

# CRACKING THE GAMAM TECHNICAL INTERVIEWS

Strategies, Tips, and  
Preparation resources

AN INSIDER'S GUIDE

---

**Dinesh Varyani**  
Engineer @ Google

# Cracking the GAMAM Technical Interviews

2nd Edition

# Cracking the **GAMAM** Technical Interviews, Second Edition

Copyright © 2023 Dinesh Varyani.

All rights reserved. No part of this book may be reproduced in any form by any electronic or mechanical means whatsoever without the author's written permission except for the use of brief quotations in a book review.

For more information, contact  
[hubberspot.publisher@gmail.com](mailto:hubberspot.publisher@gmail.com)

[www.hubberspot.com](http://www.hubberspot.com)

## Table of Contents

<b>Coding Interview</b>	<b>01</b>
<b>System Design Interview</b>	<b>02</b>
<b>Object-Oriented Design Interview</b>	<b>03</b>
<b>Schema Design Interview</b>	<b>04</b>
<b>API Design Interview</b>	<b>05</b>
<b>Behavioral Interview</b>	<b>06</b>
<b>Resume Tips</b>	<b>07</b>
<b>Preparation Strategy</b>	<b>08</b>
<b>Effective LeetCode</b>	<b>09</b>
<b>150 Days to <a href="#">GAMAM</a></b>	<b>10</b>
<b>GAMAM Progress Tracker</b>	<b>11</b>

**To my mother,  
who taught me that it's never too  
late to chase your dreams !!!**

# Preface

---

Dear Reader,

I am Dinesh Varyani, working as a Cloud Engineer at [Google](#). I am having over 12+ years of experience in Software Engineering. I am a passionate Youtuber, Blogger, Udemy & Educative Instructor, and now an author.

The main objective of this book is to help you in preparing and crack GAMAM (Google, Apple, Microsoft, Amazon, Meta) technical interviews. The prime focus of this book is to share preparation resources, strategies, tips, and a roadmap I followed that helped me clear [Google/Amazon](#) technical rounds. The book summarizes my journey like this -

- 👉 How I prepared myself for the GAMAM companies?
- 👉 What resources I used for various types of interviews?
- 👉 What strategies I used to master different topics?
- 👉 What roadmap I followed in months of my preparation?
- 👉 Which Resume tips got me to the recruiter's eye?
- 👉 How I tracked the progress of my preparation?
- 👉 My advice/tips on how to answer in a technical interviews?

Wish you all the best. I am sure you will find this book useful.

01

# Coding Interview

- Preparation resources
- do's and don'ts in an interview
- Things to do when you code
- Things to do when you are stuck

## Preparation resources

1. **Important DSA topics** - Array, Binary Search, Sliding Window, Matrix, Two Pointer, Intervals, Hash Map, String, Recursion, DP, Trees, Graph, Linked List, Stack, Queue & Heap
2. Solve [LeetCode Medium](#) level problems (at least more than 300+ covering different topics).
3. I have created an xlsx on top/important [500 LeetCode questions](#) and a video on [How to Crack The Coding Interview?](#)
4. [AlgoExpert's](#) 170 handpicked coding question (In case you want to prepare fast and only good questions)
5. Watch my [DSA playlist](#) to revise concepts.
6. [Grokking the Coding Interview: Patterns for Coding Questions](#)  
- The course is excellent and has covered various coding problems segregated based on coding patterns.



## Do's

- ✓ Keep a smiling face, and look confident/positive attitude person.
- ✓ Ask good clarifying questions about the coding problem e.g. size/range of the input, are there any duplicates, does input contain negative values, etc.
- ✓ Make the interview process a team effort. The more collaboration you do with your interviewer the more idea they get about how good team player you are.
- ✓ Think out loud. Always try to explain what you are thinking about the current state of the problem.
- ✓ Always be open to saying that you don't know how certain things work.

## Do's

- ✓ Always start thinking about the simpler version of the problem. Try to come up with a naive solution at first and later go for optimizing it.
- ✓ Listen carefully to what your interviewer wants and respond accordingly.
- ✓ Prepare a list of good questions related to the company, technology, work culture, etc, and always ask the interviewer at the end.

## Don'ts

- Never dive into solving a problem as soon as it's thrown toward you. Understand the problem, and resolve ambiguities.
- Never assume anything. Always clarify the assumptions you have with your interviewer.
- Avoid any technical jargon or famous words you know. If you do be prepared for the follow-up question.
- Never try to skip any idea or communication on which the interviewer wants to focus more.
- Not be too defensive about the mistakes that the interviewer tells you.

## When you code

- 👉 You are expected to write production-level code.
- 👉 Check for edge cases.
- 👉 Validate input and throw meaningful exceptions.
- 👉 Modularize code into different functions.
- 👉 Write meaningful variable/method names.
- 👉 You are expected to know the Time and Space complexity of the code you have written.
- 👉 You are expected to dry-run your code with the example given.
- 👉 Don't worry about the exact syntax of the code. Meaningful text can also convey the point you trying to achieve.
- 👉 Try to clean up code - check for any edge cases, refactoring, remove unwanted comments (in case you comment anything), check for conditions, etc.

## When you are stuck

- 👉 If you are stuck and unaware of any logic, just make/call a helper function (explain it will do XYZ)
- 👉 If you are stuck anywhere, your interviewer is the best person to help you out. Ask them for any hints or any question that clarifies your doubt. Remember the interviewer is not there to make you fail, they want you to succeed.
- 👉 If you are stuck in logic try to apply some coding patterns - like can two pointer help, can sort help, can binary search be applied, etc.

# Golden rules for solving a Coding problem in an Interview

- 👉 If the coding problem requires performing an operation that needs faster search in  $O(1)$ , try to use **Set** or a **Map**.
- 👉 If the coding problem requires finding/manipulating/dealing with the top, bottom, maximum, minimum, closest, and farthest "K" elements among given "N" elements, try to use a **Heap**.
- 👉 If the coding problem has input as a sorted **Array**, **List**, or **Matrix**, try to use **Two Pointer** strategy or try to use **Binary Search**.
- 👉 If the coding problem requires trying all **Permutations** and **Combinations**, we can use either **Backtracking** or **Breadth First Search**.
- 👉 If the coding problem has input in the form of a **Tree** or **Graph**, then most of the time it can be solved by applying Tree Traversals or Graph Traversals algorithms called as **Breadth First Search (BFS)** and **Depth First Search (DFS)**.
- 👉 If the coding problem is around a **Singly Linked List**, and you are stuck in traversals logic, then try to use either **Two Pointers** or **Slow/Fast Pointers**.

- 👉 If the coding problem has a recursive solution but it's hard to visualize/code, try using a **Stack** data structure with a loop.
- 👉 If the coding problem revolves around iterating an array, and takes  $O(N^2)$  time complexity,  $O(1)$  space complexity then try to use a **HashMap/HashSet**. It makes the algorithm faster with  $O(N)$  time complexity but takes more space with  $O(N)$  space complexity.
- 👉 If the coding problem revolves around iterating an array, and takes  $O(N^2)$  time complexity,  $O(1)$  space complexity then try to **sort** the array. It makes the algorithm faster with  $O(N \log N)$  time complexity and  $O(1)$  space complexity.
- 👉 If the coding problem requires optimization around the recursive the solution, there **could** be a possibility that **dynamic programming** can be used.
- 👉 If the coding problem has a group of strings or some manipulation/find/storing needs to be done around the substring, there is a high possibility that either **Tries** or **HashMap** can be used.



# Important Algorithms you should know before your Coding Interview

- 👉 **Singly Linked List Reversal**
- 👉 **Floyd Cycle Detection Algorithm**
- 👉 **Sliding Window**
- 👉 **Binary Search**
- 👉 **Kadane's Algorithm**
- 👉 **Quick Select**
- 👉 **Insertion Sort**
- 👉 **Selection Sort**
- 👉 **Counting Sort**
- 👉 **Heap Sort**
- 👉 **Merge Sort**
- 👉 **Quick Sort**
- 👉 **Topological Sort**

- 👉 **Zigzag Traversal of a Matrix**
- 👉 **Preorder Traversal of a Binary Tree**
- 👉 **Inorder Traversal of a Binary Tree**
- 👉 **Postorder Traversal of a Binary Tree**
- 👉 **Level Order Traversal**
- 👉 **Breadth First Search in a Graph**
- 👉 **Depth First Search in a Graph**
- 👉 **Flood Fill Algorithm**
- 👉 **Kruskal's Algorithm**
- 👉 **Floyd Warshall Algorithm**
- 👉 **Dijkstra's Algorithm**
- 👉 **Bellman Ford Algorithm**
- 👉 **Lee Algorithm**

- 👉 **Graph Bipartite**
- 👉 **Union-Find Algorithm**
- 👉 **KMP Algorithm**
- 👉 **Euclid's Algorithm**
- 👉 **Boyer–Moore Majority Vote Algorithm**
- 👉 **Dutch National Flag Algorithm**
- 👉 **Huffman Coding Algorithm**
- 👉 **Detect Cycle in a Directed Graph**
- 👉 **A\* Algorithm**

02

# System Design / High Level Design Interview

## Preparation resources

1. [Grokking the System Design Interview](#) - The course has step-by-step discussions on distributed systems are designed. It also covers good real-life case studies.
2. Alex Xu's System Design Interview course on [ByteByteGo](#) - The course covers all the content from his famous book (Vol 1 and Vol 2) System Design Interview.
3. [SystemsExpert](#) videos to know how real-life System Design Interviews go.

# Time Management in System Design Interview

- ✓ **Requirement Clarifications (5-7 min)**
- ✓ **Estimations (5-7 min)**
- ✓ **API Design (5-7 min)**
- ✓ **Database Schema Design (5-7 min)**
- ✓ **System's Detailed Design (15 - 20 min)**
- ✓ **Resolve bottlenecks and follow-up questions (2-3 min)**

# System Design Interview Template

A System Design Interview usually lasts for 45-60 minutes. The following template will guide you on how to manage time duration across various aspects of it -

## ✓ Requirement Clarifications - (5-7 min)

Ask clarifying questions to understand the problem and expectations of the interviewer.

### a) Functional Requirements

- 👉 Focused use cases to cover (MVP)
- 👉 Use cases that will not be covered
- 👉 Who will use the system
- 👉 Total/Daily active users
- 👉 How the system will be used

### b) Non Functional Requirements

- 👉 Is the system Highly Available or Highly Consistent?  
CAP theorem?
- 👉 Does the system requires low latency?
- 👉 Does the system needs to be reliable?



## ✓ **Estimations (5-7 min)**

- 👉 Latency/Throughput expectations
- 👉 QPS (Queries Per Second) Read/Write ratio
- 👉 Traffic estimates
- 👉 Storage estimates
- 👉 Memory estimates

## ✓ **API Design (5-7 min)**

- 👉 Outline the different APIs for required scenarios
- 👉 Identity request and response bodies required by APIs
- 👉 Identify HTTP methods APIs will target such as, GET, POST, DELETE, PATCH, etc
- 👉 Identity HTTP Status codes for different scenarios

## ✓ **Database Schema Design (5-7 min)**

- 👉 Identify the type of database (SQL or NoSQL)
- 👉 Design schema like tables/columns and relationships with other tables (SQL)

## ✓ **System's Detailed Design (15 - 20 min)**

(a) Draw/Explain high-level components of the system involving the below (if required) components -

- 👉 Client (Mobile, Browser)
- 👉 DNS
- 👉 CDN
- 👉 Load Balancers
- 👉 Web / Application Servers
- 👉 Microservices involved in fulfilling the design
- 👉 Blob / Object Storage
- 👉 Proxy/Reverse Proxy
- 👉 Database (SQL or NoSQL)
- 👉 Cache at various levels  
(Client side, CDN, Server side, Database side, Application level caching)
- 👉 Messaging Queues for asynchronous communication

(b) Identification of **algorithm/data structures** and way to scale them

(c) **Scaling individual components** - Horizontal & Vertical scaling

## (d) **Database Partitioning** -

### i) Partitioning Methods

- 👉 Horizontal Partitioning
- 👉 Vertical Partitioning
- 👉 Directory-Based Partitioning

### ii) Partitioning Criteria

- 👉 Range-Based Partitioning
- 👉 Hash-Based Partitioning (Consistent Hashing)
- 👉 Round Robin

## (e) **Replication & Redundancy** -

- 👉 Redundancy - Primary and Secondary Server
- 👉 Replication - Data replication from active to mirrored node/database

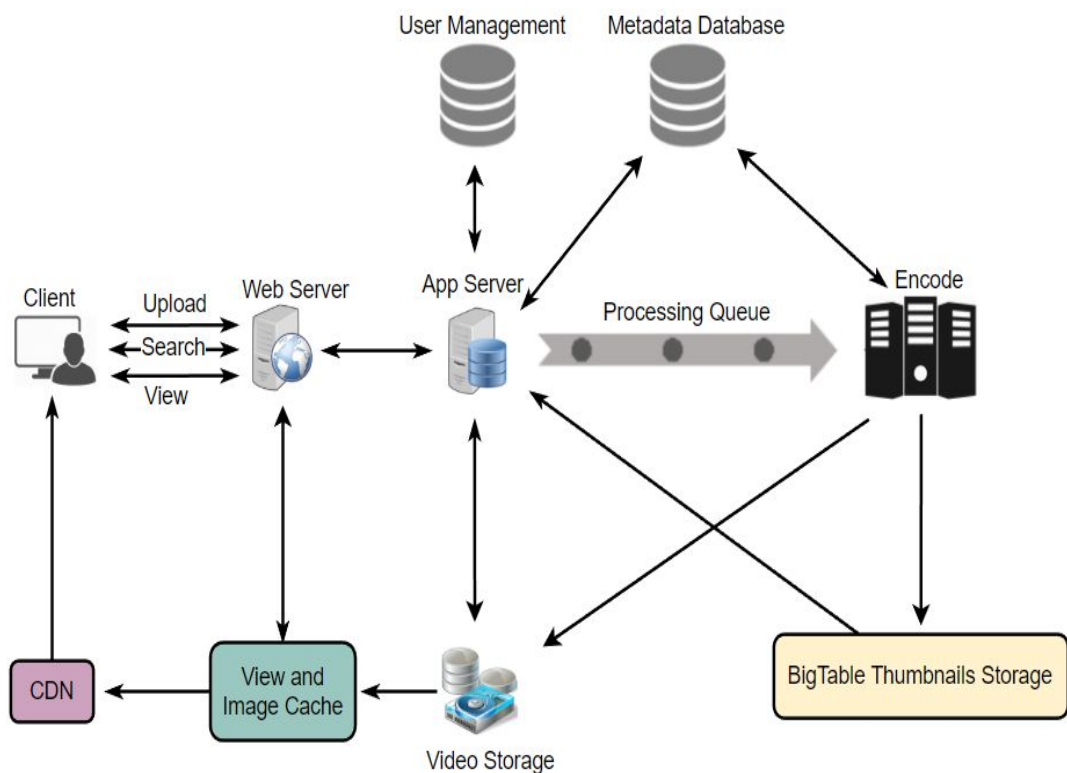
## (f) **Databases**

- 👉 SQL - Sharding, Indexes, master-slave, master-master, Denormalization
- 👉 NoSQL - Key-Value, Document, Wide-Column, Graph

## (g) Communication Protocols and standards like - IP, TCP, UDP, HTTP/S, RPC, REST, Web Sockets

✅ **Resolve bottlenecks and follow-up questions (2-3 min)**

Below kind of diagrams are expected in a System Design interviews



# **Must know Back-of-the-envelope Estimation for a System Design Interview**

# Must know Back-of-the-envelope estimations

In a system design interview, the chances are high that you will be asked to estimate QPS (Queries per second) and Storage of the system using a back-of-the-envelope estimation.

