CAMPERSO Framework - Data Engineering System Design Cheat Sheet

1. Capture (Ingestion)

Clarification Questions:

- What is the incoming event volume per second?
- Is it real-time (event-time) or batch (ingest-time)?
- Are events pushed or pulled?
- Is ordering important per user/session?

Mentor's Answer:

Ingest ~10K/sec using Kafka (push model) with topics per app, partitioned by user_id to preserve order.

Tools & Examples:

Apache Kafka, Avro, Airbyte, Kinesis

2. Audit & Validate

Clarification Questions:

- What data quality rules must be enforced?
- Are schema changes expected frequently?
- How do we handle invalid records?
- Do we need validation at ingest and after loading?

Mentor's Answer:

Use PyDantic at ingest; Great Expectations post-load. Invalids go to Kafka DLQ.

Tools & Examples:

PyDantic, Great Expectations, Kafka DLQ, Airflow

3. Model (Schema Design)

Clarification Questions:

- Should the model support historical changes (SCD2)?
- What metrics are needed? Sessions, clicks, journeys?
- Are we modeling for BI (denormalized) or storage optimized?
- Is there need for dimensional modeling (star/snowflake)?

Mentor's Answer:

Star Schema with SCD2 in dim_user. Use surrogate keys. Fact_clicks joined to dimensions.

CAMPERSO Framework - Data Engineering System Design Cheat Sheet

Tools & Examples:

dbt, Snowflake, surrogate keys, SCD2

4. Persist (Storage & Format)

Clarification Questions:

- Do we need ACID, schema evolution, or time travel?
- What?s the retention period for raw data?
- Do we query from Spark or SQL engines?
- Do we need incremental loads?

Mentor's Answer:

Store in Delta Lake on S3, partitioned by dt. Z-Order on session_id for performance.

Tools & Examples:

Delta Lake, Apache Iceberg, Spark, Hudi, dbt

5. Expose (Query Layer)

Clarification Questions:

- Are real-time dashboards required?
- Who consumes the data ? BI analysts, apps, ML?
- What are latency requirements (<1 sec or hourly)?
- Are APIs needed to access processed data?

Mentor's Answer:

Druid for real-time queries; dbt to Snowflake for batch analytics. REST APIs optional.

Tools & Examples:

Apache Druid, dbt, Snowflake, FastAPI

6. Resilience (Fault Tolerance)

Clarification Questions:

- What failure scenarios are critical to guard against?
- Should we reprocess failed data or skip?
- What?s the tolerance for delayed data?

- Are retries and checkpoints required?

Mentor's Answer:

Enable Spark checkpointing and Airflow retries. Use badRecordsPath to isolate corrupt data.

Tools & Examples:

Spark Structured Streaming, Airflow retries, DLQ

7. Security

Clarification Questions:

- Is clickstream data considered PII (e.g., user_id)?
- Do we need to mask or hash identifiers?
- How do we manage access control to raw/curated layers?
- Are we compliant with GDPR or CCPA?

Mentor's Answer:

Encrypt at rest with KMS, mask PII with SHA-256, and implement RBAC in Snowflake.

Tools & Examples:

AWS KMS, Snowflake RBAC, dbt masking, IAM

8. Observability

Clarification Questions:

- How will we detect data pipeline failures?
- Do we track freshness, lineage, and volume drops?
- Do we monitor schema changes over time?
- Should we alert on processing delays?

Mentor's Answer:

Use Airflow and Prometheus for metrics. Log schema versions and use OpenLineage for tracking.

Tools & Examples:

Prometheus, Grafana, Airflow, OpenLineage, Great Expectations logs