Member-only story

System Design Blueprint: The Ultimate Guide

Love Sharma · Follow

Published in ByteByteGo System Design Alliance · 9 min read · Apr 20, 2023

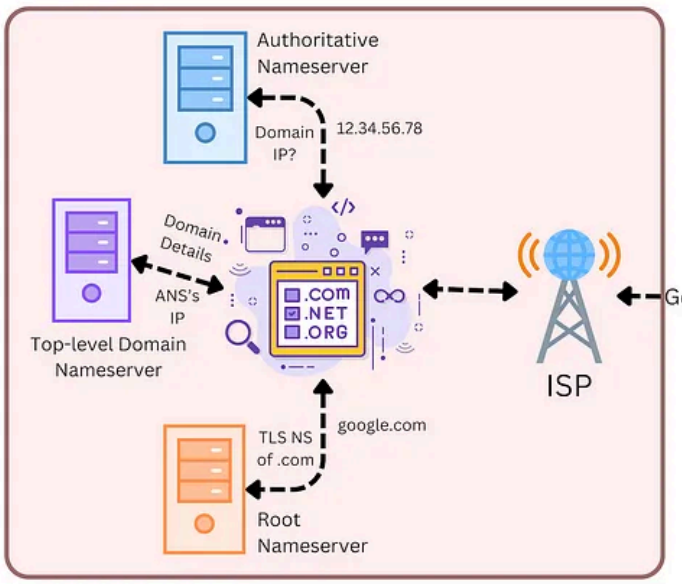
Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the process more manageable. In this blog post, we'll explore essential system design components such as DNS, load balancing, API Gateway, and more, along with a concise cheat sheet that can help developers design systems of varying complexity.

System Design Blueprint / Cheatsheet

A comprehensive visual guide that provides developers with a quick and easy reference to key concepts and best practices in system design. This handy cheatsheet or blueprint covers essential topics such as DNS, load balancing, API Gateway, video and image handling, caching, databases, unique ID

and implementing scalable, efficient, and reliable systems.

System Design Blueprint: The Ultimate Guide



Metrics | Security | Options / Algo

/@zonito

- Gzip, deflate
- Request ID
- Idempotent Key
- Many more...

Request

Response

- 4XX, 5XX, 2XX
- Pagination
- Expiry Headers
- Mime
- Cookie
- many more...

- Response Time
- Failed Status Codes
- Error Codes / Message

LOAD BALANCING

- Input Validation
- Authorization / Authentication
- Rate Limiting / Throttling
- Whitelist / Blacklisting
- Flow Control
- Authentication (OAuth 2.0)
- Request Header Validations
- TLS Termination
- Request Deduplication
- Metering / Usage data collection
- Request Dispatching

API

API Gateway

- Access Logs
- Status Codes
- Number of Requests
- Active Connections

Load Balancer
#MultiPrimary

- Round Robin
- Weighted Round Robin
- Least Connections
- Hash / Server Stickiness
- Random

Frontend Servers
#InMemoryConnections
For Live Streaming / Chat

- RTMP
- WebRTC
- Websocket
- SSE (Server Side Events)
- HTTP Short Polling
- HTTP Long Polling
- Webhook
- Stream API

User	Connection
user_1	ada_inst_obj

- Bandwidth
- Topics

Dispatch Messages to other FE Servers

CDN / Edge Servers
#Global #Regional #Static
Static / Stream with ABR

- Cache Miss Outs
- Active Resources
- Total Resources

- Push Hot Resource
- Pull Rare Resource
- Hybrid

Serves Static Content

Concurrency:
- Serially
- Serially Batch
- Pessimistic Locking
- Optimistic Locking

- Vertical Scale
- Horizontal

- DiskIO (if buffering)
- CPU
- Memory
- Latency
- Bandwidth

Backend servers
#ClusterOfServers
#MicroService

Server	Topic / Obj ID
server_1	asdf23dsf3oj23098asfdf3

Message Dispatcher

- UUID
- Auto Increment
- Auto Incr. Multiple Servers (Odd / Even)
- Twitter's SnowFlakes
- Offline Generations
- Baidu UID generator
- Sonyflake