There are various scales and numbers anyone appearing for System Design Interview should know. Some of the scales are as follows -

## Number Scale

Name	Number	Number of Zeroes
1 hundred	100	2 Zeroes
1 thousand (K)	1000	3 Zeroes
1 million (M)	1000000	6 Zeroes
1 billion (B)	1000000000	9 Zeroes
1 trillion (T)	1000000000000	12 Zeroes
1 quadrillion	1000000000000000	15 Zeroes

## Power of Two's Scale

In a system design interview, the volume of huge data is measured on the power of 2. It gets as low as bits and bytes. A byte is measured as 8 bits. Estimations become easier if we co-relate the below table with the number table and make a rough approximation. The interviewer will expect you to know these scales.

Name	Power	Value
1 KB (Kilobyte)	$2^{10}$	1024 ~ 1K
1 MB (Megabyte)	$2^{20}$	1048576 ~ 1M
1 GB (Gigabyte)	$2^{30}$	1073741824 ~ 1B
1 TB (Terabyte)	$2^{40}$	1099511627776 ~ 1T
1 PB (Petabyte)	$2^{50}$	1125899906842624 ~ 1 Quadrillion

## Latency Numbers

In a system design interview, the latency numbers play a vital role in estimations and in having the knowledge, like how much time certain components take to perform certain operations. Below are some of the latency numbers of various operations -

Operation	Time taken
L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	100 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	10,000 ns = 10 $\mu$ s
Send 2K bytes over 1 Gbps network	20,000 ns = 20 $\mu$ s



## Latency Numbers

Operation	Time taken
Read 1 MB sequentially from memory	250,000 ns = 250 $\mu$ s
Round trip within the same datacenter	500,000 ns = 500 $\mu$ s
Disk seek	10,000,000 ns = 10 ms
Read 1 MB sequentially from the network	10,000,000 ns = 10 ms
Read 1 MB sequentially from disk	30,000,000 ns = 30 ms
Send packet CA (California) $\rightarrow$ Netherlands $\rightarrow$ CA	150,000,000 ns = 150 ms

\*\*\*ns = nanosecond,  $\mu$ s = microsecond, ms = millisecond

1 ns =  $10^{-9}$  seconds

1  $\mu$ s =  $10^{-6}$  seconds = 1,000 ns

1 ms =  $10^{-3}$  seconds = 1,000  $\mu$ s = 1,000,000 ns

## Availability Numbers

In a system design interview, the **High Availability** discussion will happen for sure. It is defined as the ability of the system to be operational for a longer period of time. Below are some of the availability numbers you should know -

Availability %	Downtime per year	Downtime per month	Downtime per day
90% (one nine)	36.53 days	73.05 hours	2.40 hours
99% (two nines)	3.65 days	7.31 hours	14.40 mins
99.9% (three nines)	8.77 hours	43.83 mins	1.44 mins
99.99% (four nines)	52.60 mins	4.38 mins	8.64 secs
99.999% (five nines)	5.26 mins	26.30 secs	864.00 millisecs
99.9999% (six nines)	31.56 secs	2.63 secs	86.40 millisecs
99.99999% (seven nines)	3.16 secs	262.98 millisecs	8.64 millisecs
99.999999% (eight nines)	315.58 millisecs	26.30 millisecs	864.00 microsecs
99.9999999% (nine nines)	31.56 millisecs	2.63 millisecs	86.40 microsecs

## Blob/Object Storage Sizes

In a system design, there are various big systems that involve types of blobs/objects like images, videos, audio, etc. Below are some of the approximate storage sizes of various blobs/objects -

Object Type	Size
char	1 B
char (Unicode)	2 B
UUID	16 B
Thumbnails	20-30 KB
Website image	200-300 KB
Mobile image	2-3 MB
Documents like books, reports, govt lds etc	1-3 MB
Audio files like songs, recordings etc	4-5 MB
1 min 720px video	60 MB
1 min 1080px video	130 MB
1 min 4K video	350 MB

## Example - Quora System Design

A System Design Interview usually lasts for 45-60 minutes. The following example "Quora System Design" will guide you on how to approach a system design interview -

### ✓ Requirement Clarifications - (3-5 min)

Ask clarifying questions to understand the problem and expectations of the interviewer. It is like an MVP which is the initial version of the product being designed. It has enough set of features usable by early customers who can then provide feedback for future product development. Let's see some of the functional and non-functional requirements of Quora.

#### a) Functional Requirements

- 👉 Users should be able to post questions.
- 👉 Users should be able to answer questions.
- 👉 Users should be able to upvote/downvote questions and answers.
- 👉 Users should be able to search questions.
- 👉 Users should be able to see the feed of relevant questions.

## b) Non-Functional Requirements

The design of the system is highly affected by the non-functional requirements of the system. **Availability** and **Consistency** are two major factors we need to look at while designing the system.

👉 **Availability** means that every request for data receives a response, even if it may not be the most up-to-date version of the data.

👉 **Consistency** means that all nodes in the system see the same data at the same time. Thus at any instant user is working on the same data (read and write the same data)

Other factors are -

👉 **Eventual Consistency** - In a distributed system Availability and Consistency can't be achieved simultaneously. Thus, many times we prefer Availability over Consistency. This means that system becomes highly available but can produce inconsistent reads and writes. In an eventually consistent system, multiple copies of the same data are stored on different nodes, and the nodes may not be in perfect sync with one another at all times. This means that it's possible for different nodes to have slightly different versions of the same data, but the system will eventually converge on a single, consistent state.

👉 **Latency** - It is the amount of time it takes for a request to be processed and a response to be returned. In a distributed system requests and responses can be handled by many nodes which are connected over a network. Thus, it becomes important to measure the latency of the system because of the network lag involved. It is typically measured in secs and millisecs. The system should have low latency which means requests should be fulfilled quickly. It improves the performance of the system.

Quora must have the following non-functional requirements -

- 👉 It should be Highly Available.
- 👉 It should have Eventual Consistency.
- 👉 It should have Low Latency.

Other non-functional requirements can be discussed with interviewer's scope in mind.

In a later section, we will see how we can achieve non-functional requirements.

## ✓ Estimations (3-5 min)

The estimations give a high-level idea about the scale of the system in the future. The scale of the system can be estimated by-

- 👉 Storage estimates
- 👉 QPS (Queries Per Second) Read/Write ratio

Let's understand both estimation in greater detail.

**a) Storage estimates** - Storage estimates can be estimated via a number of users using the system. It can be MAU (monthly active users) or DAU (daily active users). These numbers can be either discussed with the interviewer or you can make an assumption (with an agreement with the interviewer).

- Let's assume Quora has 300 Million MAU (monthly active users).
- Let's assume Quora has 40 Million DAU (daily active users).
- Let's assume out of 40 Million users daily only .01% post questions on Quora.

Thus, no. of questions each day = .01% of 40 Million  
= 4000 questions

## Storage requirements for Questions

Questions may have properties/attributes like -

- question\_id (unique question identifier)
- question\_title (actual question)
- topic\_id (questions category like sport, entertainment, etc)
- created\_datetime (creation date and time)
- user\_id (the user who posted it)

Let's assume that each question takes 1 KB of storage. Then, the space required for storing 4000 questions will be

$$= 1 \text{ KB} * 4000$$

$$= 4000 \text{ KB}$$

$$= 4 \text{ MB}$$

👉 **Storage required to store per day questions = 4 MB**

👉 **Storage required to store per year questions**

$$= 4 \text{ MB} * 365$$

$$= 1460 \text{ MB}$$

$$= 1.46 \text{ GB} \sim 1.5 \text{ GB}$$

👉 **Storage required to store 10-year questions = 15 GB**



## Storage requirements for Answers

Answers may have properties/attributes like -

- answer\_id (unique answer identifier)
- answer\_description (actual answer text)
- created\_datetime (creation date and time)
- user\_id (the user who posted it)

Let's assume that each answer takes 10 KB of storage. It is because the answer text can be a descriptive text.

Assume for each question we have an average of 3 answers.

Then, the space required for storing  $3 * 4000$  answers will be

$$= 10 \text{ KB} * 4000 * 3$$

$$= 120000 \text{ KB}$$

$$= 120 \text{ MB}$$

👉 **Storage required to store per day answers = 120 MB**

👉 **Storage required to store per year answers**

$$= 120 \text{ MB} * 365$$

$$= 43800 \text{ MB}$$

$$= 43.8 \text{ GB} \sim 45 \text{ GB}$$

👉 **Storage required to store 10-year answers = 450 GB**

👉 **Total storage required = 15 GB + 450 GB = 465 GB**

**b) QPS estimates** - QPS can be defined as queries per second. Its total of reads and writes per second occurring in a system. Reads are actions performed by users to view some resources such as questions, answers, etc. Writes are actions performed by users to post some resources such as adding questions, adding answers, etc.

## Reads in the system

Let's assume the number of pages visited by each user per day is 10. If we take 40 Million as daily active users then the number of reads per day will be =  $10 * 40 \text{ Million} = 400 \text{ Million reads per day}$ .

## Writes in the system

- 👉 Number of questions posted each day = 4000.
- 👉 Number of answers posted each day = 12000.
- 👉 Number of writes each day = 16000.

It can be seen that it's a read-heavy system because reads are more as compared to writes.

- 👉 Number of reads per day = 400 M
- 👉 Number of writes per day = 16 K
- 👉 Total number of queries per day =  $400 \text{ M} + 16 \text{ K} \sim 400 \text{ M}$
- 👉 Total seconds in a day =  $24 * 60 * 60 = 86400 \sim 100 \text{ K secs}$
- 👉 **Queries per secs =  $400 \text{ M} / 100 \text{ K} \sim 4 \text{ K queries per secs}$**

## ✓ API Design (3-5 min)

Outline the different APIs for the required scenarios. The APIs should cover all the requirements of the system. These functional requirements are the same as what has been captured in the **Requirement Clarification** step. For Quora, some of the APIs would be -

### 👉 Post Questions - Add a new question

- **Url** - /questions
- **HTTP Method** - POST
- **Request Body** -

```
{  
  "question_id": "27baf31e-7cf5-11ed-a1eb-0242ac120002",  
  "question_title": "How to crack GAMAM interviews",  
  "topic_id": "10",  
  "created_datetime": "16/12/2023T13:45:00.000Z",  
  "user_id": "37baf31e-8cf5-11ed-a1eb-0242ac560002"  
}
```

- **Response Body** - Returns a string "Question added successfully" with status code 201. Otherwise returns a failure string with error code 4xx (client-side error) or 5xx (server-side error)

## ✓ API Design (3-5 min)

### 👉 Post Answer - Add a new answer

- Url - /answers
- HTTP Method - POST
- Request Body -

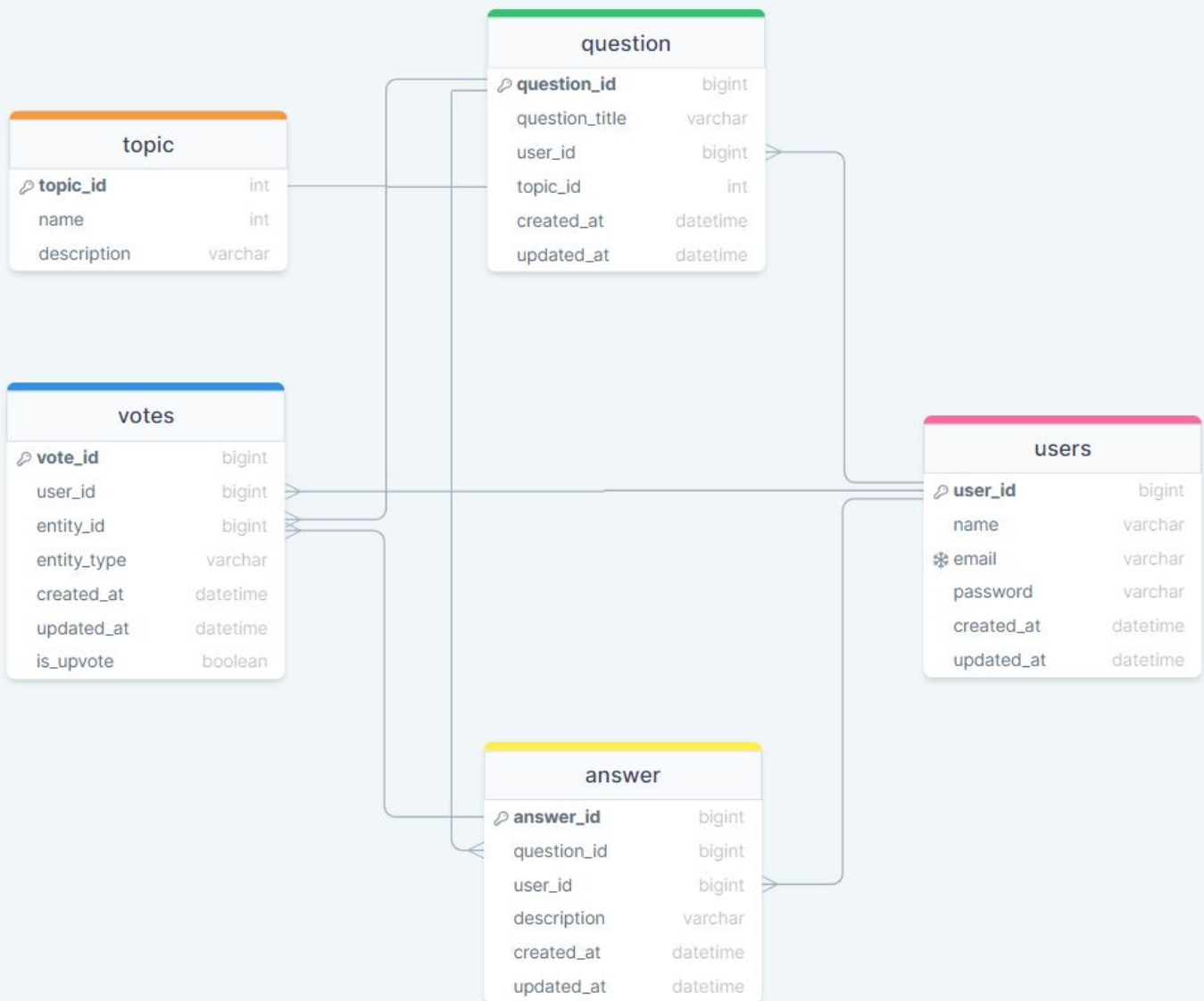
```
{  
  "answer_id": "35caf31e-7cf5-11ed-a1eb-0242ac120010",  
  "question_id": "27baf31e-7cf5-11ed-a1eb-0242ac120002",  
  "answer_description": "Practice DSA and Design daily !!!",  
  "created_datetime": "16/12/2023T13:45:00.000Z",  
  "user_id": "30baf31e-8cf5-13ed-a1eb-0242ac260019"  
}
```

- Response Body - Returns a string "Answer added successfully" with status code 201. Otherwise returns a failure string with error code 4xx (client-side error) or 5xx (server-side error)

and other similar apis ...

