What HLD Has to capture?

1. **Discuss and be clear about the use cases**

2. **Functional Requirements**

3. **Non-Requirements:**
   Low Latency (Cache, Precompute, CDN, Pagination if list of output has to be sent, Streaming Input/Output, Asynchronous – Delegate work to background threads/workers using Queue/ Distributed Queue),
   Highly Available (No Single Point of Failure),
   Scalability (Auto Scale out, down),
   Security – Authentication (SSO, JSON Web Token part of cookie), Authorization
   Eventual Consistency
   Higher Throughput, Handle huge volume of data
   Compatibility – API Versioning, Schema Versioning

4. **Data Model:** Study the data, schema, relationship, data volume, structure or unstructured
   Based on that come up with Relational Database or NoSQL Database
   If Relational – Add Backup Machine with Centralized Storage

   Come up with all tables going to be involved – columns part of it. Primary Keys, Foreign Keys, Composite Keys – Index on multiple columns (Userid, Feedid), Avoid Duplication.

   If NoSQL: Come up with Tables, Columns, RowKey – (Salting, Hashing, Reverse Row Key), Sharding / Partition – Hash Based, Range Based on Key data, Avoid Read/Write HotSpot

5. **API Contract**

   GET -> https -> /ws/v1/users/feed

   Query Parameters:

   Request Body:

   Cookie: JSON Web Token (User Identity)

   Pagination Support

   ?Size=100&start=0
   ?size=100&start=101

   Filtering Support

?limit=100

Response:

Status Code:

200 - Success
400 – Bad Request
401 – Unauthenticated
403 – Unauthorized
404 – Not Found
500 – Internal Server Error

POST -> /ws/v1/users/feed -> used for subscribing

Query Parameter or Request Body: Feed_Id which user want to subscribe

POST is used when adding a sub item into a set of items.
PUT is used when completely update the resource

DELETE -> /ws/v1/users/feed -> used for unsubscribing

Query Parameter or Request Body: Feed_id which user want to unscubscribe

Fail when user is not part of the subscriptions.

6. **Block Diagram:**

- User -> Load Balancer -> Set of Web Servers (Web Service) -> Scale Out/Down Automatically
- Relational Databases (With Backup & Centralized Storage) / NoSQL (Sharding)
- Distributed Cache (Precompute home page, Database Rows – Frequent Users Details) -> Write Through Cache
- CDN – Content Delivery Network – HTML Pages, Audio, Video, Images, Articles – Local to users – Low Latency – HTML Page based on location (language)
- Micro Services – Each with Clear APIs, Responsibilities – Online / Offline
- Load Balancer – Round Robin / Load Based on Server Metrics / Consistent Hashing (Add / Remove Node)
- Distributed Queue – Asynchronous, Faster Response
- NoSQL Database – Sharding – Hash Based (Row Key – Salting, Reverse Key, Hashing), Ranged Based on Key – Place data near which will be fetched together.
    - Daily Data, Retention, Archive

- Web Socket – For Chat Servers like Messenger – Bi-Directional Request Alternatives: Long Polling or Long Socket Connection
- Streaming Input / Output for videos – Buffer – Add next packet
- Device Resolution Based – Audio, Video, Image storage
- Adaptive Resolution – Based on user bandwidth change the resolution of data getting transmitted.
- Precompute – Timeline Page for frequent users – Analytics
- HTTPS – TLS/SSL -> Server will have private key and share public key to Client -> Client prepares a session key encrypted using public key and send it to server which uses private key to decrypt. Both Client and Server will talk using the session key throughout the session.
- **Distributed Concepts** – Leader Election, ZAB Protocol, Write Ahead Log (WAL), Replication
- CAP – Consistency, Availability, Partition Tolerance

  Relational Database – Consistency + Partition Tolerance

  NoSQL, Zookeeper – Availability + Partition Tolerance + Eventual Consistency

- ACID Principles – Atomicity (Transaction commit or rollback completely), Consistency (Read what is committed by some other write immediately), Isolation (Two transactions can run in isolation and sees other write), Durability (data is persisted)
- Retention – Archive History
- Design Patterns – Open Closed Principles
- Users – From Multiple Devices

7. **Distributed System Algorithms:**
   - Leader Election
   - ZAB
   - Raft Protocol – Consensus Algorithms
   - Data Pipelines – Write
   - Read Path – Follower, Observer, Leader
   - Distributed Coordination, Configuration
   - In-Memory Databases
   - Failover Providers
   - Distributed State Store
   - WAL – Write Ahead Log, In-Memory Buffers for read and write
   - Distributed Cache Algorithms – LRU, LFU, Write Ahead, Read through
   - Pull and Push Based Models
   - Notification Service