Distributed ID Generator

- Leaderless (#Gossip Protocol)
- Leader + Followers
- Multi-Leaders

- Redlock (Redis)
- Google Chubby
- Apache Zookeeper

Distributed Resource Locking

Object Storage
#S3 #Chunk #Raw

For every new chunk object

- Count of messages
- Consumption Rate
- In-Transit (Waiting for Ack)
- Queue limit

- Broker
- Pub/Sub

Message Queue

Validate Checksum

- Count of messages
- Consumption Rate
- In-Transit (Waiting for Ack)
- Queue limit

Processed / Encoded Storage
#Lossless #Lossy #Compression

Validate checksum

Processing Workers

- Compute Time
- Failed Count
- CPU / Disk, etc

- Compression Ratio
- Storage Consumed
- Object Count

Push Processed Media Files

UPLOAD MEDIA

Key	Checksum	Timestamp
chunk_1	asdf23dsf3oj23098asfdf3	1532873423

STREAMING CHUNK METADATA TABLE

- Write-Through
- Read-Through
- Write-Around
- Write-Back

In-Memory Cache
#Fast

- No. of items
- Cache Miss & Hit
- Disk & Memory Usage

Distributed Cache
#Eviction #Invalidation

Eviction:
- LRU (Least Recently used)
- LFU (Least Freq. Used)
- FIFO
- MRU
- Random Eviction
- Least Used
- On-Demand Expiration
- Garbage Collector

Databases
#ACID #BASE

- RDMS
- Column Wide
- Document
- Key-Value
- Graph
- QuadTree
- Time-series

Cold Storage
#OldRecords

- Consistent Hashing
- Mod Hashing / Bloom Filter

- RBAC (Role Based Access Control)
- Data Encryption
- Audit Trail

- Range based
- Hash based
- Geographical
- Directory based

- Query response time
- CPU / Disk Usage
- Network Throughput
- Active Connections
- Storage

Shards
#HotNode #Hash

Replicas
#CrossGeo

- Quorum (Read / Write)
- Hinted-off
- Merkle Tree (Algo)

Pub / Sub Queue

Recommendation Service

Filtering:
- Content-Based
- Collaborative

Log Processing Service

Search Service

Notification Service

- Spamming
- Stop Words
- Dedup
- Retry with Idempotent Key

Analytics Service

- Expired / Blocked / Invalid Card
- Service Down
- Insufficient Balance

Payment Charge Service

With Idempotent Key

Third-Party Banking Service

COMMON FAN-OUT SERVICES

Section 1: Principles of System Design

1.1: Modularization

Dividing the system into smaller, manageable modules helps reduce complexity, improve maintainability, and increase reusability.

1.2: Abstraction

Hiding the implementation details and showing only the essential features helps simplify complex systems and promote modularity.

1.3: Layering

Organizing the system into layers, each layer providing a specific set of functionalities promotes the separation of concerns and enhances maintainability.

1.4: Scalability

Design systems to handle the increased load by adding more resources (horizontal scaling) or optimizing the system's capacity (vertical scaling).

1.5: Performance

Optimizing the system's response time, throughput, and resource utilization is crucial for a successful design.

1.6: Security

Ensure the system's confidentiality, integrity, and availability by implementing proper security measures and practices.