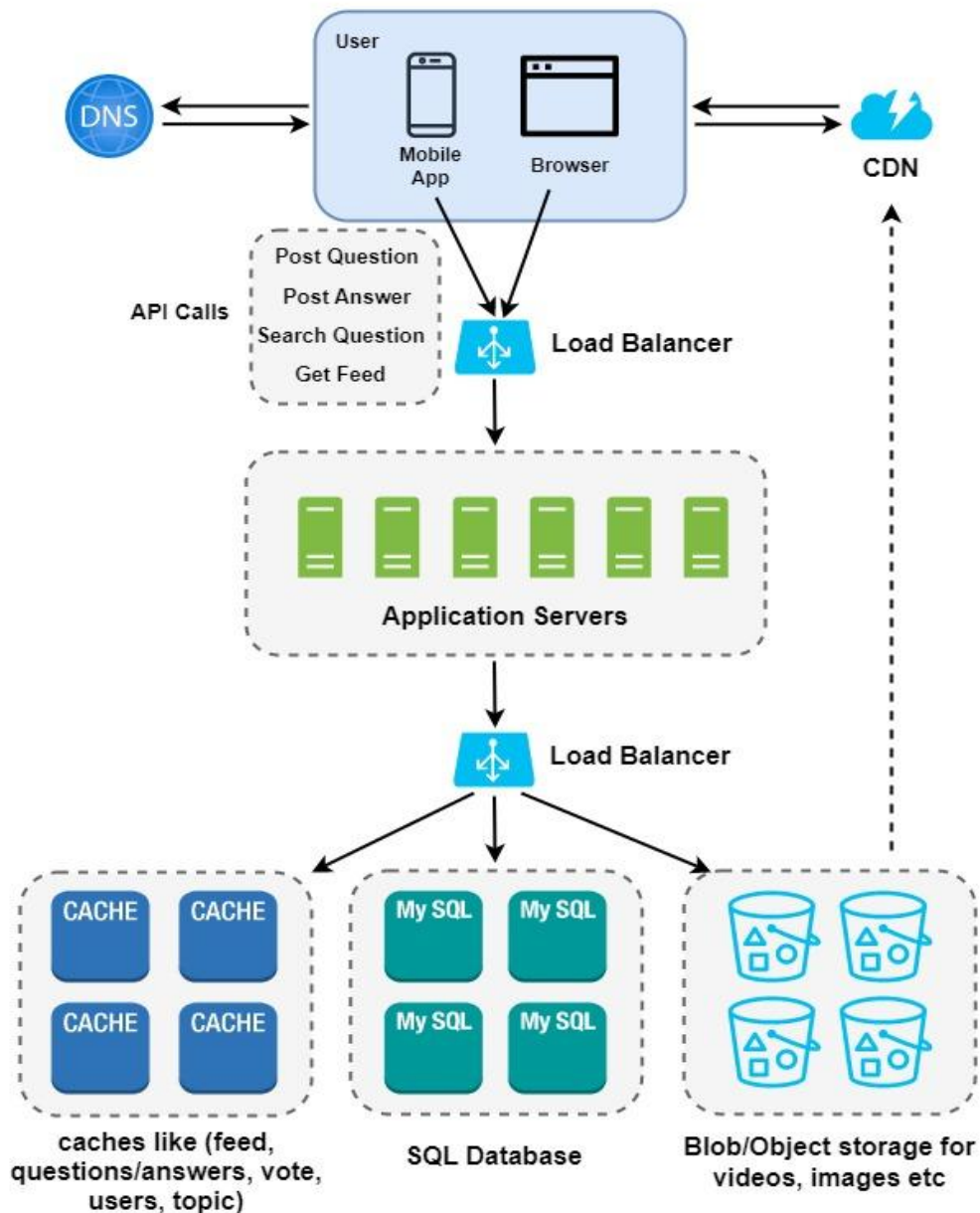
## ✓ Database Design (3-5 min)

👉 Design schema like tables/columns and relationships with other tables (SQL). For Quora's MVP below image demonstrate a good starting schema -



## ✓ System's Detailed Design (20 - 25 min)

👉 Draw/Explain high-level components (detailed design) of the system like the below diagram -



# Golden Rules to answer in a System Design Interview

- 👉 If we are dealing with a read-heavy system, it's good to consider using a **Cache**.
- 👉 If we need low latency in the system, it's good to consider using a **Cache & CDN**.
- 👉 If we are dealing with a write-heavy system, it's good to use a **Message Queue** for Async processing.
- 👉 If we need a system to be an ACID complaint, we should go for **RDBMS or SQL Database**.
- 👉 If data is unstructured & doesn't require ACID properties, we should go for **NoSQL Database**.
- 👉 If the system has complex data in the form of videos, images, files etc, we should go for **Blob/Object storage**.
- 👉 If the system requires complex pre-computation like a news feed, we should use a **Message Queue & Cache**.
- 👉 If the system requires searching data in high volume, we should consider using a **search index, tries, or a search engine like Elasticsearch**.



- 👉 If the system requires to Scale SQL Database, we should consider using **Database Sharding**.
- 👉 If the system requires High Availability, Performance, & Throughput, we should consider using a **Load Balancer**.
- 👉 If the system requires faster data delivery globally, reliability, high availability, & performance, we should consider using a **Content Delivery Network (CDN)**.
- 👉 If the system has data with nodes, edges, and relationships like friend lists, & road connections, we should consider using a **Graph Database**.
- 👉 If the system needs scaling of various components like servers, databases, etc, we should consider using **Horizontal Scaling**.
- 👉 If the system requires high-performing database queries, we should use **Database Indexes**.
- 👉 If the system requires bulk job processing, we should consider using **Batch Processing & Message Queues**.

- 👉 If the system requires reducing server load and preventing DOS attacks, we should use a **Rate Limiter**.
- 👉 If the system has microservices, we should consider using an **API Gateway** (Authentication, SSL Termination, Routing etc)
- 👉 If the system has a single point of failure, we should implement **Redundancy** in that component.
- 👉 If the system needs to be fault-tolerant, & durable, we should implement **Data Replication** (creating multiple copies of data on different servers).
- 👉 If the system needs user-to-user communication (bi-directional) in a fast way, we should use **Websockets**.
- 👉 If the system needs the ability to detect failures in a distributed system, we should implement a **Heartbeat**.
- 👉 If the system needs to ensure data integrity, we should use **Checksum** algorithm.
- 👉 If asked how to limit the huge amount of data for a network request like youtube search, trending videos etc. One way is to implement **Pagination** which limits response data.

- 👉 If the system needs to transfer data between various servers in a decentralized way, we should go for the **Gossip Protocol**.
- 👉 If the system needs to scale servers with add/removal of nodes efficiently, with no hotspots, we should implement **Consistent Hashing**.
- 👉 If the system needs anything to deal with a location like maps, nearby resources, we should consider using **Quadtree, Geohash, etc.**
- 👉 Avoid using any specific technology names such as - **Kafka, S3, or EC2**. Try to use more generic names like message queues, object storage etc.
- 👉 If **High Availability (HA)** is required in the system, it's better to mention that system cannot have Strong Consistency. However, Eventual Consistency is possible.
- 👉 If asked how domain name query in the browser works and resolves IP addresses. Try to sketch or mention about **DNS (Domain Name System)**.
- 👉 If asked which policy you would use to evict a Cache. The preferred/asked Cache eviction policy is **LRU (Least Recently Used)** Cache. Prepare around its Data Structure and Implementation.

03

# Object Oriented / Low Level Design Interview

## Preparation resources

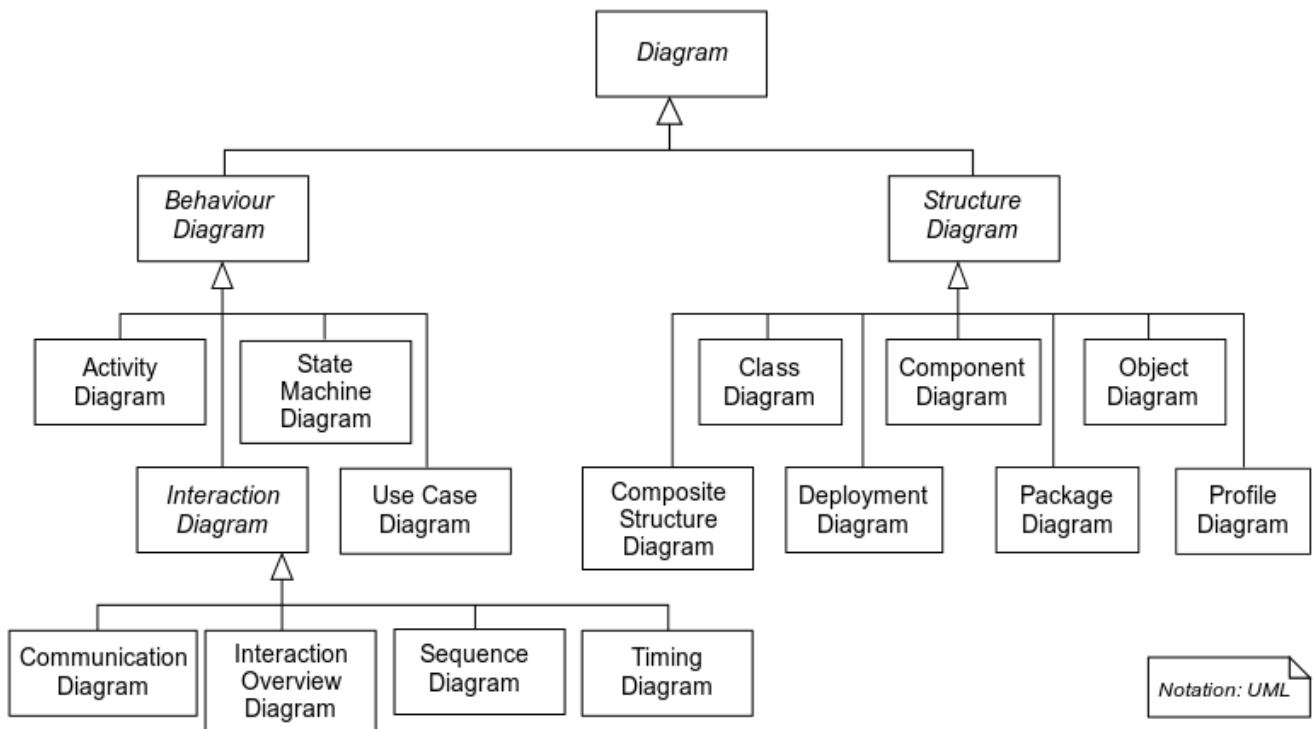
1. [Grokking the Object Oriented Design Interview](#) - A very detailed and step-by-step approach to various object-oriented design case studies.
2. Try practicing drawing UML diagrams like - Class diagrams, Use case diagrams, and Activity diagrams. Try to use pen and paper or practice on [Diagrams.net](#)

# UML and Class Diagrams in a Object-Oriented Design

# UML (Unified Modeling Language)

The Unified Modeling Language (UML) is a general-purpose, developmental modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. It helps to visually represent the architecture, design, and implementation of a system.

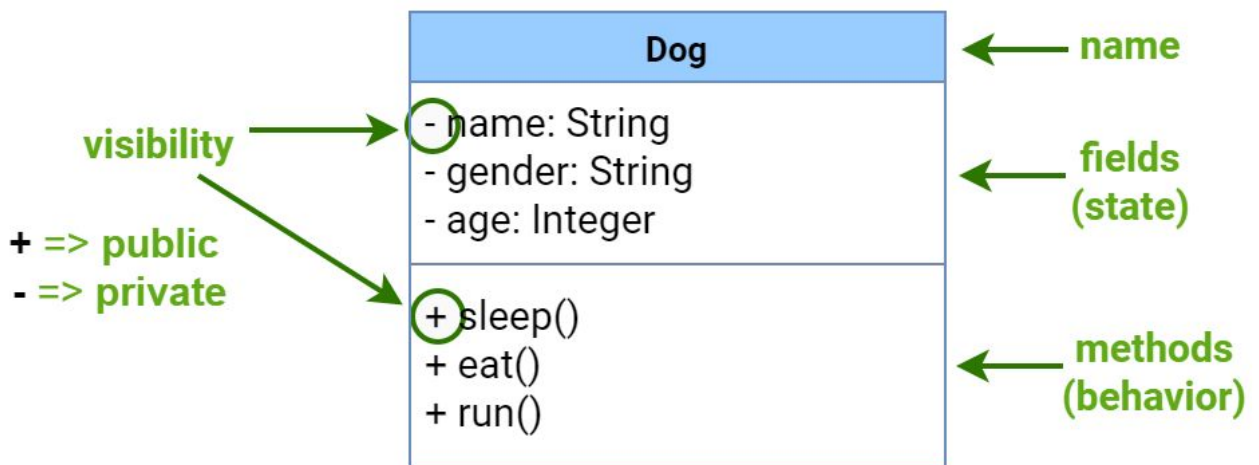
The various types of important UML diagrams are -



The most important UML diagram for Object-Oriented Design interviews is **Class Diagram**. Let's look at and discuss the diagram in more detail.

# Class Diagram

The Class Diagram in the Unified Modeling Language (UML) is the most useful and important diagram that is being focused on in a low-level design interview. It is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



In the diagram, classes are represented with boxes that contain three compartments:

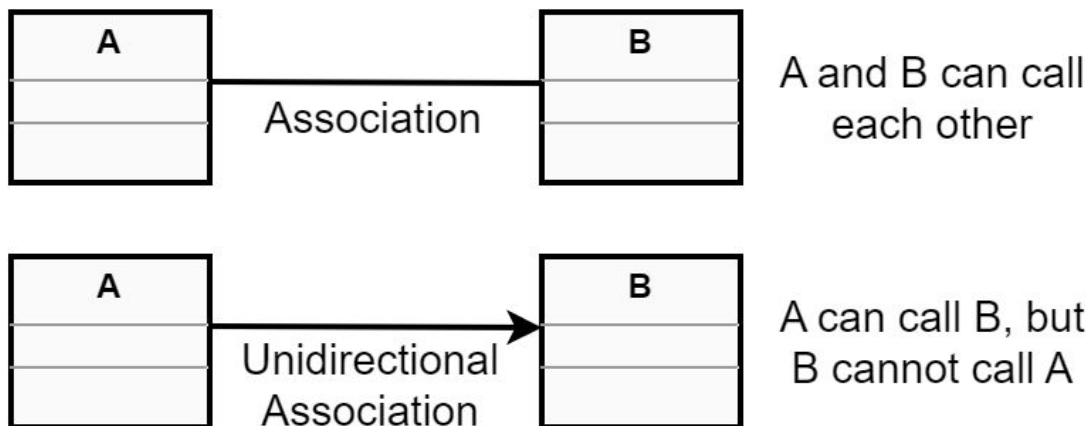
- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.



# Relationships

The Class Diagram shows the classes, their attributes, methods, and the relationships between different classes. Several types of relationships can be represented in class diagrams, each indicating a different type of association between classes. Here are some of the most common relationships:

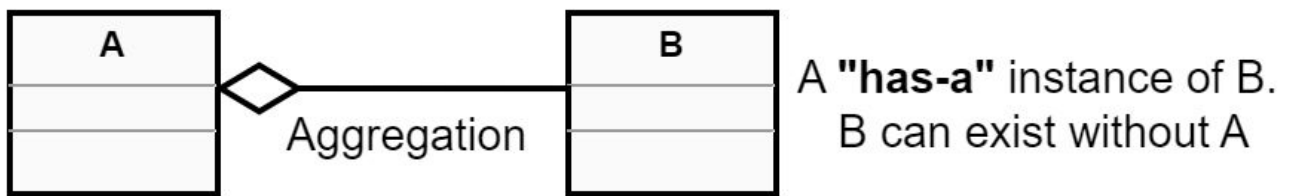
👉 **Association** - An association represents a relationship between two classes, where objects of one class are related to objects of the other class in some way. The relationship can be bi-directional or unidirectional. In a bi-directional relationship, both classes are aware of each other and their relationship. In a unidirectional relationship, two classes are related, but only one knows about the other and can call it.



