Questions to Ask/Try to Answer:

1. In regards to this aspect of the system, do we want to focus on consistency or availability?
2. Who are the users of the system? Are we looking at this on a consumer level or more of an enterprise solution level as our main target audience?
3. Are there tiers (different Service Level Agreements) to the user experience?
4. What will be the main workflows (reads vs writes)?
5. How many requests do we expect to see every second?
6. What is the maximum number of users we are expected to support and what is the average?
7. Define the core entities and relationships that they have (ex. Objects that I would put in an ER diagram)
8. What is our main lookup key for the items in our database?
9. Expected end-to-end latency? Should some operations be faster than others?
10. How should we handle conflicts/duplicates?
    a. Standard answer could just be going with operational transform which is where writes can happen from the same version, but if one write happens after another then its position is shifted
11. Is there a target uptime for a feature?
    a. The standard answer for this is to usually have backup replicas and such that can replace the system if it needs to be able to. We can use a load balancer to help make it so that if any one server/compute goes down then traffic is rerouted to the other instances to maintain availability.
12. What type of authorization are we planning to use?
    a. Amazon Cognito can be used in sync with email/password availability or Google OAuth
13. Are we expecting that the functionality of the application to be based in an API and how will we connect the API?
    a. We can use Amazon API Gateway to be able to connect to our service using JSON Web Tokens (JWTs)
    b. Determine whether it should be HTTPS one way (CRUD) or Websocket so server and client can communicate bidirectionally
14. What are our success measures?

Tool / Layer Segments	Purpose & Typical Integration Point	Key Advantages
Edge / Ingress Layer		
Route 53	Global DNS plus health-check–based routing (latency, weighted, fail-over) in front of any public endpoint.	Anycast DNS, fast fail-over, traffic-policy engine
AWS Global Accelerator	Any-cast edge entry that forwards TCP/UDP to the nearest healthy ALB/NLB across regions.	Single Anycast IP, automatic multi-region fail-over, lower RTT
Amazon CloudFront	Global CDN that caches static & dynamic assets; fronts S3, ALB, API Gateway.	400+ PoPs, TLS off-load, edge compute (Lambda@Edge/Functions)
Elastic Load Balancer (ALB/NLB)	Distributes traffic across ECS/EKS/EC2; ALB adds HTTP routing & WebSockets, NLB offers ultra-low-latency TCP/UDP.	Managed autoscaling, multi-AZ, native TLS termination
Amazon API Gateway	Single HTTPS or WebSocket endpoint for REST/GraphQL; integrates with Lambda, ALB, Cognito, WAF.	Built-in auth, throttling, request transformation, serverless
Compute Layer		
AWS Lambda	Event-driven FaaS; runs code on triggers (API Gateway, S3, Kinesis) with per-ms billing.	Zero ops, auto-scale to 1000s, sub-second cold starts
Amazon ECS (Fargate)	Serverless container runtime; schedule Docker tasks without cluster admin.	Pay-per-second, integrated autoscale, AWS-native IAM/SG
Amazon EKS	Managed Kubernetes control plane for portable CNCF workloads.	Upstream-compatible K8s, IAM Roles for Service Accounts, Fargate pods
Amazon EC2	Elastic VMs when you need custom OS/kernel, stateful services, or GPUs.	Full OS control, Spot & Graviton cost options, wide instance catalog
Data Storage Layer		
Amazon S3	“11-nine” durable object store for backups, media, logs, and large blobs.	Infinite scale, lifecycle tiering (IA/Glacier), event notifications

Amazon RDS / Aurora	Managed relational DB (MySQL, Postgres, Aurora) with backups, read-replicas, Multi-AZ.	Familiar SQL, automatic patching, autoscaling readers, >100 k TPS
Amazon DynamoDB	Serverless key-value/JSON store with single-digit-ms latency and Global Tables.	Unlimited scale, on-demand capacity, point-in-time restore
Amazon ElastiCache (Redis)	In-memory cache or pub/sub bus; drops p99 read latency to sub-ms.	Managed fail-over, clustering, Redis 7 features, data-tiering
AWS Glue Data Catalog	Central schema/metadata catalog + serverless ETL (Glue Jobs) for S3 & lakehouses.	Auto-crawl, Parquet conversion, native Athena/Redshift integration
Messaging / Streaming Layer		
Amazon SQS	Durable at-least-once message queue for decoupling producers/consumers.	Infinite backlog, dead-letter queues, 12 h retention free
Amazon SNS	Fan-out pub/sub topics pushing to SQS, Lambda, HTTPS, SMS/e-mail.	Push-based, multiple protocols, FIFO option
Amazon Kinesis Streams / Firehose	Ordered log for real-time ingestion; Firehose auto-loads to S3/Redshift/ES.	70 MB/s shard, 365-day retention, serverless delivery transform
Amazon MSK (Kafka)	Fully managed Apache Kafka clusters for large-scale event streaming.	Native Kafka API, automated patching, tiered storage
Amazon EventBridge	SaaS/AWS/custom event bus with JSON filtering rules; connects micro-services without polling.	Schema registry, cross-account routing, 100% serverless
Orchestration / Workflow		
AWS Step Functions	State-machine orchestrator chaining Lambdas, ECS, API calls with retries and parallel branches.	Visual workflow, built-in exponential back-off, 1-year execution history
Monitoring		
Amazon CloudWatch	Central metric, log, trace, and alert platform; feeds autoscaling & dashboards.	One-minute (custom 1 s) metrics, Logs Insights, X-Ray traces
Identity & Security		

AWS Cognito	Managed user directory/OAuth broker; issues JWTs for SPA & mobile apps.	Social sign-in, adaptive MFA, device tracking
AWS IAM	Fine-grained policies & roles for people, services, and resources.	Least privilege, cross-account access, IAM Access Analyzer