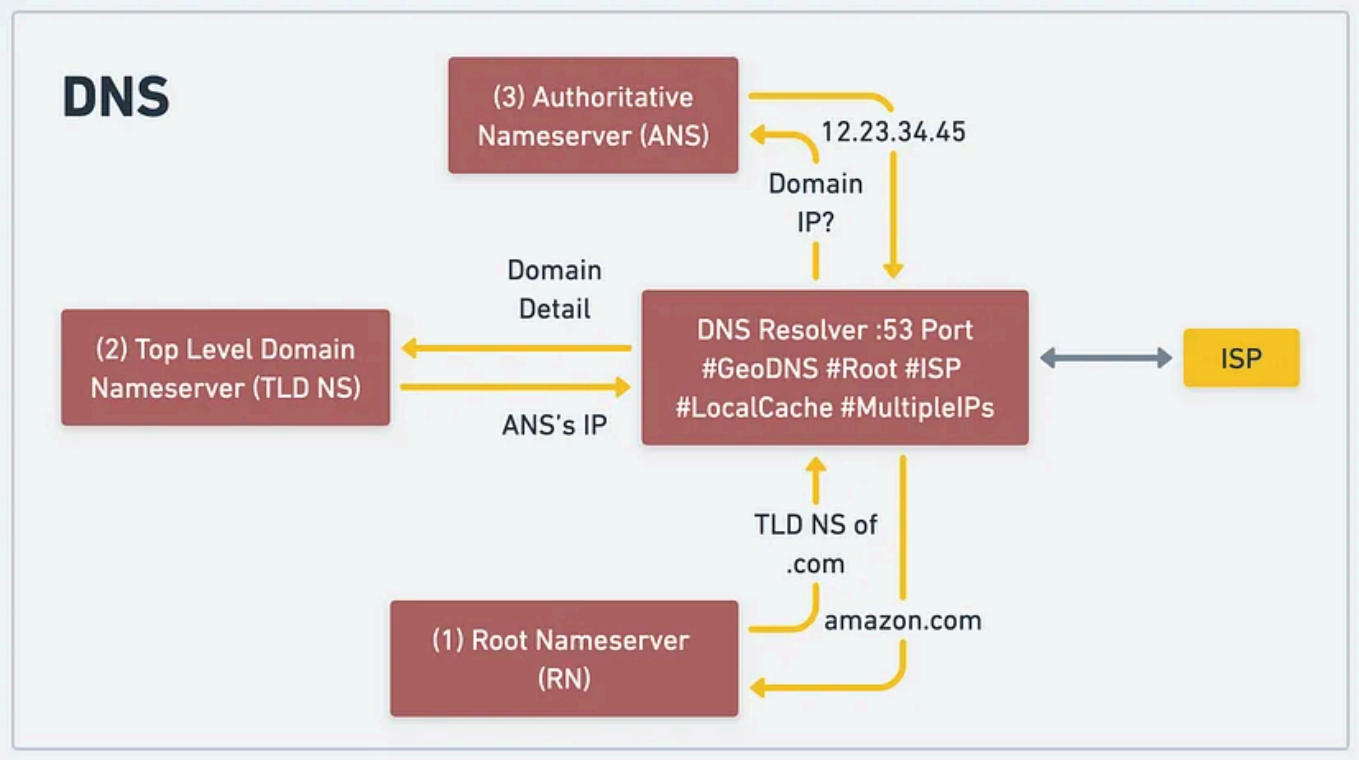
1.7: Fault Tolerance and Resilience

Design systems to withstand failures and recover gracefully from errors, ensuring reliability and availability.

Section 2: Key Components of System Design

2.1: DNS (Domain Name System)

DNS is a hierarchical and decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It translates human-readable domain names (e.g., www.example.com) into IP addresses, allowing users to access websites and services more efficiently. Read More



2.2: Load Balancing

Load balancing refers to distributing network traffic across multiple servers to ensure no single server is overwhelmed. This approach improves the system's availability, reliability, and performance. Standard load balancing algorithms include Round Robin, Least Connections, and IP Hash. Read More with Cheatsheet

2.3: API Gateway

An API Gateway is a server that acts as an intermediary between clients and microservices in a distributed system. It manages and routes requests, enforces security policies, and may provide additional features such as caching, logging, and monitoring.

2.4: Content Delivery Network (CDN)

A CDN is a network of servers distributed across various locations, designed to serve content to users with lower latency and higher bandwidth. CDNs cache content on edge servers close to end-users, improving the system's performance and reducing the load on origin servers. Read More with Cheatsheet

2.5: Message Queue

Message queues facilitate communication between distributed system components by temporarily storing messages in a queue. They enable asynchronous processing and help decouple components, improving the system's scalability and fault tolerance. Read more

2.6: Communication Protocols

Different communication protocols are used in system design, such as HTTP/HTTPS, WebSocket, and gRPC. These protocols have advantages and trade-offs, and the choice depends on factors like latency, security, and data transmission requirements.