For example, consider the classes "Car" and "Driver". There is an association between these classes because a car is driven by a driver. The association can be represented with an arrow pointing from the car class to the driver class, indicating the direction of the relationship.

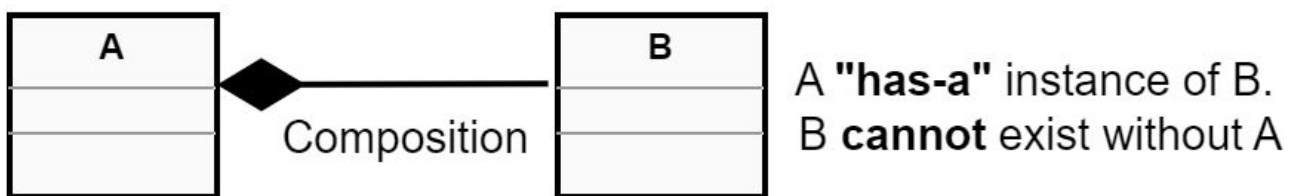
## Relationships

**Aggregation** - Aggregation is a special type of association that represents a **"has-a"** relationship between classes. It indicates that one class as a **"whole"** contains objects of another class as a **"part"** of its state. The **"part"** class can exist independently without **"whole"** class. It is being represented by diamond symbol.



For example, consider the classes "University" and "Department". A university has several departments, and a department belongs to one university. This can be represented using an aggregation relationship, with a diamond symbol on the side of the university class pointing to the department class.

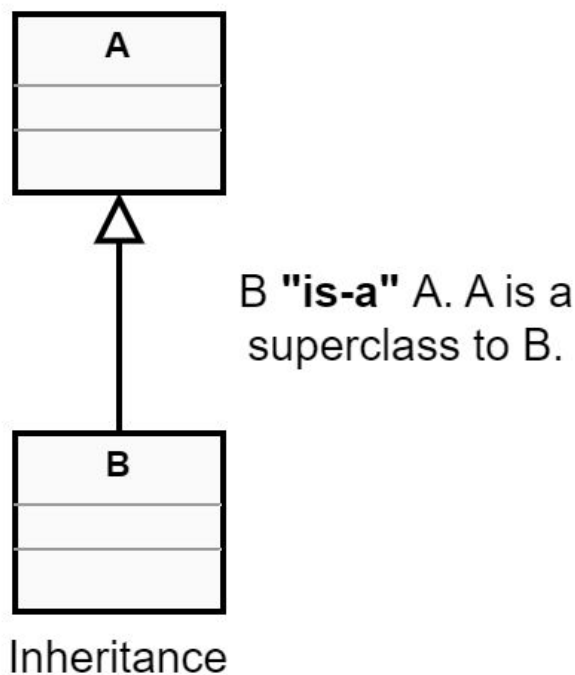
**Composition** - It is similar to aggregation, but it represents a stronger **"whole-part"** relationship between classes. It indicates that one class is made up of one or more objects of another class, and those objects cannot exist without the parent class.



For example, consider the classes "House" and "Room". A house is composed of one or more rooms, and a room cannot exist without a house. This can be represented using a filled diamond symbol on the side of the house class pointing to the room class.

# Relationships

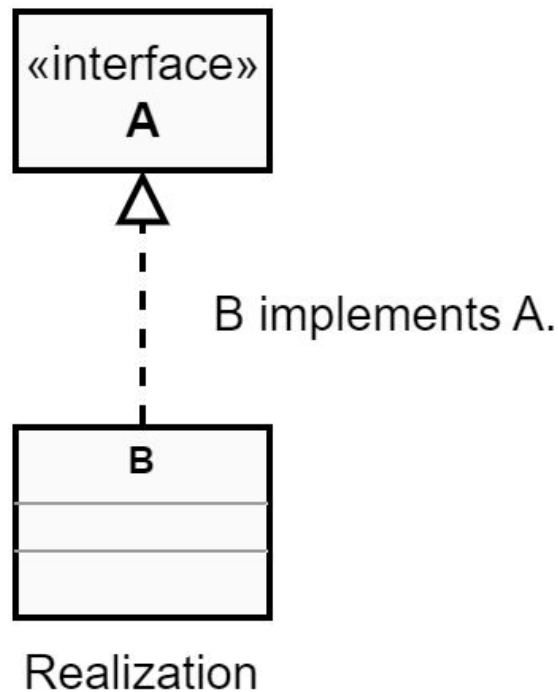
**Inheritance** - Inheritance represents an **"is-a"** relationship between classes, where one class inherits the attributes and methods of another class. The inherited class is called the superclass or parent class, and the inheriting class is called the subclass or child class.



For example, consider the classes "Animal" and "Cat". A cat is an animal, so the class "Cat" can inherit from the class "Animal". This can be represented using an arrow pointing from the subclass to the superclass, with a hollow triangle symbol on the side of the superclass.

## Relationships

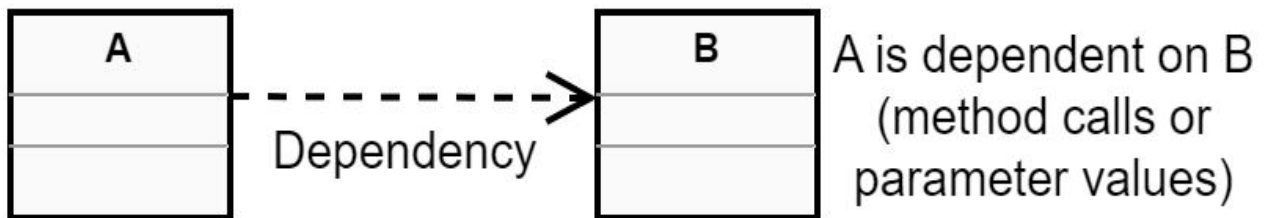
**Realization** - A realization relationship is a relationship between two entities, in which one class implements the behavior of the other class or interface. It is represented as a hollow triangle shape on the interface end of the dashed line that connects it to one or more implementers.



For example, consider an interface "**Shape**" which has the draw() method. The classes such as "**Rectangle**" and "**Circle**" implements draw() method of Shape and provide their implementation like how a Rectangle or Circle can be drawn.

## Relationships

**Dependency** - Dependency represents a weaker relationship between classes, where one class depends on another class in some way. This can be because the dependent class uses objects or methods of the other class, or because it receives parameters or returns values from the other class.



For example, consider the classes "BillingSystem" and "Customer". The billing system depends on the customer class to get information about customers' orders, billing addresses, etc. This can be represented with a dashed arrow pointing from the billing system class to the customer class.

# Multiplicity

Multiplicity refers to the number of instances of one class that can be associated with the instances of another class. It specifies the range of valid cardinalities for a given association between classes. It is represented by two integers separated which tells a range. The first integer represents the minimum number of instances of the related class that must be associated with an instance of the source class, while the second integer represents the maximum number of instances of the related class that can be associated with an instance of the source class. It can be applied to all UML relationships.



For example, consider a relationship between the classes "Order" and "Item", where an order can have one or more items, but an item can only be associated with one order. It is represented as **1..\* (One to Many)**

Other common multiplicity values include:

**0..1** - Indicates that zero or one instance of the related class can be associated with an instance of the source class.

**1..1** - Indicates that exactly one instance of the related class must be associated with an instance of the source class.

**0..\*** - Indicates that zero or many instances of the related class can be associated with an instance of the source class.

# Time Management in Object Oriented Design Interview

- ✓ Requirement Gathering (3-5 mins)
- ✓ Use Cases (3-6 mins)
- ✓ Identify the Core classes (3-6 mins)
- ✓ Identify the fields of each class (5-10 mins)
- ✓ Identify the Relationship between the classes (5-10 mins)
- ✓ Identify the Actions of the classes (5-10 mins)
- ✓ Code (5-8 mins)
- ✓ Follow-up questions (3-4 mins)

# Object Oriented Design Interview Template

An Object Oriented Design Interview usually lasts for 45-60 minutes. The following template will guide you (with a basic example to get you an idea) on how to manage time duration across various aspects of it -

## ✅ Requirement Clarifications (3-5 mins)

The OOD interview questions are often abstract and vague. Always ask clarifying questions to understand the problem and find the exact scope of the system that the interviewer has in mind.

- 👉 Focused use cases to cover (MVP)
- 👉 Use cases that will not be covered
- 👉 Who will use the system (Actors)
- 👉 How the system will be used

e.g. Let's **design an online shopping site**. A few requirements would be -

- 👉 Users should be able to search products based on the name.
- 👉 Users should be able to view/buy products.
- 👉 Users should be able to add/remove product items in their shopping cart.
- 👉 User can place an order.
- 👉 Users should get notifications about orders.
- 👉 User should be able to pay through different modes.



## ✅ Use cases to cover (3-6 mins)

You are expected to find possible actors of the system and write down different use cases the system will support.

👉 Actors could be -

1. Customer
2. Admin
3. System

Some of the use cases could be -

- 👉 Search products based on the name.
- 👉 add/remove/modify products in the shopping cart.
- 👉 Check out to buy items in the shopping cart.
- 👉 Make payment to place an order
- 👉 Send a notification to the user about the order.

## ✓ Identify the Core classes (3-6 mins)

After gathering requirements and drafting a few use cases, our understanding of what we are designing becomes clear. Now we should consider what could be the main classes of the system. You are expected to sketch a class diagram or write down class names.

The way to identify classes or entities is -

👉 Nouns in the requirements are possible candidates for Classes.

Some of the core classes could be -

- 👉 Product
- 👉 Item
- 👉 User
- 👉 ShoppingCart
- 👉 Order
- 👉 Payment
- 👉 Notification

## ✓ Identify the fields/properties of each class (5-10 mins)

Once we know the core classes/objects of the system, it is expected to draw a class diagram along with class fields. Take each class identified in the above steps and add a few important properties which drive the use cases of the system.  
eg.

### 👉 Product

- name
- description
- price ...

### 👉 User

- name
- email
- phone ...

## ✓ Identify the relationship between the classes (5-10 mins)

Once we know the core classes/objects of the system, it is expected to draw what is the relationship between the classes. The relationship mainly focuses on is-a and has-a between classes. The different types of relationships to draw or write are -

👉 Are there any classes that are very generic and more concrete classes that can be sketched?

👉 Are there any one-to-one, one-to-many, and many-to-many relationships between the classes. eg.

👉 Customer, Guest, and Admin inherit from User

👉 Customer has One Shopping Cart

👉 Shopping Cart has Many Items  
etc ...

## ✓ Identify the possible actions of the classes (5-10 mins)

Once we are clear with the requirements, use cases and possible design of the system, etc, it is time to identify the different actions classes will perform based on their relationship.

The way to identify class actions is -

👉 Verbs in the requirements/use cases are possible candidates for actions these classes perform. Those can be taken as methods of the classes.

eg.

👉 Customer can add the item to the shopping cart -

**addItemToCart(Item item)**

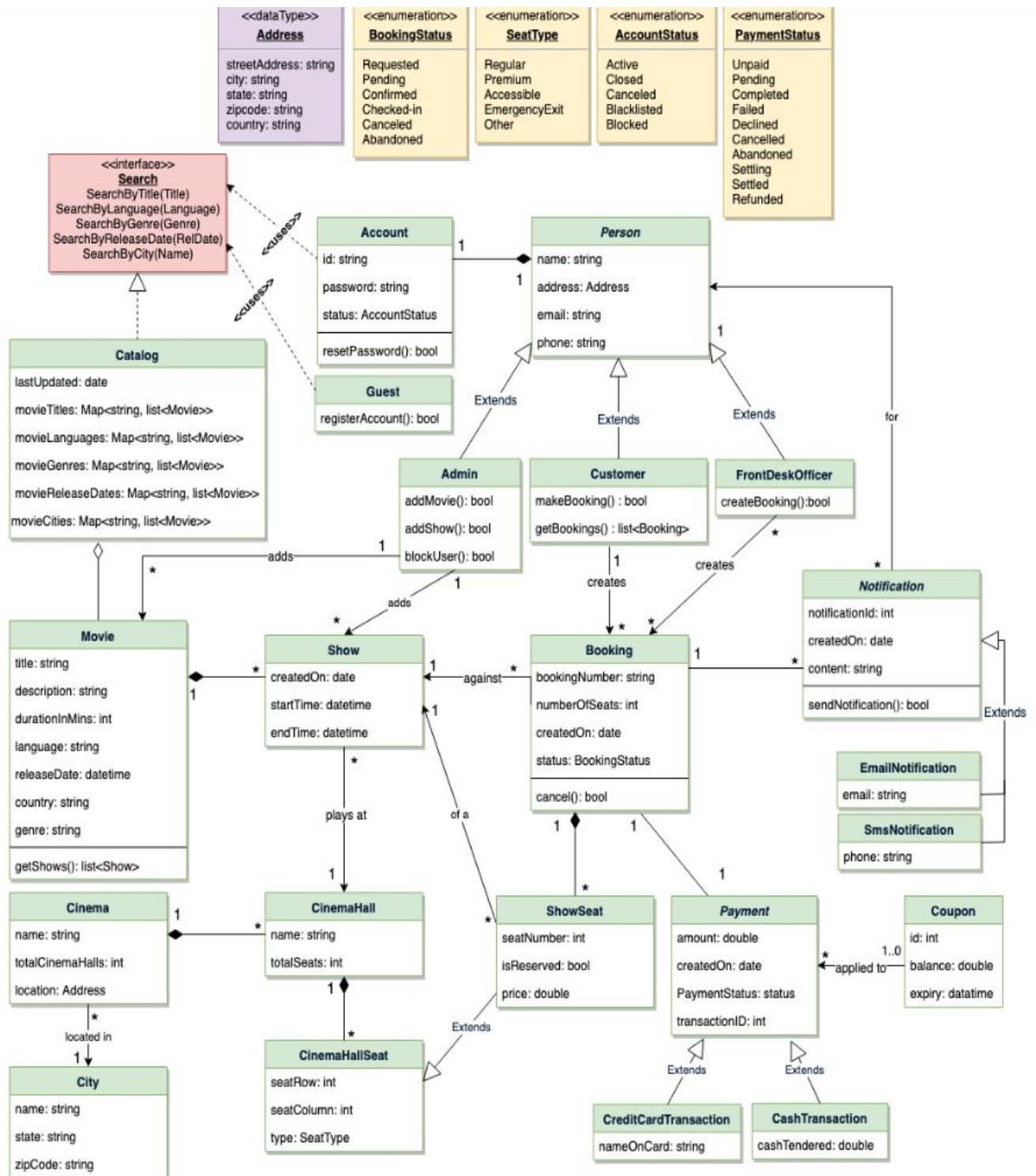
👉 Customer can place an order - **placeOrder(Order order)**  
etc...

### ✓ **Code (5-8 mins) (Optional)**

The interviewer will ask you to write code for a specific use case by taking the above classes. The class diagram will give you an idea about the class's name, fields, and methods. You are expected to write code for the methods which fulfill the use case interviewer wants or any algorithm/data structure which handles certain use cases.

