SOFTWARE ARCHITECTURAL PROBLEMS FACES BY INDUSTRY:

Problem 1:

Walmart, a global retail giant, faces challenges in managing its inventory across thousands of stores. The central inventory management system occasionally fails to sync stock levels in real-time, leading to overstocking in some locations and stockouts in others. This affects customer satisfaction and increases operational costs.

Root Causes:

The root causes identified by Walmart software architects are:

1. **Outdated database architecture** that struggles to handle large-scale, concurrent operations.
2. **Lack of real-time synchronization** between local stores and central warehouses.
3. **Poor network resilience** in regions with unreliable internet connectivity.

Solution:

1. Transition to a **distributed database system** like Apache Cassandra for real-time updates.
2. Use **event-driven microservices** to decouple inventory updates and improve fault tolerance.
3. Deploy **edge computing nodes** at regional warehouses for faster processing and syncing with the central system.
4. Integrate **machine learning models** to predict demand and optimize inventory distribution.

Problem 2:

JPMorgan Chase(a financial services organization) experiences periodic downtime in its online banking services during peak hours, especially on payday weekends. This impacts customer trust and results in regulatory scrutiny.

Root Causes:

1. **Monolithic architecture** that slows down scaling efforts.
2. Insufficient **disaster recovery planning** for peak load scenarios.
3. **Single point of failure** in the transaction processing module.

Solution:

1. Migrate to a **microservices-based architecture** for scalability and resilience.
2. Implement **circuit breakers and retry mechanisms** to handle transaction failures gracefully.

3. Use **containerization tools** like Kubernetes to scale services dynamically.
4. Create **active-active disaster recovery sites** with real-time failover capabilities.

Problem 3:

Netflix faces intermittent buffering and quality drops during the release of highly anticipated shows. This occurs due to an overwhelming number of simultaneous user requests.

Root Causes:

1. **Inefficient load distribution** among global data centers.
2. Over-reliance on centralized servers for content delivery.
3. Limited **predictive analytics** for traffic spikes.

Solution:

1. Expand the use of **content delivery networks (CDNs)** to cache content closer to users.
2. Use **adaptive bitrate streaming** to adjust video quality dynamically based on network conditions.
3. Employ **AI-driven traffic forecasting** to prepare for expected spikes in demand.
4. Transition to **multi-cloud deployment** to leverage resources across different cloud providers.

Problem 4:

During events like New Year's Eve, the platform's systems fail to handle the sudden surge in demand, leading to delays and customer dissatisfaction.

Root Causes:

1. **Overloaded databases** handling surge pricing calculations and ride requests.
2. **Latency issues** in the ride-matching algorithm during high-demand periods.
3. Insufficient **scaling policies** to handle rapid traffic surges.

Solution:

1. Use **in-memory databases** like Redis for real-time pricing calculations.
2. Deploy **serverless functions** for ride-matching to scale instantly based on demand.
3. Implement **geo-distributed microservices** to reduce latency in specific regions.
4. Enhance **data streaming pipelines** for faster analytics and decision-making.

Problem 5:

Mayo Clinic's patient management system experiences slowdowns when doctors access medical records during peak outpatient hours. This delays patient care and increases the workload for medical staff.

Root Causes:

1. Centralized EMR systems with **limited scalability**.
2. **Inconsistent data formats** across different departments.
3. Lack of **real-time synchronization** for shared patient data.

Solution:

1. Adopt **cloud-native EMR systems** for better scalability.
2. Implement **FHIR (Fast Healthcare Interoperability Resources)** standards for consistent data exchange.
3. Use **AI-powered data indexing** to quickly retrieve patient records.
4. Introduce **smart caching mechanisms** for frequently accessed patient data

Author: Rehman Ghani