2.7: Cache

Caching is a temporary technique used to store copies of data, allowing for faster retrieval in future requests. It helps reduce latency, server load, and bandwidth consumption. Popular caching mechanisms include in-memory caching, distributed caching, and browser caching. Read more with Cheatsheet.

2.8: Database

Choosing the appropriate database for a system depends on data structure, scalability, consistency, and latency. Common database types include relational databases (e.g., MySQL, PostgreSQL), NoSQL databases (e.g., MongoDB, Cassandra), and NewSQL databases (e.g., Cockroach DB, Google Spanner).

2.9: Replication Techniques

Replication is maintaining multiple copies of data across different nodes for increased reliability, availability, and fault tolerance. Standard replication

techniques include synchronous replication, asynchronous replication, and semi-synchronous replication.

2.10: Distributed Unique ID Generation

Creating unique identifiers in a distributed system can be challenging but is essential for maintaining data consistency and integrity. Read more with Comparison Table

Section 3: Uploading Videos and Images in Chunks Using Signed URLs

In this section, we'll explore how to upload large video and image files in chunks using signed URLs. This method can significantly improve the efficiency and reliability of file uploads, especially in scenarios where network conditions are less than ideal.

3.1: What are Signed URLs?

Signed URLs are specially-crafted URLs that grant temporary, secure access to a specific resource, such as an object in cloud storage. These URLs contain an authentication signature that allows the user to perform a particular action, such as uploading or downloading a file, for a limited period. Popular cloud storage providers like Amazon S3 and Google Cloud Storage support generating signed URLs. Here's an example of how a signed URL might look:

```
https://example-bucket.s3.amazonaws.com/my-file.txt?\
  X-Amz-Algorithm=AWS4-HMAC-SHA256&
  X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20220407%2Fus-east-1%2Fs3%2Faws4_reque
  X-Amz-Date=20220407T123456Z&\
  X-Amz-Expires=3600&\
  X-Amz-SignedHeaders=host&\
  X-Amz-Signature=a9c8a7d1644c7b351ef3034f4a1b4c9047e891c7203eb3a9f29d8c7a74676d
```

3.2: Uploading in Chunks

Uploading large files in a single request can lead to timeouts, high memory consumption, and an increased risk of failure due to network instability.

Instead, breaking large files into smaller chunks and uploading them sequentially or in parallel can improve upload efficiency and reliability. This approach is known as “chunked” or “multipart” uploads.

3.3: Combining Signed URLs and Chunked Uploads

To upload video and image files in chunks using signed URLs, follow these general steps:

1. **Divide the file into smaller chunks**: Split the large file into smaller chunks on the client side, typically using JavaScript. The chunk size can vary, but it’s essential to balance the number of requests and the size of each chunk to optimize upload performance.
2. **Request signed URLs for each chunk**: Send a request to your server to generate a signed URL for each chunk. The server should create a signed URL with the appropriate permissions and expiry time and return it to the client.
3. **Upload chunks using signed URLs**: Using the signed URLs, upload each chunk to the cloud storage service. Depending on the desired concurrency level and network conditions, these uploads can be done sequentially or in parallel.
4. **Confirm successful uploads and reassemble**: Once all chunks have been uploaded successfully, notify the server to confirm the completion of the upload process. The server can then reassemble the chunks into the original file and perform any necessary processing or validation.
5. **Handle failed uploads**: If any chunk fails to upload, retry the upload using a new signed URL or implement an error-handling strategy to ensure a smooth user experience.

By using signed URLs and chunked uploads, developers can efficiently and securely handle large video and image uploads, improving the reliability and performance of their systems.

Section 4: Chat and Streaming Protocols

This section will discuss various chat and streaming protocols facilitating real-time communication and data streaming between clients and servers. Understanding these protocols can help developers build responsive and interactive applications.

4.1: RTMP (Real-Time Messaging Protocol)

RTMP is a proprietary protocol developed by Adobe Systems for streaming audio, video, and data over the Internet. It is commonly used in video streaming applications and provides low-latency communication between clients and servers. However, due to its reliance on the Flash Player, its popularity has waned in recent years.