### ✓ **Resolve bottlenecks and follow-up questions (3-4 mins)**

# Below kind of diagrams are expected in OOD interviews



04

# API Design Interview



## Preparation resources

1. [Best Practices](#), [Implementation](#), and [Guidelines](#) to follow for API Design.
2. Look for use cases like - [Stripe](#) and [Twitter](#) API Documentation.
3. [SystemsExpert](#) also has a few case studies on API design as well.
4. Follow the [link](#) to understand how apis should be designed.

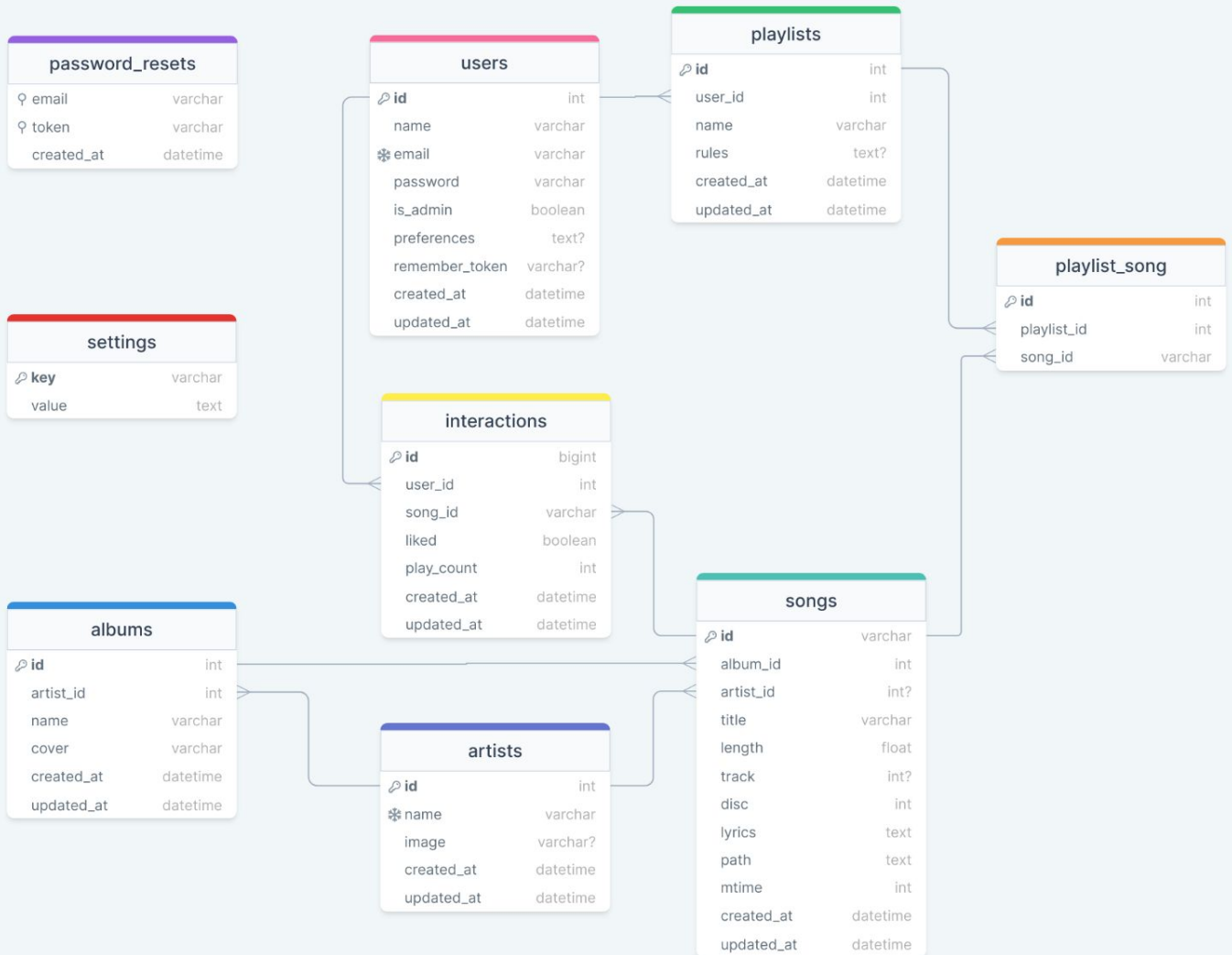
05

# Schema / Database Design Interview

## Preparation resources

1. [Grokking the Object Oriented Design Interview](#) - Take the case studies and try to apply Objects to the Relational Mapping strategy.
2. Try practicing drawing relational schema diagrams like - tables, columns, and relationships between tables. Try to use pen and paper or practice on [drawSQL](#)

# Below kind of diagrams are expected in Schema Design interviews



06

# Behavioral Interview

# Introduction

Behavioral Interviews are taken to evaluate a candidate's past behavior or conduct in certain different situations to analyze/predict their future behavior towards the future employer.

## Common Myths regarding Behavioral Interviews

1. **Are Simple Interviews** - Many candidates treat Behavioral Interviews to be easy and simple to crack. The reality is these interviews are often complex and very dense if not prepared well.
2. **Are just formality/no rejection interviews** - Many candidates believe that Behavioral Interviews are just for formality. The reality is these interviews can generate red flags based on your answers and can reject your whole candidature even if other interviews have gone well.
3. **Can be prepared in a day** - It is believed that Behavioral Interviews can be prepared a day before the actual interview. The reality is even though we know what questions can be asked, the complexity and rejection often come from the follow-up questions. Thus, sufficient time needs to be invested in its preparation.

## Preparation resources

1. Watch CareerVidz [YouTube channel](#) for behavioral interview questions.
2. Watch Jeff H Sipe's [YouTube channel](#) for behavioral questions.
3. Check out a list of good [Behavior Interview](#) questions.
4. [STAR](#) Pattern - Situation, Task, Action, Result
5. Prepare questions and career stories around [Amazon's leadership principles](#). It will cover every aspect of Behavior interviews.
6. Apply STAR pattern to write experience stories around various questions. Do not prepare for this kind of interview a day before. Keep the stories handy for all such interviews.

## Tips for Behavioral Interviews

1. There is no right or wrong answer. Try to focus on better and positive ways to answer them.
2. Apply the **STAR** pattern to write experience stories around the above behavioral questions.
3. Keep the stories handy for all such interviews. Write the question and describe stories following the **STAR** pattern.
4. If you don't have any such experience stories, try to come up with stories around your past life like school, colleague, events etc
5. If you don't have any such experience stories, try to come up with stories about your seniors, and colleagues and see what they have done when faced with such situations.
6. If you don't have any such experience for a question, you should tell that I don't have any such experience but I can tell you how would I will react in such situation.



## Tips for Behavioral Interviews

7. Avoid being disrespectful, arrogant, and confrontational to any person or company you have worked with.
8. Don't lie. Whatever you speak be ready with proof and explain things that looks genuine.
9. Avoid useless details, make stories/answer short and to the point.
10. Always highlight your good qualities in every sentence/paragraph of your answers.
11. Do not prepare for this kind of interview a day before.

# STAR Pattern to answer in a Behavioral Interviews

# STAR Method

The **STAR** method is a structured manner of responding to a behavioral interview question. The **STAR** method refers to **S**ituation, **T**ask, **A**ction, and **R**esult. Let's look it in more detail -

1. **Situation** - Describe the scene and provide the details about the situation or task you accomplished. The situation can be from past job, experience, any event or from personal life. The interviewer must understand the exact situation and its relevance to the question.
2. **Task** - Describe your responsibility in that situation or the goal you worked toward. It's like a list of steps that could help you accomplish the goal or situation.
3. **Action** - Describe the actions you took to complete each and every task. These actions should describe how much contribution you made towards accomplishing the situation.
4. **Result** - Share the outcomes of the tasks you accomplished by your actions. These results should demonstrate the impact of your actions and describes the lesson learned from them.

## Story Preparation

Take the [Amazon's leadership principles](#) and prepare 1-2 experience stories around it using the **STAR** method. It will cover most of your Behavioral Interview Questions. Let's take an example for one important principle - "**Customer Obsession**" -

1. **Situation** - I had just joined XYZ company and was assigned to a project that was already in the development phase for 6 months. The UI presented a screen to the Client for uploading documents. The UI showed all uploaded documents on UI, along with their metadata. Initially, it worked fine due to the limited no. of documents. As the documents increased in the system, this started making UI slower and slower. Soon, the problem became evident to the client and the team was asked about probable solutions.
2. **Task** - Looking at the problem I created a few tasks that can tackle the problem efficiently such as,
  - Improving the performance of UI. (Rendering of grid)
  - Improving the performance of backend logic.
  - Optimizing the queries that get us the data.
  - Handling and storage of the document blob efficiently (upload/download/store)
  - Improving the search, sorting and filters on UI.

## Story Preparation

3. **Action** - Taking up tasks step by step, we implemented various improvements such as, -
  - Implemented Client Side Pagination. Instead, of rendering all the documents on UI, we paginated it to show a max of 10-20 records. This made rendering of UI faster.
  - Implemented Server Side Pagination to make queries faster to fetch limited data.
  - Implemented Document blob fetching/preview on user action, instead of fetching along with document metadata. The idea was to show only document metadata and interested document preview was done via download API.
  - Implemented filters, search, and sorting along with paginated queries i.e. removed all the UI processing to the backend.
  - Initially, the Document blob was stored in a database. It resulted in slowness of queries. Later, we migrated it to a blob storage solution.
4. **Result** - After implementing the above actions step by step, it resulted in a drastic improvement in the performance of the system. The queries which took almost 1 minute to fetch the data of X no. of documents, now took nearly 1 sec. The clients were happy and it resulted in trust and new initiatives for the future.

07

# Resume Tips

- 👉 Provide a valid resume name for your file e.g. - Dinesh\_Varyani\_Resume.pdf. Avoid names such as MyResume.pdf, Resume.docx, DineshVaryani.docx, etc



Dinesh\_Varyani\_Resume



Resume



- 👉 Keep resume in multi-color format. Using single color will make important parts of your resume look the same. e.g. links/email become hard to recognize from normal text.

**AVERY DAVIS**  
Digital Designer

#### CONTACT

123-456-7890  
hello@reallygreatsite.com  
reallygreatsite.com

#### EDUCATION

**EDUCATIONAL QUALIFICATION**  
University  
2017-2020

**EDUCATIONAL QUALIFICATION**  
High School  
2010 - 2016

#### SKILLS

Content Writing  
Web Design  
Illustration  
Graphic Design

#### REFERENCES

Provided upon request.

#### SUMMARY

A passionate digital designer with over 10 years experience in online content, design and development. Known for my hardworking ethic and exceptional people skills.

#### WORK EXPERIENCE

**WEB DESIGNER**  
Really Great Company  
2020 - Present

- Website design and development
- Delivered 20 websites from concept to completion
- Developed visual style guide

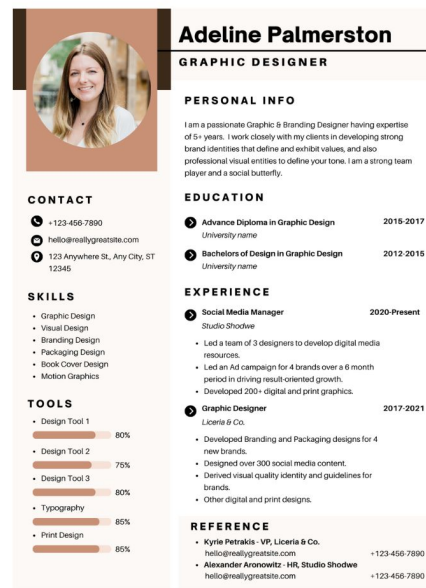
**WEB DEVELOPER**  
Really Great Company  
2014 - 2016

- Database administration and IT management
- Website back ups and server migration
- Front end and back end development

#### CONTENT MANAGER

**Really Great Company**  
2010 - 2016

- Content writing and editing
- Developed and maintained editorial content plan
- Created and trained staff on editorial style guide



**Adeline Palmerston**

**GRAPHIC DESIGNER**

#### PERSONAL INFO

I am a passionate Graphic & Branding Designer having expertise of 5+ years. I work closely with my clients in developing strong brand identities that define and exhibit values, and also professional visual identities to define your tone. I am a strong team player and a social butterfly.

#### EDUCATION

• **Advance Diploma in Graphic Design** 2015-2017  
University name

• **Bachelors of Design in Graphic Design** 2012-2015  
University name

#### EXPERIENCE

• **Social Media Manager** 2020-Present  
Studio Shodwe

- Led a team of 3 designers to develop digital media resources.
- Led an ad campaign for 4 brands over a 6 month period in driving result-oriented growth.
- Developed 200+ digital and print graphics.

• **Graphic Designer** 2017-2021  
Licerta & Co.

- Developed Branding and Packaging designs for 4 new brands.
- Designed over 300 social media content.
- Derived visual quality identity and guidelines for brands.
- Other digital and print designs.

#### REFERENCE

• **Kylie Petrakis - VP, Licerta & Co.** +123-456-7890  
hello@reallygreatsite.com

• **Alexander Anonowitz - HR, Studio Shodwe** +123-456-7890  
hello@reallygreatsite.com



- 👉 Keeping a One-page resume so that it's easy for the recruiter to go through it in 5-10 secs is not always correct. If you have done great things/amazing projects in your professional career you can showcase it in a 2-3 page resume.
- 👉 Showcase your skills/technologies in one place rather than scattered with every project you have worked on.

## TECHNICAL SKILLS

---

Core Java Spring Boot Micronaut JPA/Hibernate Restful webservices API Development/Integration

Spring Cloud Microservices HTML CSS Javascript React Angular Design Patterns Data Structures

Algorithms GraphQL Apollo Oracle Software Development Agile Methodology

- 👉 The resume header is the first thing the recruiter/manager will check out. Provide a good introductory summary covering your passion and ambitions at the top.

## SUMMARY

---

Dynamic and creative software developer with over 11 years of experience in software development. With proven leadership and development skills, have delivered complex systems to clients. In the overall experience, have been consistently rewarded in delivering technology-based solutions. Would like to give my best to my organization and be an important asset in an environment that provides me immense opportunities to learn, grow and enhance my skills and knowledge to the fullest.

- 👉 Keep the most important things in the resume first, like the introduction summary, technical skills, projects, achievements, certifications, education, etc. These top sections will be the first thing recruiters will check out. Things like hobbies, references, and contact info can come at the end



- 👉 Provide project impact via numbers and percentages rather than normal description.
- **Avoid** - Implemented the project with XYZ feature that helped in the productivity of the users.
  - **Focus** - Implemented the project with XYZ feature that increased the productivity of the users by 30%.
  - **Focus** - Implemented XYZ feature that reduced the manual time from 3 weeks to 10 mins.
  - **Focus** - Reduced the client onboarding duration from 1 month to 1 week etc.

Thus, projects outcome in numbers and percentages shows how much impact your deliverables have done in Company's growth.

- Designed the project which included the critical process of onboarding a Highend client. It spanned a complex workflow of gathering information of client's source of wealth. The system eased the process and has reduced client onboarding from 120+ days to less than 30 days.
- Responsible for requirement gathering from the legacy system and challenging businesses to adapt certain requirements to the new system.
- Responsible for overcoming issues of the legacy system. It was slow and provided locks while capturing the SOW of any partner. The new system was designed and implemented to overcome slowness and performance issues.
- The project reduced the timeframe of maker/checker communication to 60% and thus, increased the productivity of RMs and middle office teams by 50%.
- Led the end-to-end product development, deployment, monitoring, onboarding of Banking partners, and ensuring the quality of service of the product.

