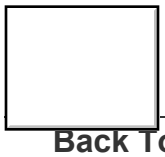



Back To Course Home

Grokking Modern System Design Interview for Engineers & Managers

0% completed

Search Course

System Design Interviews

Introduction

Abstractions

Non-functional System Characteristics

Back-of-the-envelope Calculations

Building Blocks

Domain Name System

Load Balancers

Databases

Key-value Store

☐

Content Delivery Network (CDN)

☐

Sequencer

☐

Distributed Monitoring

☐

Monitor Server-side Errors

☐

Monitor Client-side Errors

☐

Distributed Cache

☐

Distributed Messaging Queue

☐

Pub-sub

☐

Rate Limiter

☐

Blob Store

☐

Distributed Search

☐

Distributed Logging

☐

Distributed Task Scheduler

☐

Sharded Counters

☐

Concluding the Building Blocks Discussion

☐

Design YouTube

☐

Design Quora

☐

Design Google Maps

☐

Design a Proximity Service / Yelp

☐

Design Uber

☐

Design Twitter

☐

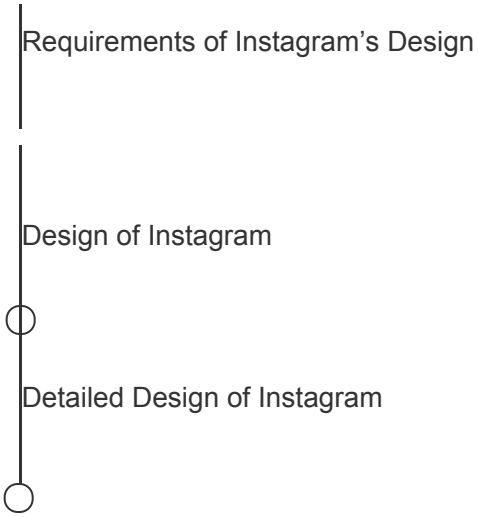
Design Newsfeed System

☐

Design Instagram

☐

System Design: Instagram



Design a URL Shortening Service / TinyURL

Design a Web Crawler

Design WhatsApp

Design Typeahead Suggestion

Design a Collaborative Document Editing Service / Google Docs

Spectacular Failures

Concluding Remarks

Course Certificate

Mark Course as Completed

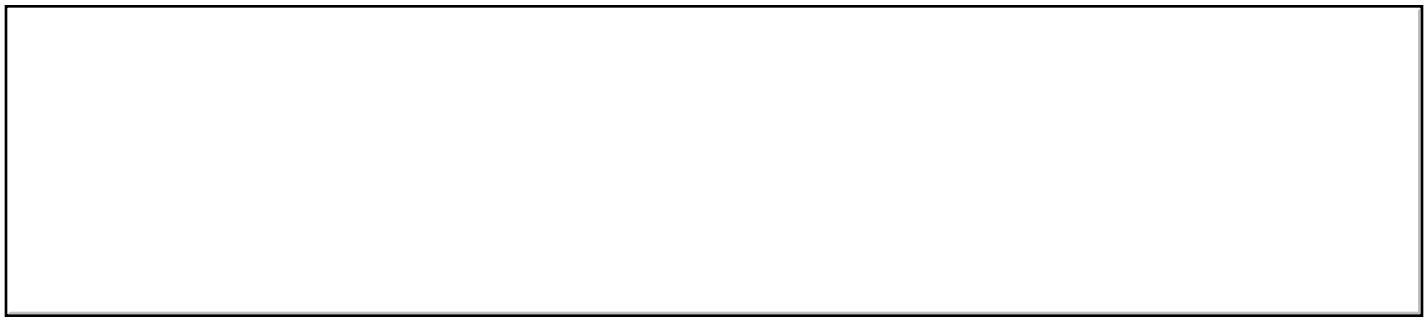
Design of Instagram

Learn the high-level design of Instagram and understand its data model.

We'll cover the following

High-level design#

Our system should allow us to upload, view, and search images and videos at a high level. To upload images and videos, we need to store them, and upon fetching, we need to retrieve the data from the storage. Moreover, the users should also be allowed to follow each other.



The high-level design of Instagram

API design#

This section describes APIs invoked by the users to perform different tasks (upload, like, and view photos/videos) on Instagram. We'll implement REST APIs for these tasks. Let's develop APIs for each of the following features:

- Post photos and videos
- Follow and unfollow users
- Like or dislike posts
- Search photos and videos
- Generate a news feed

All of the following calls will have a `userID`, that uniquely specifies the user performing the action. We'll only discuss new parameters in the calls.

Post photos or videos#

The POST method is used to post photos/videos to the server from the user through the `/postMedia` API. The `/postMedia` API is as follows:

```
postMedia(userID, media_type, list_of_hashtags, caption)
```

Parameter	Description
`media_type`	It indicates the type of media (photo or video) in a post.

`list_of_hashtags`	It represents all hashtags (maximum limit 30 hashtags) in a post.
`caption`	It is a text (maximum limit is 2,200 characters) in a user's post.

Follow and unfollow users#

The `/followUser` API is used when a user follows other users on Instagram. The `/followUser` API is as follows:

```
followUser(userID, target_userID)
```

Parameter	Description
`target_userID`	It indicates the user to be followed.