4.2: WebRTC (Web Real-Time Communication)

WebRTC is an open-source project that enables real-time audio, video, and data communication in web browsers and mobile applications. It supports peer-to-peer connections, reducing latency and server load. WebRTC is widely used in video conferencing, online gaming, and other applications requiring real-time communication.

4.3: WebSocket

WebSocket is a communication protocol that enables bidirectional, full-duplex communication between a client and a server over a single, long-lived connection. Due to its low latency and efficient communication capabilities, WebSocket is often used for real-time applications like chat, notifications, and live updates.

4.4: SSE (Server-Sent Events)

Server-Sent Events (SSE) is a technology that enables servers to push client updates over an HTTP connection. It is designed for one-way, real-time communication from the server to the client, making it suitable for applications like live updates, news feeds, and notifications.

4.5: HTTP Short Polling

Short polling involves clients repeatedly sending HTTP requests to the server to check for new updates. While simple to implement, short polling can

result in high server load and increased latency due to constant polling, especially when updates are infrequent.

4.6: HTTP Long Polling

Long polling is an improvement over short polling, where the client sends a request to the server, and the server holds the request open until new data is available. This approach reduces the number of requests and server load, but it may still suffer from latency issues and requires careful management of server resources.

4.7: Webhook

Webhooks are user-defined HTTP callbacks triggered by specific events in a system. When an event occurs, the source site makes an HTTP request to the URL configured for the webhook. This approach allows for efficient, event-driven communication between different systems or services.

4.8: Stream API

Stream APIs enable clients to consume a continuous data stream from a server, often using HTTP or WebSocket connections. These APIs are designed for applications that require real-time updates, such as social media feeds, stock market data, or live analytics.

By understanding and utilizing these chat and streaming protocols, developers can build responsive, real-time applications that cater to various use cases and provide engaging user experiences.

Section 5: Common Components in System Design

This section will explore some standard components often found in modern system designs. Understanding these components can help developers seamlessly integrate them into their systems and enhance overall functionality.

5.1: Payment Service

Payment services handle transactions between customers and businesses. Integrating a reliable payment service is crucial for e-commerce and subscription-based platforms. Popular payment service providers include Stripe, PayPal, and Square. These services usually provide APIs to facilitate secure transactions and manage recurring payments, refunds, etc.

5.2: Analytic Service

Analytic services enable data collection, processing, and visualization to help businesses make informed decisions. These services can track user behaviour, monitor system performance, and analyze trends. Standard analytic service providers include Google Analytics, Mixpanel, and Amplitude. Integrating analytic services into a system can help businesses optimize their offerings and improve the user experience.

5.3: Notification

Notification services keep users informed about updates, alerts, and important information. These services can deliver notifications through various channels, such as email, SMS, and push notifications. Examples of notification service providers include Firebase Cloud Messaging (FCM), Amazon Simple Notification Service (SNS), and Twilio.

