve the fault tolerance of the service? 2. There are performance issues at certain times of the day when the number of requests exceeds the average load. What type of scaling should be used to resolve the issue?

Level 4 – Fine Tuning of the System to Handle Additional Constraints and Edge Cases

For this level, your task is to optimize the system design based on feedback and service changes (i.e., structural changes to the parking lot).

Parking Lot Upgrade
As the parking lot became very popular, new entrances have been added to improve traffic flow within the lot.

Questions
1. How would you handle the consistency for the cars coming through the multiple entrances? 2. Analytics suggest that usage of the lot on Fridays to Sundays is almost twice as high compared to other days of the week. How do you react to this information?