👉 Day-to-day work is ok and being done by everyone. But, an actual resume looks good when you showcase contributions outside work in a Company. Always highlight the things you do apart from normal work like - Training, Events, Hackathons, Team bonding activities KT, Sessions, etc

👉 Big Tech Giants will always look for innovative and creative work you have done in your career. Thus, showcasing the innovative things you have done throughout your career like working on cool personal projects, YouTube, Blogging, Instructor, Tools, Building Apps, Coding Platforms Rank etc

## TECHNICAL ABILITIES AND CONTRIBUTIONS

---

- Creator and owner of <http://www.hubberspot.com/> - a site containing blogs for Java learners and enthusiasts. It's thrice referred by the Spring Source Community for Spring Tutorials.
- Creator and owner of Youtube channel for technical videos by name - "[Dinesh Varyani](#)".
- Instructor on Udemy as [Dinesh Varyani](#) with courses on - Data Structures and Algorithms, JUnit 5 and Mockito 3.
- Created a tool named "Crud Builder" for Credit Suisse. The tool took DB tables metadata and generated code from Controller, Service, Repositories to Entities. The tool reduced the development time of developers by autogenerating code across all the multi-tier applications.
- Created a tool for QA's and Developers in Mastercard to directly configure Swagger within Postman to ease testing and debugging across various environments.
- Created a tool named "Log Extractor" for Mastercard. The tool presented UI to the developer and extracted the logs with one click from the configured environment.
- Created a tool named "Report Configuration Console" for ADP. The tool presented UI to the developer to configure reports related to the ADP Tax Credits Product. The tool configured reports in the database and also generated SQL for it. Thus, it saved a lot of development time for the team. It also had features to export SQL for all the reports in one go.
- Created a tool named "Product GUI Translation Tool" for Amdocs. This tool translated the GUI of Amdocs Product to various international languages. Initially, it was a manual process that took months to complete. This tool made it automatic and reduced overall development time to 20%.
- Working as a Java Trainer in Accenture and providing training to new joiners in technologies such as Java, Hibernate, JPA, and Spring Core.

👉 Adding your exceptional professional achievements or landmarks will make you standout among others. Thus, showcasing your achievements, certifications, awards, and recognitions is a must.

## ACHIEVEMENTS AND RECOGNITIONS

---

- Received **STAR Team Award (Credit Suisse)** in recognition of exceptional contribution to SOW Delivery.
- Received **STAR India IT Performance Award (Credit Suisse)** in recognition of outstanding performance, accomplishment, and lasting commitments to the organization.
- Received **Spot Award (Altimetrik/Mastercard)** in recognition of outstanding performance and splendid efforts at work.
- Received **Accenture Celebrates Excellence Award - Pathfinder** for contribution to Innovative Excellence, effective project delivery by using new and innovative methods, reusing existing best practices and work methodologies to work more efficiently
- Received **Accenture Celebrates Excellence Award - Numero Uno** for contribution to Delivery Excellence in the individual category, demonstrating a key sense of accountability and ownership for the processes that impact the quality, delivery timelines, and service level metrics.
- Received **Zenith Propel Award** for demonstrating the quality of work in the Accenture Financial Services domain and having a significant impact on Accenture Success.
- Received prestigious awards through Accenture Recognize Performance Program.
- Received consistently **Top ratings** across different organizations for outstanding performance.

## CERTIFICATES

---

- Java Standard Edition 6 Programmer Certified Professional.
- System Design certification from Educative.
- Object Oriented Design certification from Educative.
- Data Structures and Algorithms certification from Algoexpert.
- Software Olympiad 2015.
- Won Altimetrik Cricket Tournament.
- Quiz competition winner.

👉 Choose a colorful resume template that can accommodate the above tips. Websites like [hiration.com](https://www.hiration.com) provide many such templates. It has one free template that fits to what a good resume should have.

08

# Preparation Strategy

- 👉 Try to solve at least 1 medium / 2 easy-level coding question(s) every day.
- 👉 Try to solve problems on your own with no help. Look for hints if provided by the coding platform.
- 👉 The more time you spent on the problem (solving it on your own) the more concepts will become strong.
- 👉 After spending over an hour if you don't get the solution, look out for the solution, write it down on paper and make notes of things you missed.
- 👉 Revision is the key. Revise the concepts, notes, and problems often.
- 👉 Try to prepare at least 1 System and 1 Object Oriented Design case studies every week.
- 👉 Consistency is the key (You break, You fail)
- 👉 Apply [Pomodoro Technique](#) (Plan, 25 mins of focused prep, 5 mins of break, repeat)
- 👉 Give equal importance to Behavioral Interviews as well.

09

# Effective LeetCode

## Tips for using LeetCode effectively

- 👉 Solving problems in Quantity won't give you Quality preparation. Follow a roadmap of Quality problems - [100 Days to GAMAM](#).
- 👉 Solve LeetCode problems daily without breaking Consistency. Make it 1-2 hours per day. Remember the motto - "you break, you fail"
- 👉 Solve LeetCode curated list of [Top Interview Questions](#) (Very Important Questions)



**Top Interview Questions**  
LeetCode

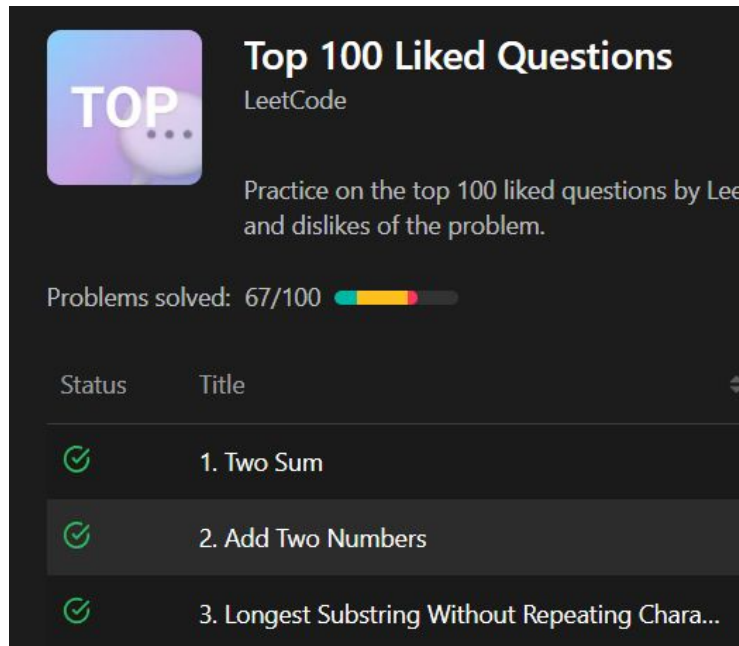
LeetCode is the golden standard for tech companies like Facebook, Hulu and Google.

Problems solved: 85/145

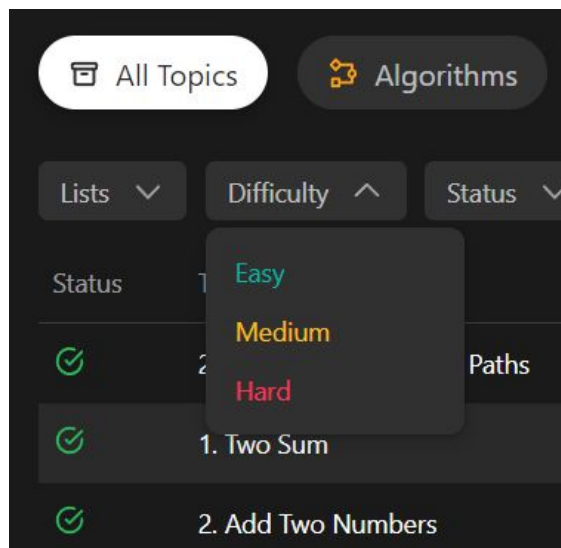
Status	Title
✓	1. Two Sum
✓	2. Add Two Numbers

## Tips for using LeetCode effectively

- 👉 Solve LeetCode curated list of [Top 100 Liked Questions](#) (Very Important Questions)



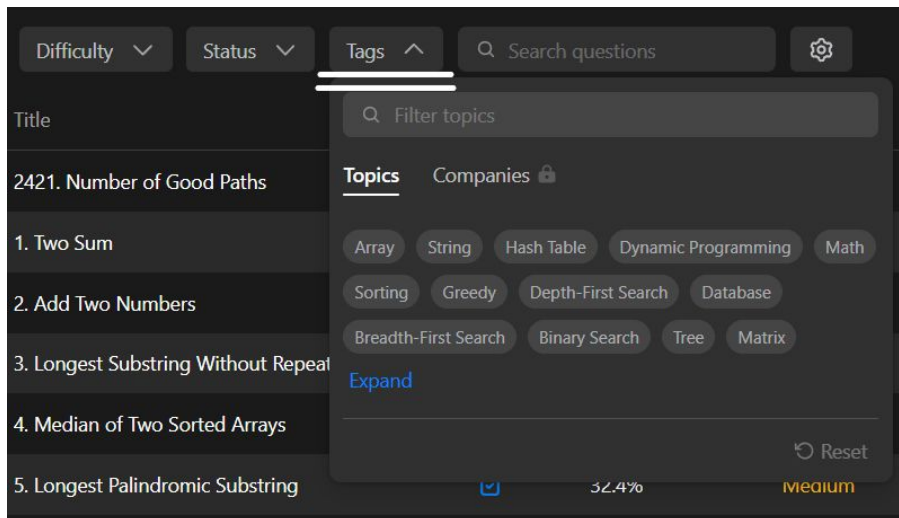
- 👉 If you are a beginner in coding always try to solve "Easy" level problems first.





## Tips for using LeetCode effectively

- Practice Questions from the topics in which you are weak and need confidence in logic building. Use Tags filter to choose specific topics questions.



- After Choosing any difficulty level, it's better to sort the problems by "Acceptance". This increases chances of successful attempt.

Lists

Difficulty

Status

Tags

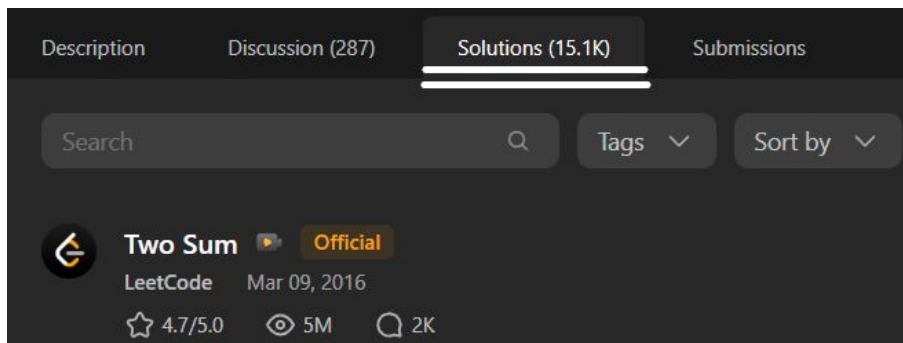
Q

Search questions

Status	Title	Solution	Acceptance
✓	2421. Number of Good Paths		57.9%
✓	1. Two Sum		49.3%

## Tips for using LeetCode effectively

- 👉 If you are a beginner, Choose a topic/easy-level questions and solve at least 25 problems in that topic to reach its Medium level.
- 👉 If you are **not** a beginner, Choose a topic/medium-level questions & solve at least 30 problems in that topic to reach its Hard level.
- 👉 Always check out multiple ways of solving the same problem. Checkout the **Solutions** Tab.



- 👉 Always look for Time & Space Complexity after solving the problem in the **Discussions** and **Solutions** Tab.



# Tips for using LeetCode effectively

- 👉 Target to solve Quality problems with at least -
  - ✓ 100 Easy Level
  - ✓ 250 Medium Level
  - ✓ 75 Hard Level
- 👉 Don't spend more than 45-60 minutes, if you are stuck in a problem. After above time limit -
  - ✓ Check Hints
  - ✓ Check Solutions
- 👉 If you are checking out the Solutions of others. Make sure you understand the solution, write down the solution in a paper, code it in LeetCode editor, and dry run.
- 👉 If you are checking out the Solutions of others. Make sure you write the notes around what you missed under **Notes** tab.

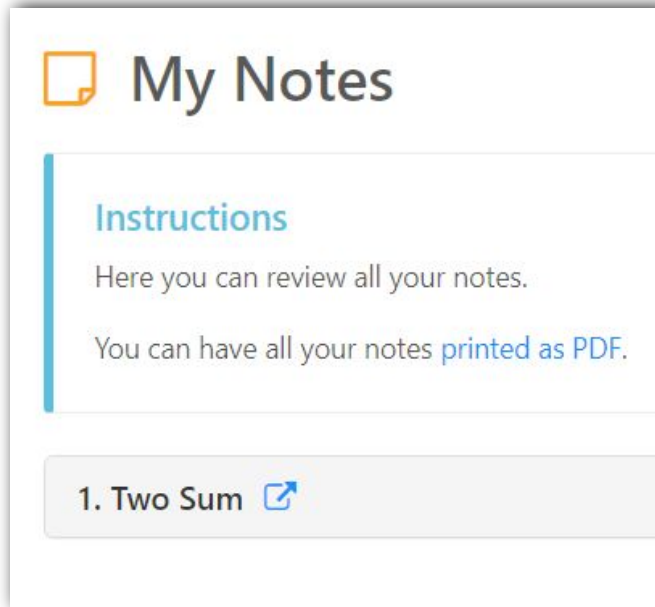
```
(int[] nums, int target) {  
    new int[2];  
    Integer> map = new HashMap<>()  
  
    < nums.length; i++){  
        containsKey(nums[i])){  
            target - nums[i], i)  
  
        } = map.get(nums[i]);  
        i = i;
```

```
1 class Solution {  
2     public int[] twoSum(int[] nums, int target) {  
3         int[] result = new int[2];  
4         HashMap<Integer, Integer> map = new HashMap<>();  
5         for(int i = 0; i < nums.length; i++){  
6             if(!map.containsKey(nums[i])){  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  

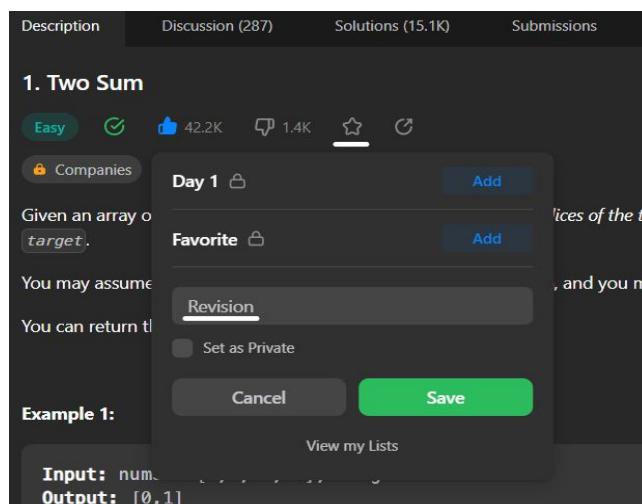
```

## Tips for using LeetCode effectively

- 👉 Notes you write in the **Notes** tab is very useful and LeetCode provides a way (**Notebook**) to print it for personal use and revision.

