System Design - End Term - Yerlan Dias

Task 1

[Github repo link]

Functional requirements:

1. Real-Time event ingestion: likes, comments, shares with user metadata(user ID, timestamps etc.)
2. Real-Time processing
3. Enrichment of profile and metadata events and joining of streams
4. Storage: aggregated metrics in real-time OLAP DB
5. Data indexing
6. Expose REST/GraphQL APIs
7. Parametrized queries
8. Frontend dashboard
9. Autorefresh widgets via WebSockets
10. Historical analysis
11. Admin panel
12. Monitoring

Non-Functional requirements:

1. System response time for analytics queries must be < 200ms
2. API latency < 500 ms for 95th percentile queries
3. Strong consistency and atomicity for all critical operations
4. Front refresh every 5 sec
5. Horizontal scalability at every layers: Kafka topics with partitions, OLAP Db with partitioning and replication
6. 3x HDD (250 GB each)
7. 16 GB RAM total per server
8. 8 vCPU per server
9. 100K+ eps at peak loads

10. System availability 99.9%
11. No single point of failure
12. Authentication/Authorization (OAUTH2. JWT)
13. RBAC for user roles - Manager, Admin, Analyst
14. Data Encryption in transit - TLS and at rest AES-256
15. Audit logs for all sensitive access
16. Maintainability: Rolling deployments with zero downtime
17. Cold storage for historical data

In following architecture we capture and ingest events from multiple sources (users, backend systems, ad systems) in real-time. The Data sources are:

- Frontend Clients - mobile/dev, where they make events as likes, shares, comments etc.
- Backend Services - events as post creation, ad impressions, search queries
- Ad systems or Third-party APIs - campaign performance, conversions, cost per click

We use ingestion tools like Apache Kafka, which acts as a real-time message queue and partitioned for horizontal scalability. Kafka Connect to integrate with other Rest APIs, databases

In Stream Processing Layer, the frameworks such as Apache Flink and Spark Structured Streaming are used for real-time and windowed aggregations. Flink is responsible for engagement aggregation - aggregates likes, comments and calculates engagement scores per region/post/user, for trending topic detection - performs windowed aggregations on hashtags, keywords and to monitor ad performance.

In data storage layer we store real-time and historical data for querying and visualization, specifically the OLAP Store such as ClickHouse, Apache Pinot. Reasons to use the OLAP store is that it ingests the processed data from Flink or directly from Kafka and supports fast and high-concurrent analytical queries. For cold storage which will store raw and enriched events in Parquet we use Amazon S3. And finally for hot data store we use Redis for fast access to data such live sessions, real-time search etc.

Finally we need API Layer to provide REST/GraphQL endpoints for frontend. For visualization tools we use Grafana + ClickHouse plugin