BOSSCODER ACADEMY

Best Way To Learn

System Design

FT.

REVANTH MURIGIPUDI

BOSSCODER ACADEMY

Swipe

1. Build Your Fundamentals

Learn about :

- ☐ Load balancers
- ☐ Scaling
- ☐ API Contracts
- ☐ Caching
- ☐ Database types
- ☐ Containerization
- ☐ CDN
- ☐ Message Queues

BOSSCODER ACADEMY

Swipe

Example :

- Booking Systems,
- Calendar Applications &
- Reservation Systems

BOSSCODER ACADEMY

Swipe

2. Database Schemas

Practise and come up with DB Schemas for popular systems on your own first, and then look at references online.

Coming up with a good and proper schema is extremely crucial from an interview perspective as well, so understand it from the roots.

BOSSCODER ACADEMY

Swipe

3. Learn The Architecture Of Popular Systems

Learning the architecture design of famous applications gives an understanding of how things work under the hood.

Try to understand why a component is being drawn in the design and learn about the component and how it fits into the overall design choice!

BOSSCODER ACADEMY

Swipe

While practicing, use your previous knowledge gained so far & try to come up with your architectural decisions & compare them to proper designs.

You can watch videos on big apps like (Microsoft Teams and Netflix) to understand their components.

Tap here to watch the **Microsoft Teams System Design**

Tap here to watch the **Netflix System Design**

BOSSCODER ACADEMY

Swipe

Hang on!

Preparing for

Then along with **System Design**, you have to be top-notch with problem solving in DSA and development skills as well

Bosscoder has got your covered entirely

Within a span of 7 months, you will
Develop Skills + **Hands On Experience** + **Confidence**
to grab your dream job!

Tell me more

BOSSCODER ACADEMY

Swipe

4. Learn Capacity Estimations

To learn how to design a scalable system, you need to understand some numbers.

That's where capacity estimations come into play :)

Learn how to calculate storage space, network bandwidth, daily active users (DAU), monthly active users (MAU), database load (read/write heavy) etc etc., by taking assumptions from real world systems.

BOSSCODER ACADEMY

Based on these number estimations, you will understand a few things about your system & then make choices on your design based on these numbers

Example :

- You identified that your system is read heavy, can you throw a caching layer?
- You identified requests that might take a long time & you don't want users to wait, can you use a message queue for async processing?

BOSSCODER ACADEMY

5. Component Level Drill Down

Oftentimes, Learning just the design is **not enough**.

You also need to understand every component individually, identify bottlenecks in the component and address the issues of scalability & reliability.

BOSSCODER ACADEMY

Swipe

Example :

- What would you do if you are anticipating a huge spike on DB writes?
- What happens if your caching layer starts giving cache misses very frequently?
- What happens to user data if a db server goes down? What happens to messages in your queue if they're not processed?

BOSSCODER ACADEMY

6. Trade Offs

A design is never right or wrong, it depends on the choice you make. That's where you have to consider trade offs.

Every trade off you do will have an impact on the architecture of the overall system.

Be it picking between a relational vs non relational database, stream processing vs queue processing, eventual consistency vs strong consistency in the system.