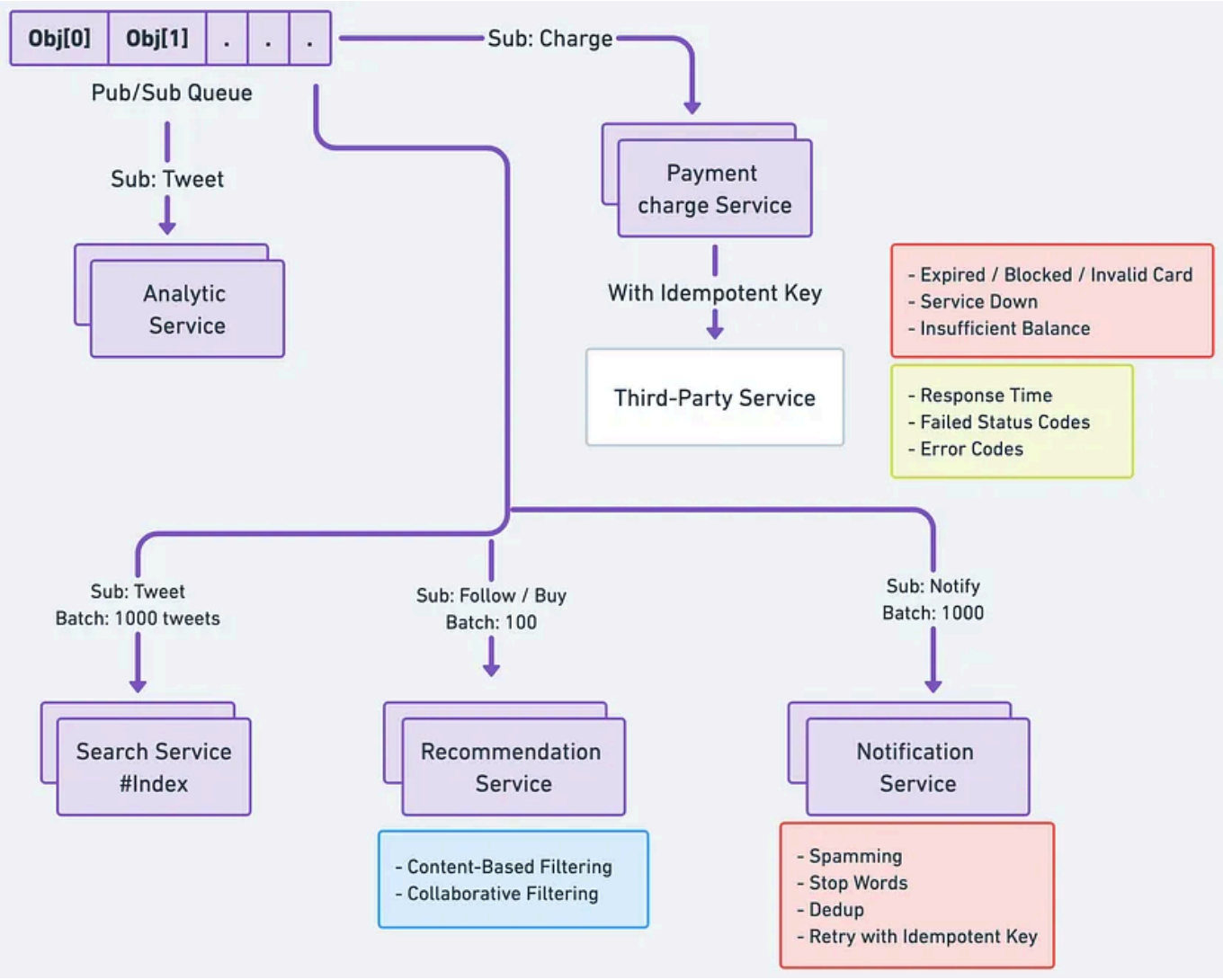
5.4: Search

Integrating a powerful search component is essential for systems with large amounts of data or content. A search service should provide fast, relevant, and scalable search capabilities. Elasticsearch, Apache Solr, and Amazon CloudSearch are popular choices for implementing search functionality. These services typically support full-text search, faceted search, and filtering, enabling users to find the information they're looking for quickly and efficiently.

5.5: Recommendation Service

Recommendation services use algorithms to provide personalized suggestions to users based on their preferences, behaviour, and other factors. These services can significantly improve user engagement and satisfaction. Techniques for generating recommendations include

collaborative filtering, content-based filtering, and hybrid approaches. Machine learning algorithms, such as matrix factorization and deep learning, can also be used to generate more sophisticated recommendations.



By incorporating these standard components into their system designs, developers can enhance the functionality of their applications and provide a more seamless and engaging experience for users.

Section 6: Best Practices for System Design

6.1: Requirement Gathering

Thoroughly understand and document the system requirements before starting the design process.

6.2: Design Patterns

Leverage proven design patterns to address recurring design problems and improve the overall architecture.

6.3: Documentation

Document your design decisions, assumptions, and rationale to ensure better communication and maintainability.

6.4: Iterative Design

Refine your Design through multiple iterations and feedback, allowing it to evolve and improve.

6.5: Testing and Validation

Validate your Design against the requirements and conduct testing to identify and address potential issues.