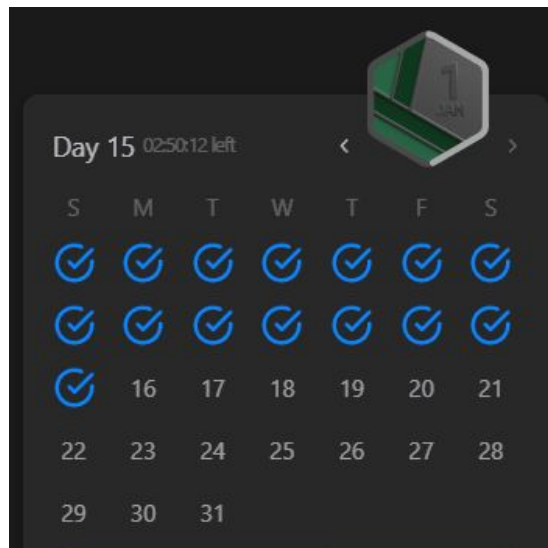


- 👉 Revision is a must. If you are checking out the Solutions of others. Make sure you are adding that problem in a Revision List. You can create custom list via LeetCode editor.

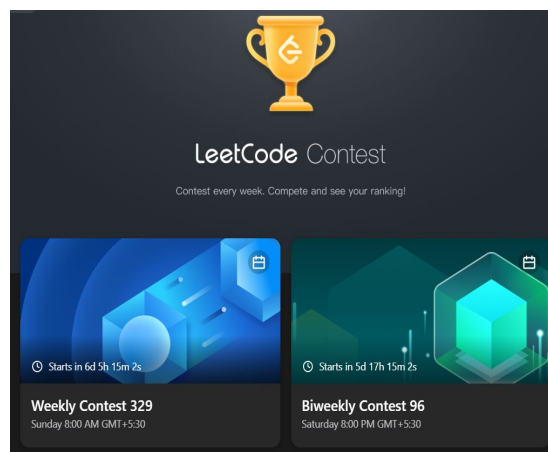


## Tips for using LeetCode effectively

- 👉 Revision is a must. As per Ebbinghaus's Forgetting Curve, keep revising the problems on which you faced difficulty often. It should be revised in span of 1st day, 7th day, 30th day.
- 👉 Once you are comfortable with LeetCode practice and have solved over 200+ problems, try to solve Daily LeetCode challenge.



- 👉 Once you have solved over 150+ problems, try to participate in Weekly/Biweekly Contest using **Contest** tab.



## Tips for using LeetCode effectively

- 👉 Always keep track of your progress by creating a new session using **Session Management** tab.

### Session Management

#### Instructions

To make a fresh start, **create** a new session and click on the new session to **activate** it.

Your progress will be tracked separately in the newly created session and you can [start another practicing round](#).

Create

- 👉 Keep looking for articles posted on the homepage of LeetCode. It covers detailed solutions of a given problem.

A

2 days ago

The [Shortest Path with Alternating Colors solution](#) has been published

You are given an integer  $n$ , the number of nodes in a directed graph where the nodes are labeled from 0 to  $n - 1$ . Each edge is red or...

A

3 days ago

The [As Far from Land as Possible solution](#) has been published

Given an  $n \times n$  grid containing only values 0 and 1, where 0 represents water and 1 represents land, find a water cell such that its distance to the nearest land cell...

A

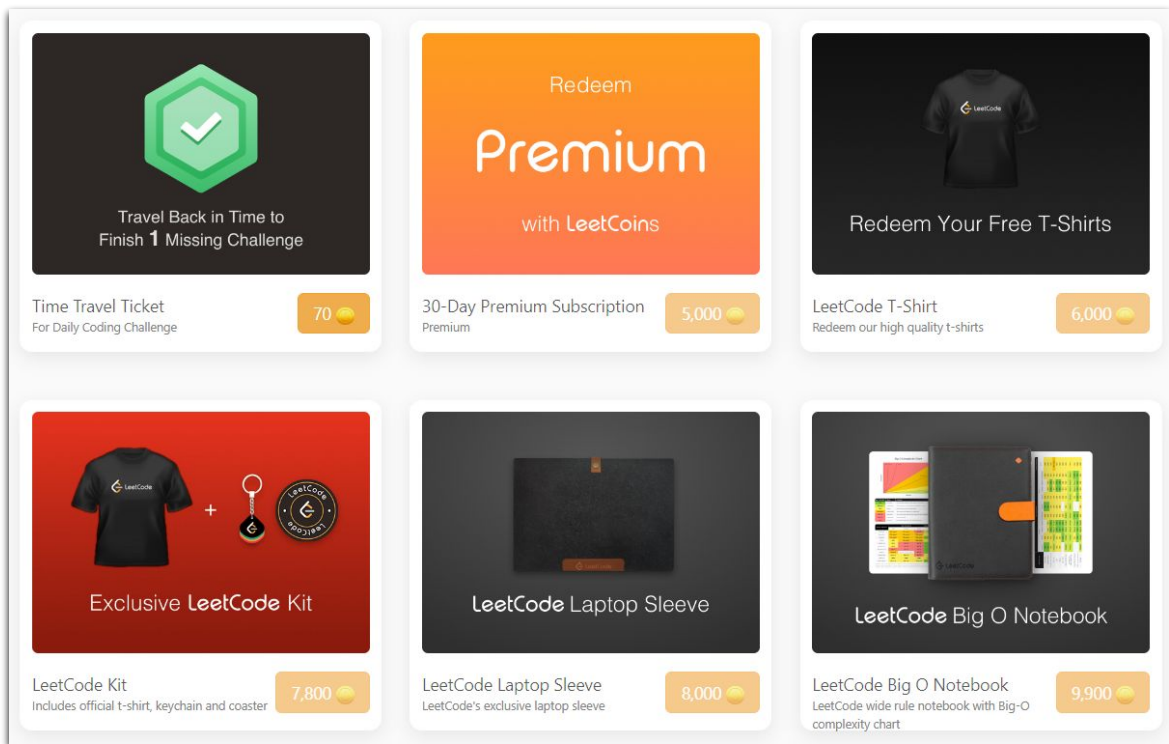
4 days ago

The [Edit Distance solution](#) has been published

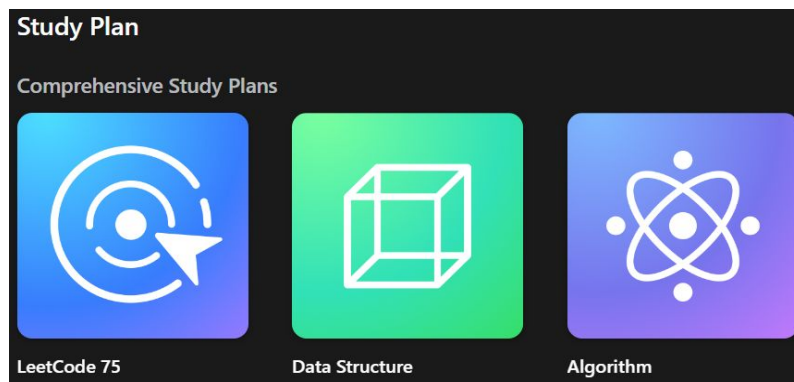
Given two strings `word1` and `word2`, return the minimum number of operations required to convert `word1` to `word2`. You have the following three operations permitted on a word: Insert a character Delete a...

## Tips for using LeetCode effectively

- 👉 Motivation Tip - Keep checking in daily, solve the Daily LeetCode challenge, participate in Contest to earn LeetCode points, and redeem gifts.



- 👉 Try to complete free study plans and assessments those are great and quick resources to master DSA.



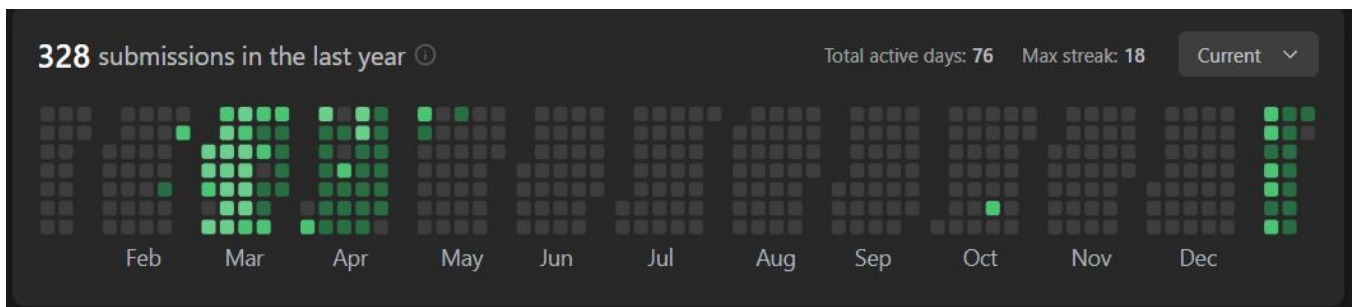


## Tips for using LeetCode effectively

- 👉 Track your LeetCode rank every day. It will motivate you to improve it more by solving problems daily.



- 👉 Track your consistency via the LeetCode submission graph. It shows the total active days, max streak, and number of submissions in the last year.





10

# 150 Days Roadmap to GAMAM

# 150 Days to Roadmap to GAMAM

- 👉 Coding questions are from LeetCode and in order (Easy, Medium, Hard)
- 👉 Every 6 days have coding questions.
- 👉 Every 7th day have one system design and one low level design questions.
- 👉 System design and Low-level design questions are taken from the resources mentioned in previous sections.
- 👉 Every 15th day is a revision day for things practiced in previous 14 days.
- 👉 Last section starting 120th day has behavioral interview questions as well.
- 👉 Choose days and questions based on your time and availability.
- 👉 If you miss any question on a particular day, just carry it over to next day.
- 👉 Idea is to be consistent for 150 days and not to solve all questions in hurry.

## DAY 1

- Two Sum
- Best Time to Buy and Sell Stock
- Majority Element
- Move Zeroes
- Squares of a Sorted Array
- Merge Sorted Array

## DAY 2

- Remove Duplicates from Sorted Array
- Remove Duplicates from Sorted Array II
- Find All Numbers Disappeared in an Array
- Intersection of Two Arrays
- Intersection of Two Arrays II
- Maximum Population Year
- Find Pivot Index

## DAY 3

- Running Sum of 1d Array
- Remove Element
- Find Winner on a Tic Tac Toe Game
- Build Array from Permutation
- Third Maximum Number
- Valid Mountain Array

## DAY 4

- Find Common Characters
- Sum of All Odd Length Subarrays
- Range Sum Query - Immutable
- Shuffle the Array
- Max Consecutive Ones
- Sort Array By Parity

## DAY 5

- Reverse Linked List
- Remove Linked List Elements
- Remove Duplicates from Sorted List
- Merge Two Sorted Lists
- Middle of the Linked List
- Palindrome Linked List

## DAY 6

- Intersection of Two Linked Lists
- Linked List Cycle
- Valid Parentheses
- Implement Queue using Stacks
- Backspace String Compare
- Next Greater Element I

## DAY 7

- Design a Rate Limiter (System Design)
- Design a Library Management System (OOD Design)

## DAY 8

- Binary Tree Preorder Traversal
- Binary Tree Inorder Traversal
- Binary Tree Postorder Traversal
- Maximum Depth of Binary Tree
- Invert Binary Tree
- Symmetric Tree

## DAY 9

- Subtree of Another Tree
- Diameter of Binary Tree
- Balanced Binary Tree
- Merge Two Binary Trees
- Same Tree

## DAY 10

- Path Sum
- Binary Tree Paths
- Cousins in Binary Tree
- Convert Sorted Array to Binary Search Tree
- Range Sum of BST

## DAY 11

- Valid Palindrome
- Valid Palindrome II
- Longest Palindrome
- Longest Common Prefix
- Valid Anagram
- First Unique Character in a String

## DAY 12

- Is Subsequence
- Reverse String
- Reverse String II
- Reverse Words in a String III
- Isomorphic Strings
- Remove All Adjacent Duplicates In String

## DAY 13

- Defanging an IP Address
- Reverse Only Letters
- Reverse Vowels of a String
- Length of Last Word
- Add Strings
- Fizz Buzz

## DAY 14

- Design Consistent Hashing (System Design)
- Design a Parking Lot (OOD Design)

## DAY 15

- Revise 1-14 days

## DAY 16

- Roman to Integer
- Palindrome Number
- Happy Number
- Power of Two
- Sqrt(x)
- Plus One

## DAY 17

- Count Odd Numbers in an Interval Range
- Rectangle Overlap
- Add Digits
- Maximum Product of Three Numbers
- Excel Sheet Column Number

## DAY 18

- Add Binary
- Counting Bits
- Number of 1 Bits
- Single Number
- Missing Number
- Reverse Bits
- Hamming Distance

## DAY 19

- Binary Search
- Search Insert Position
- First Bad Version
- Valid Perfect Square
- Kth Missing Positive Number
- Kth Largest Element in a Stream

## DAY 20

- Design HashMap
- Ransom Note
- Contains Duplicate
- Contains Duplicate II
- Jewels and Stones
- Unique Number of Occurrences

## DAY 21

- Word Pattern
- Number of Good Pairs
- Flood Fill
- Island Perimeter
- Find if Path Exists in Graph



## DAY 22

- Design A Key-value Store (System Design)
- Design Amazon - Online Shopping System (OOD Design)

## DAY 23

- Fibonacci Number
- Min Cost Climbing Stairs
- Climbing Stairs
- Pascal's Triangle
- Can Place Flowers
- Maximum Units on a Truck

## DAY 24

- 3Sum
- 3Sum Closest
- Non-decreasing Array
- Product of Array Except Self

## DAY 25

- Merge Intervals
- Insert Interval
- Non-overlapping Intervals
- Interval List Intersections

## DAY 26

- Container With Most Water
- Sort Colors
- Rotate Array
- Contiguous Array

## DAY 27

- Subarray Sum Equals K
- Shortest Unsorted Continuous Subarray
- Maximum Points You Can Obtain from Cards
- Max Consecutive Ones III

## DAY 28

- Permutation in String
- Wiggle Sort II
- Max Chunks To Make Sorted
- H-Index

## DAY 29

- Design A Distributed Unique ID Generator (System Design)
- Design Stack Overflow (OOD Design)

## DAY 30

- Revise 16-29 days

## DAY 31

- Remove Nth Node From End of List
- Delete Node in a Linked List
- Remove Duplicates from Sorted List II
- Next Greater Node In Linked List

## DAY 32

- Add Two Numbers
- Add Two Numbers II
- Copy List with Random Pointer
- Reverse Linked List II

## DAY 33

- Swap Nodes in Pairs
- Odd Even Linked List
- Partition List

## DAY 34

- Sort List
- Reorder List
- Rotate List

## DAY 35

- Evaluate Reverse Polish Notation
- Min Stack
- Daily Temperatures
- Decode String

## DAY 36

- Next Greater Element II
- Next Greater Element III
- Minimum Remove to Make Valid Parentheses
- 132 Pattern

## DAY 37

- Design A URL Shortener (System Design)
- Design a Movie Ticket Booking System (OOD Design)

## DAY 38

- Asteroid Collision
- Basic Calculator II
- Remove K Digits
- Remove Duplicate Letters

## DAY 39

- Remove All Adjacent Duplicates in String II
- Flatten Nested List Iterator
- Simplify Path
- Longest Absolute File Path

## DAY 40

- Open the Lock
- Shortest Bridge
- LRU Cache

## DAY 41

- Longest Substring Without Repeating Characters
- String to Integer (atoi)
- Find All Anagrams in a String
- Group Anagrams
- Pancake Sorting