The `/unfollowUser` API uses the same parameters when a user unfollows someone on Instagram.

Like or dislike posts#

The `/likePost` API is used when users like someone's post on Instagram.

```
likePost(userID, target_userID, post_id)
```

Parameter	Description
`target_userID`	It specifies the user whose post is liked.
`post_id`	It specifies the post's unique ID.

The `/dislikePost` API uses the same parameters when a user dislikes someone's post on Instagram.

Search photos or videos#

The GET method is used when the user searches any photos or videos using a keyword or hashtag. The `/searchPhotos` API is as follows:

```
searchPhotos(userID, keyword)
```

Parameter	Description
`keyword`	It indicates the string (username, hashtag, and places) typed by the user in the search bar.

> **Note:** Instagram shows the posts with the highest reach (those with more likes and views) upon searching for a particular key. For example, if a user does a location-based search using “London, United Kingdom," Instagram will show the posts in order with maximum to minimum reach. Instead of showing all the posts, the data will be loaded upon scrolling.

Generate news feed#

The GET method is used when users view their news feed through the `/viewNewsfeed` API. The `/viewNewsfeed` API is as follows:

```
viewNewsfeed(userID, generate_timeline)
```

Parameter	Description
`generate_timeline`	It indicates the time when a user requests news feed generation. Instagram shows the posts that are not seen by the user between the last news feed request and the current news feed request.

Storage schema#

Let's define our data model now:

Relational or non-relational database#

It is essential to choose the right kind of database for our Instagram system, but which is the right choice — SQL or NoSQL? Our data is inherently relational, and we need an order for the data (posts should appear in chronological order) and no data loss even in case of failures (data durability). Moreover, in our case, we would benefit from relational queries like fetching the followers or images based on a user ID. Hence, SQL-based databases fulfill these requirements.

So, we'll opt for a relational database and store our relevant data in that database.

Define tables#

On a basic level, we need the following tables:

- **Users**: This stores all user-related data such as ID, name, email, bio, location, date of account creation, time of the last login, and so on.
- **Followers**: This stores the relations of users. In Instagram, we have a unidirectional relationship, for example, if user A accepts a follow request from user B, user B can view user A's post, but vice versa is not valid.
- **Photos**: This stores all photo-related information such as ID, location, caption, time of creation, and so on. We also need to keep the user ID to determine which photo belongs to which user. The user ID is a foreign key from the users table.
- **Videos**: This stores all video-related information such as ID, location, caption, time of creation, and so on. We also need to keep the user ID to determine which video belongs to which user. The user ID is a foreign key from the users table.

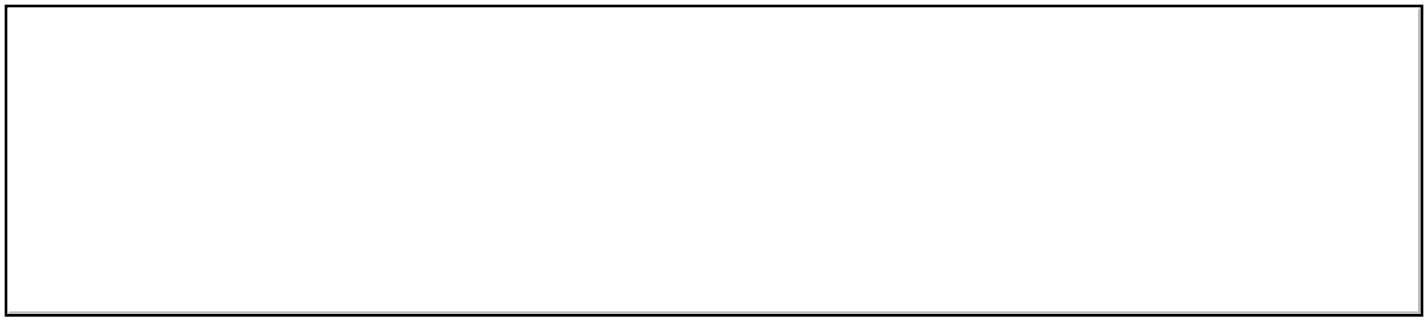
Point to Ponder

Question

Where should we store the photos and videos?

Show Answer

The following illustration visualizes the data model:



The Data model of Instagram

Data estimation#

Let’s figure out how much data each table will store. The column per row size (in bytes) in the following calculator shows the data per row of each table. It also calculates the storage needed for the specified counts. For example, the storage required for 500 million users is 111000 MB, 2000 MB for 250 followers of a single user, 23640 MB for 60 million photos, and 13790 MB for 35 million videos.

You can change the values in the calculator to observe the change in storage needed. This gives us an estimate of how fast data will increase in our tables.

			Storage Needed (in MB

Table Name	Per row size (in bytes)	Count in Millions	s)
Users	222	500	f 111000
Followers	8	250	f 2000
Photos	394	60	f 23640
Videos	394	35	f 13790

> **Note:** Most modern services use both SQL and NoSQL stores. Instagram officially uses a combination of SQL (PostgreSQL) and No-SQL (Cassandra) databases. The loosely structured data like timeline generation is usually stored in No-SQL, while relational data is saved in SQL-based storage.

In the next lesson, we'll identify more components to tweak our design.

Requirements of Instagram's Design

Next

Detailed Design of Instagram

Mark as Completed

Report an Issue