Conclusion

In conclusion, system design is a multifaceted and complex process that requires a deep understanding of various components, protocols, and techniques. This blog post has provided an overview of essential topics such as DNS, load balancing, API Gateway, video and image handling, caching, databases, unique ID generation, standard components like payment and recommendation services, and chat and streaming protocols.

By leveraging this knowledge, developers can create scalable, efficient, and reliable systems that cater to diverse requirements and provide a seamless user experience. It's crucial to remember that system design is an iterative process and that continuous improvement is vital to building and maintaining successful applications. With a solid foundation in these core concepts and a focus on adaptability, developers can confidently tackle the challenges of designing and implementing robust systems.

Follow me on Medium, Thread, Twitter & LinkedIn

Software Engineering | Distributed Systems | System Design Concepts | Software Architecture | System Design Interview

More from the list: "Reading list"

Curated by Tomasz Kruk

Iain St... in Digital Global ...

I Went Speed-Dating in Tokyo: It Explains Japan'...

✦ · 8 min read · Jan 16, 2024

Akhilesh Mishra

You should stop writing Dockerfiles today— Do...

5 min read · Feb 8, 2024

Shiran (Honig) ..

How to Share Da Between Micros

6 min read · Feb 18,

View list

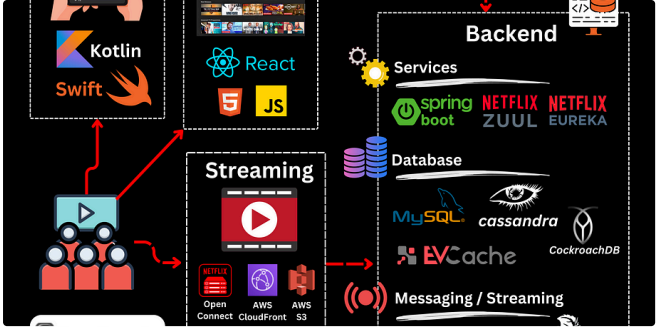
Written by Love Sharma

Follow

6.7K Followers · Editor for ByteByteGo System Design Alliance

Love is an experience cloud engineer with a demonstrated history of building large scale enterprise application.

More from Love Sharma and ByteByteGo System Design Alliance



Love Shar... in ByteByteGo System Design Allian...

Decoding Netflix: An In-Depth Look at the Tech Stack Powering the...

In the world of streaming media, Netflix stands as a titan, boasting over 232.5 million...

✦ · 16 min read · Aug 30, 2023

491 3

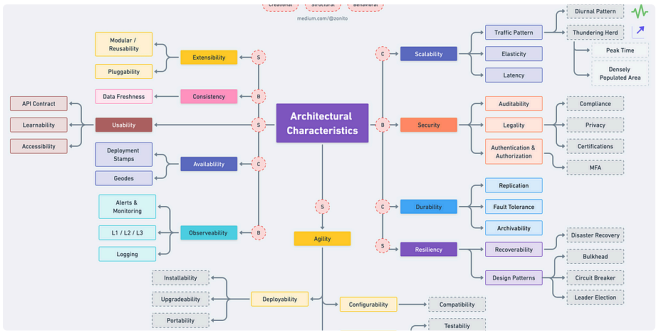
Love Shar... in ByteByteGo System Design Allian...

The Architect's Blueprint: Understanding Software Styles a...

In software development, architecture plays a crucial role in shaping the structure and...

✦ · 13 min read · Jul 10, 2023

1.6K 14



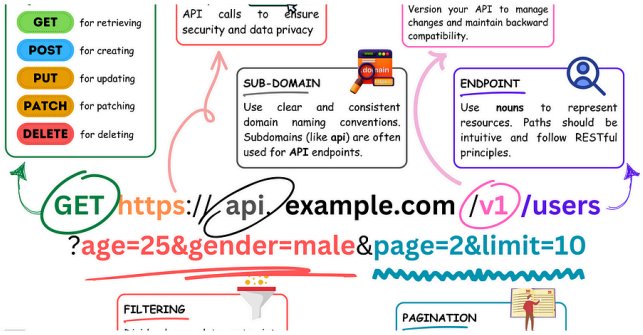
Love Shar... in ByteByteGo System Design Allian...