## DAY 42

- Longest Repeating Character Replacement
- Largest Number
- Number of Matching Subsequences
- Find the Index of the First Occurrence in a String

## DAY 43

- Longest Substring with At Least K Repeating Characters
- Zigzag Conversion
- Reverse Words in a String
- String Compression
- Count and Say

## DAY 44

- Design Pastebin (System Design)
- Design an ATM (OOD Design)

## DAY 45

- Revise 31-44 days

## DAY 46

- Binary Tree Level Order Traversal
- Binary Tree Zigzag Level Order Traversal
- Construct Binary Tree from Preorder and Inorder Traversal
- Lowest Common Ancestor of a Binary Tree

## DAY 47

- Binary Tree Right Side View
- Populating Next Right Pointers in Each Node
- Populating Next Right Pointers in Each Node II
- Maximum Width of Binary Tree

## DAY 48

- Path Sum II
- Path Sum III
- All Nodes Distance K in Binary Tree
- Flatten Binary Tree to Linked List

## DAY 49

- Count Complete Tree Nodes
- Sum Root to Leaf Numbers
- Find Bottom Left Tree Value
- Distribute Coins in Binary Tree

## DAY 50

- Delete Node in a BST
- Validate Binary Search Tree
- Kth Smallest Element in a BST
- Lowest Common Ancestor of a Binary Search Tree

## DAY 51

- Convert Sorted List to Binary Search Tree
- Construct Binary Search Tree from Preorder Traversal
- Binary Search Tree Iterator
- Recover Binary Search Tree

## DAY 52

- Design Instagram (System Design)
- Design an Airline Management System (OOD Design)

## DAY 53

- Binary Tree Maximum Path Sum
- Step-By-Step Directions From a Binary Tree Node to Another
- Maximum Level Sum of a Binary Tree

## DAY 54

- Trim a Binary Search Tree
- Balance a Binary Search Tree
- Serialize and Deserialize Binary Tree

## DAY 55

- Search in Rotated Sorted Array
- Search in Rotated Sorted Array II
- Time Based Key-Value Store
- Find Minimum in Rotated Sorted Array

## DAY 56

- Find First and Last Position of Element in Sorted Array
- Find the Duplicate Number
- Minimum Size Subarray Sum
- Single Element in a Sorted Array

## DAY 57

- Find Peak Element
- Capacity To Ship Packages Within D Days
- Koko Eating Bananas
- Peak Index in a Mountain Array

## DAY 58

- Search a 2D Matrix
- Search a 2D Matrix II
- Spiral Matrix
- Spiral Matrix II

## DAY 59

- Design A Web Crawler (System Design)
- Design Blackjack and a Deck of Cards (OOD Design)



## DAY 60

- Revise 46-59 days

## DAY 61

- Valid Sudoku
- Rotate Image
- Set Matrix Zeroes
- Game of Life

## DAY 62

- Diagonal Traverse
- Matrix Block Sum
- Battleships in a Board
- Snapshot Array

## DAY 63

- Number of Islands
- 01 Matrix
- Clone Graph
- Rotting Oranges

## DAY 64

- Course Schedule
- Course Schedule II
- Accounts Merge
- Word Search

## DAY 65

- Minimum Height Trees
- Pacific Atlantic Water Flow
- Cheapest Flights Within K Stops
- Max Area of Island

## DAY 66

- Evaluate Division
- Number of Provinces
- Surrounded Regions
- Network Delay Time

## DAY 67

- Design A Notification System (System Design)
- Design a Hotel Management System (OOD Design)

## DAY 68

- All Paths From Source to Target
- Redundant Connection
- Shortest Path in Binary Matrix
- Number of Operations to Make Network Connected

## DAY 69

- Majority Element II
- Longest Consecutive Sequence
- Insert Delete GetRandom O(1)
- Find All Duplicates in an Array

## DAY 70

- Continuous Subarray Sum
- Find and Replace Pattern
- K-diff Pairs in an Array
- Custom Sort String

## DAY 71

- Fraction to Recurring Decimal
- Fruit Into Baskets
- Encode and Decode TinyURL
- Minimum Area Rectangle

## DAY 72

- Maximum Subarray
- Maximum Product Subarray
- Coin Change
- Coin Change II

## DAY 73

- Jump Game
- Jump Game II
- Jump Game III
- Partition Equal Subset Sum

## DAY 74

- Design A News Feed System (System Design)
- Design a Restaurant Management system (OOD Design)

## DAY 75

- Revise 61-74 days

## DAY 76

- Longest Increasing Subsequence
- Unique Paths
- Unique Paths II
- Maximal Square

## DAY 77

- House Robber
- House Robber II
- House Robber III
- Decode Ways

## DAY 78

- Best Time to Buy and Sell Stock II
- Minimum Path Sum
- Longest Common Subsequence
- Palindrome Partitioning

## DAY 79

- Unique Binary Search Trees
- Unique Binary Search Trees II
- Target Sum
- Triangle

## DAY 80

- Longest Palindromic Subsequence
- Partition to K Equal Sum Subsets
- Delete and Earn
- Palindromic Substrings

## DAY 81

- Longest String Chain
- Minimum Cost For Tickets
- Delete Operation for Two Strings
- Perfect Squares

## DAY 82

- Design A Chat System (System Design)
- Design Chess (OOD Design)

## DAY 83

- Different Ways to Add Parentheses
- Longest Palindromic Substring
- Largest Divisible Subset
- Integer Break

## DAY 84

- Matchsticks to Square
- Knight Dialer
- Minesweeper

## DAY 85

- Random Pick with Weight
- Pow(x, n)
- Reverse Integer
- Multiply Strings

## DAY 86

- Count Primes
- Integer to Roman
- Robot Bounded In Circle
- Angle Between Hands of a Clock

## DAY 87

- K Closest Points to Origin
- Task Scheduler
- Top K Frequent Elements
- Find K Closest Elements

## DAY 88

- Kth Largest Element in an Array
- Kth Smallest Element in a Sorted Matrix
- Top K Frequent Words
- Reorganize String

## DAY 89

- Design A Search Autocomplete System (System Design)
- Design an Online Stock Brokerage System (OOD Design)

## DAY 90

- Revise 76-89 days

## DAY 91

- Sort Characters By Frequency
- Car Pooling
- Find K Pairs with Smallest Sums
- Maximum Number of Events That Can Be Attended

## DAY 92

- Implement Trie (Prefix Tree)
- Word Break
- Design Add and Search Words Data Structure
- Search Suggestions System
- Remove Sub-Folders from the Filesystem

## DAY 93

- Permutations
- Permutations II
- Subsets
- Subsets II

## DAY 94

- Next Permutation
- Combinations
- Letter Combinations of a Phone Number
- Generate Parentheses

## DAY 95

- Combination Sum
- Combination Sum III
- Combination Sum IV
- Restore IP Addresses

## DAY 96

- Gas Station
- Partition Labels
- Valid Parenthesis String
- Minimum Number of Arrows to Burst Balloons

## DAY 97

- Design YouTube (System Design)
- Design a Car Rental System (OOD Design)

## DAY 98

- Single Number II
- Single Number III
- Maximum XOR of Two Numbers in an Array
- Divide Two Integers

## DAY 99

- Sum of Two Integers
- Bitwise AND of Numbers Range
- Gray Code



## DAY 100

- Sliding Window Maximum
- Trapping Rain Water
- Count of Smaller Numbers After Self

## DAY 101

- Candy
- Reverse Pairs
- Subarrays with K Different Integers
- Number of Submatrices That Sum to Target

## DAY 102

- Shortest Subarray with Sum at Least K
- Maximum Gap
- First Missing Positive

## DAY 103

- Shuffle an Array
- Reverse Nodes in k-Group
- LFU Cache

## DAY 104

- Design Google Drive (System Design)
- Design LinkedIn (OOD Design)

## DAY 105

- Revise 91-104 days

## DAY 106

- Basic Calculator
- Largest Rectangle in Histogram
- Longest Valid Parentheses

## DAY 107

- Maximum Frequency Stack
- The Skyline Problem
- Minimum Window Substring

## DAY 108

- Palindrome Pairs
- Shortest Palindrome
- Text Justification

## DAY 109

- Nth Digit
- Integer to English Words
- Max Points on a Line

## DAY 110

- Maximum Profit in Job Scheduling
- Median of Two Sorted Arrays
- Find Minimum in Rotated Sorted Array II

## DAY 111

- Word Ladder
- Word Ladder II
- Longest Increasing Path in a Matrix

## DAY 112

- Design Twitter (System Design)
- Design Facebook - a social network (System Design, OOD Design)

## DAY 113

- Word Search II
- Bus Routes
- Critical Connections in a Network

## DAY 114

- Shortest Path in a Grid with Obstacles Elimination
- Reconstruct Itinerary
- Making A Large Island

## DAY 115

- Merge k Sorted Lists
- Find Median from Data Stream
- Smallest Range Covering Elements from K Lists

## DAY 116

- Minimum Number of Refueling Stops
- Swim in Rising Water
- Longest Duplicate Substring

## DAY 117

- N-Queens
- Permutation Sequence
- Sudoku Solver
- Palindrome Partitioning II

## DAY 118

- K-th Symbol in Grammar
- Remove Invalid Parentheses
- Unique Paths III

## DAY 119

- Proximity Service (System Design)
- Design Cricinfo (OOD Design)

## DAY 120

- Revise 106 - 119 days

## DAY 121

- What are your strengths and weaknesses?
- Tell me about your most challenging customer. How did you resolve their issues and make them satisfied?
- Describe a time when you had to make a decision without having all the data or information you needed.
- Which {company's} leadership principle resonates with you most?
- Tell me about a time when you were working on a project, and you realized that you needed to make changes to what you were doing. How did you feel about the work you had already completed?

## DAY 122

- Nearby Friends (System Design)
- Google Maps (System Design)

## DAY 123

- Edit Distance
- Regular Expression Matching
- Maximal Rectangle

## DAY 124

- Can you give me an example of a time when you exceeded expectations?
- Can you describe a time when you took the lead on a project?
- Think about a time you received negative feedback. How did you deal with that?
- Tell me about a time when you had to deal with ambiguity. How did you overcome the ambiguity to reach a positive outcome?
- Have you been stressed over a certain project delivery in the past? Did it affect your work-life balance? How did you deal with it?

## DAY 125

- Distributed Message Queue (System Design)
- Distributed Email Service (System Design)

## DAY 126

- Split Array Largest Sum
- Burst Balloons
- Wildcard Matching

## DAY 127

- Tell me about a time you have disagreed with your manager and how you handled it.
- How do you motivate others? Can you give me an example of a time you have motivated someone?
- Tell me about a time when you took a risk and failed. What did you learn from that experience?
- What obstacles have you encountered in your career? How did you overcome them?
- Tell me about a project you are proud of. How did you ensure high standards were met in delivering that project?

## DAY 128

- S3-like Object Storage (System Design)
- Real-time Gaming Leaderboard (System Design)

## DAY 129

- Best Time to Buy and Sell Stock IV
- Word Break II
- Russian Doll Envelopes
- Validate Stack Sequences

## DAY 130

- Why do you want to work for {company}?
- Tell me about a time when you have had to work to earn someone's trust.
- Describe a time when you were given a project to work on, but your responsibilities were unclear. What did you do?
- Tell me about a time you showed initiative.
- You see a co-worker struggling with a task. What do you do?

## DAY 131

- Payment System (System Design)
- Digital Wallet (System Design)

## DAY 132

- Minimum Insertion Steps to Make a String Palindrome
- Minimum Cost to Cut a Stick
- Minimum Number of Taps to Open to Water a Garden
- Binary Tree Cameras

## DAY 133

- Describe for me a time when you had to choose short-term sacrifices to achieve long-term gains.
- How do you deal with having to provide feedback to someone?
- Tell me about a time you failed to meet a deadline. How did you cope with that?
- Has there been a time when your contribution was overlooked and somebody else from your team took credit for it? How did you deal with it?
- Tell me about a project you are proud of. How did you ensure high standards were met in delivering that project?
- Have you been in a conflict with a fellow coworker? How did you deal with it and what was the end result?

## DAY 134

- Design Uber backend (System Design)
- Design Ticketmaster (System Design, OOD Design)

## DAY 135

- Revise 121 - 134 days

## DAY 136 - 150 (Revise 1-135 days)

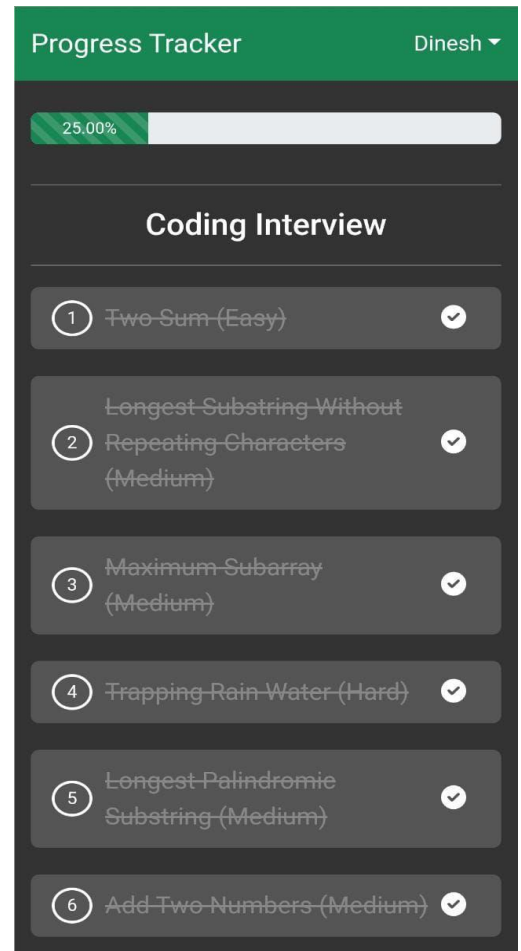


11

# GAMAM Progress Tracker

👉 Follow the below app for GAMAM-level companies preparation. It has the best resources which will cover various topics of interviews. The app will help you in keeping track of your progress for interview preparation.

👉 Follow the resources (links), understand the concepts behind them, and mark them complete. Once you reach over 80-100% the progress you are well prepared for the GAMAM-level company interviews.



<https://progress-tracker-5b3b0.web.app/>

- 📌 All the best for your preparation.
- 📌 If you find it useful, then follow [Dinesh Varyani](#) on LinkedIn.
- 📌 Subscribe to my [YouTube](#) channel.
- 📌 Credits - @Arslan Ahmed, @Clement Mihailescu, @Alex Xu.

# Thank you.

# CRACKING THE **GAMAM** TECHNICAL INTERVIEWS

## WHAT'S INSIDE?

The main objective of this book is to help you in preparing and crack **GAMAM** (Google, Apple, Microsoft, Amazon, Meta) technical interviews. The book covers preparation resources, strategies, tips, and a roadmap I followed that helped me clear Google/Amazon technical rounds. The book summarizes my journey like this -

- 👉 How I prepared myself for the GAMAM companies?
- 👉 What resources I used for various types of interviews?
- 👉 What strategies have I used to master different topics?
- 👉 What roadmap I followed in months of my preparation?
- 👉 Which resume tips got me in the recruiter's eye?
- 👉 How I tracked the progress of my preparation?
- 👉 My advice/tips on how to answer in a technical interview?



## Dinesh Varyani

I am working as a Cloud Engineer at Google. I am having over 12+ years of experience in Software Engineering. I am a passionate Youtuber, Blogger, Udemy & Educative Instructor, and now an Author.