SUBMITTED BY:

MUHAMMAD SAQIB

SUBMITTED TO:

SIR MUKHTIAR ZAMIN

REGISTRATION NUMBER:

FA22-BSE -005

MAJOR ARCHITECTURE PROBLEMS IN SOFTWARES:

Security Vulnerabilities in API Architecture

Software: Twitter (Social Media)
Problem:
APIs were vulnerable to unauthorized access and abuse, such as bots spamming the platform.

Solution by Twitter:

1. Adopted **OAuth 2.0**:
   - Secured user authentication and authorization.
2. Introduced **Rate Limiting**:
   - Limited the number of API calls per user to prevent abuse.
3. Used **API Gateways**:
   - Centralized API management and applied security policies.

Outcome:
Improved API security and reduced bot-related spam activities.

Poor Fault Tolerance in Distributed Systems

Software: **Spotify (Music Streaming Service)**
Problem:
A failure in one service could cascade and impact other services, leading to application-wide outages.

Solution by Spotify:

1. **Resilience Engineering**:
    - Built fault-tolerant systems using tools like Hystrix for circuit breakers.
2. **Geo-Redundancy**:
    - Deployed services across multiple regions to ensure availability during failures.
3. **Chaos Engineering**:
    - Regularly simulated failures to identify and fix weak points.

Outcome:
Spotify achieved high availability and resilience, ensuring uninterrupted music streaming globally.

Real-Time Data Processing Bottlenecks

Software: **Uber (Ride-Hailing App)**
Problem:
Uber's system struggled with real-time data processing to calculate routes, find nearby drivers, and handle surge pricing dynamically.

Solution by Uber:

1. Introduced **Lambda Architecture**:
    - Combined batch processing (for historical data) and stream processing (for real-time data).
2. Used **Apache Kafka**:
    - Streamlined real-time messaging between services.

Outcome:
Uber achieved low-latency responses for ride requests, even during peak traffic.

High Latency in Content Delivery

Software: **Netflix (Streaming Platform)**
Problem:
High latency in delivering video content to users worldwide, especially during peak hours.

Solution by Netflix:

1. Implemented **Content Delivery Network (CDN)**:
    - Built their own CDN called **Open Connect**, placing servers closer to users.
2. Used **Dynamic Video Encoding**:
    - Adjusted video quality based on user bandwidth in real-time.

Outcome:
Seamless streaming experience for millions of users, even during global releases.

Monolithic Architecture Scalability

Software: **Amazon (E-commerce)**
Problem:
Amazon initially used a monolithic architecture, where the entire platform was a single application. As user traffic grew, scaling specific components (e.g., search or recommendations) became inefficient.

Solution by Amazon:

1. Migrated to **Microservices Architecture**:
   - Broke down the monolithic system into hundreds of microservices.
   - Each service handled a specific functionality (e.g., cart service, payment service).
2. Adopted **Event-Driven Architecture**:
   - Used services like Amazon SQS and SNS for inter-service communication.

Outcome:
Improved scalability and fault isolation, enabling Amazon to handle billions of requests daily.