Top 10 Architecture Characteristics / Non-Functional Requirements...

These top 10 Architectural Characteristics covers most of the aspect of a large-scale...

✦ · 7 min read · Jun 30, 2022

1.8K 14



Love Sharma in Dev Genius

The Art of REST APIs: A Beginner's Journey Through API Space!

REST APIs are a critical component of the digital world, allowing different applications...

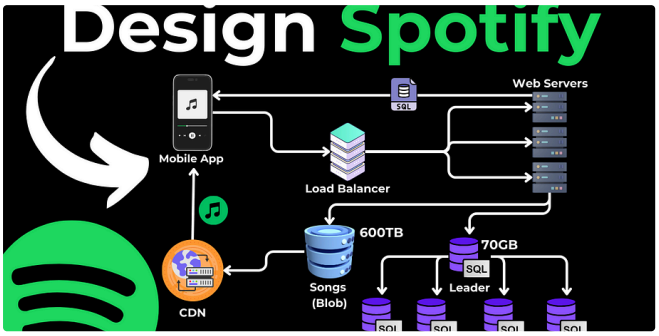
✦ · 7 min read · Jan 9, 2024

239

See all from Love Sharma

See all from ByteByteGo System Design Alliance

Recommended from Medium



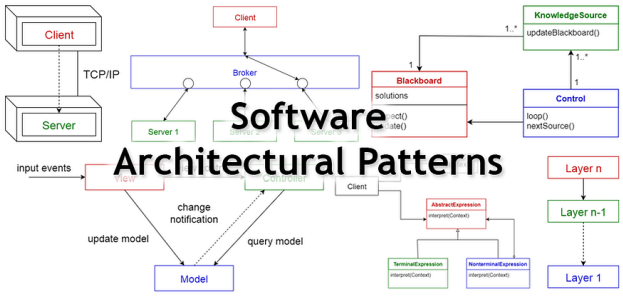
Hayk Simonyan in Level Up Coding

System Design Interview Question: Design Spotify

High-level overview of a System Design Interview Question - Design Spotify.

6 min read · Feb 21, 2024

3K 25



Vijini Mallawaarachchi in Towards Data Science

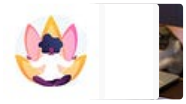
10 Common Software Architectural Patterns in a nutshell

Ever wondered how large enterprise scale systems are designed? Before major softwar...

5 min read · Sep 4, 2017

39K 128

Lists



Stories to Help You Grow as a Software Developer

19 stories · 873 saves

General Coding Knowledge

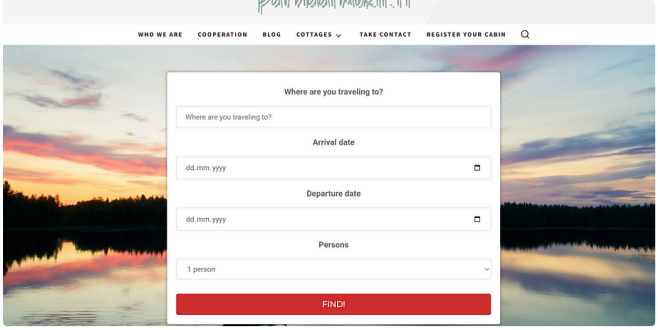
20 stories · 986 saves

Leadership

48 stories · 256 saves

Good Product Thinking

11 stories · 485 saves



Artturi Jalli

I Built an App in 6 Hours that Makes $1,500/Mo

Copy my strategy!

✨ · 3 min read · Jan 23, 2024

12.1K 146

Benoit Ruiz in Better Programming

Advice From a Software Engineer With 8 Years of Experience

Practical tips for those who want to advance in their careers

22 min read · Mar 20, 2023

14.4K 268

Talha Şahin

High-Level System Architecture of Booking.com

Take an in-depth look at the possible high-level architecture of Booking.com.

8 min read · Jan 10, 2024

2.6K 20

Serokell

Top 15 Software Development Trends in 2024

As we step into 2024, the landscape of software development continues to evolve...

12 min read · Dec 28, 2023

2.3K 31

See more recommendations