>> **How to Make applications speak about its problems**

1.Logging:

- Applications often use logging frameworks to record events, errors, and warnings. Logs can provide detailed information about what happened, including error messages, stack traces, and context.
- Example: In a web application, you might use a logging framework like Log4j or SLF4J to log errors, warnings, and information about the application's behavior.

2.User Interface Messages:

- Applications with graphical user interfaces (GUI) often display error messages or notifications to users when something goes wrong. This helps users understand the issue and take appropriate action.
- Example: A file management application might display a message like "File not found" or "Permission denied" when a user tries to access a file without the necessary permissions.

3.Email Notifications:

- Some applications send email notifications to administrators or users when critical errors occur. This allows for timely awareness and intervention.
- Example: An e-commerce platform might send an email to the site administrators when the payment processing system encounters an error.

4.API Responses:

- Web services and APIs often return specific error codes and messages to indicate problems. Clients can then interpret these responses and handle errors accordingly.
- Example: A RESTful API might return a JSON response with a specific error code and message when an invalid request is made.

5.Event System:

- Applications with event-driven architectures may emit events or signals when errors occur. Other components or modules can subscribe to these events and respond accordingly.
- Example: A microservices architecture might have a centralized event bus where services can publish and subscribe to events, including error events.

Monitoring Tools:

1.Zabbix

Zabbix is a monitoring solution that offers network and server monitoring together with application and service monitoring.

One can use it to monitor your entire physical server and/or cloud infrastructure. It is a versatile monitoring solution that can be used for monitoring network devices even on a data center level.

Another advantage of Zabbix is its integration with multiple popular ticketing and alerting systems, not to mention ITSM and IoT integration capabilities. You can use Zabbix together with Jira, Oracle, Jenkins, and NGINX.

2.Nagios

One of the oldest monitoring software solutions that is still popular today is Nagios. This monitoring solution can keep an eye on your entire IT infrastructure and offers several different versions, including the completely free Nagios Core.

With Nagios you can monitor all your infrastructure components, including operating systems, applications, services, multiple network protocols, and systems metrics.

3.Prometheus

Prometheus is a versatile monitoring tool, which you can use to monitor a variety of infrastructure and application metrics. Here are a few common use cases. Prometheus is typically used to collect numeric metrics from services that run 24/7 and allow metric data to be accessed via HTTP endpoints.

A typical monitoring platform with Prometheus is composed of multiple tools:

- **Prometheus server:** the main Prometheus server which scrapes and stores time series data
- **Client libraries:** client libraries for instrumenting application code
- **Push gateway:** a push gateway for supporting short-lived jobs
- **Exporters:** special-purpose exporters for services like HAProxy, StatsD, Graphite, etc.
- **Alertmanager:** an alertmanager to handle alerts

>>> Prometheus's main features are:

A multi-dimensional data model with time series data identified by metric name and key/value pairs

PromQL, a flexible query language to leverage this dimensionality. No reliance on distributed storage; single server nodes are autonomous.

Time series collection happens via a pull model over HTTP. Pushing time series is supported via an intermediary gateway

Targets are discovered via service discovery or static configuration. Multiple modes of graphing and dashboarding support.

4.Graphite:

Originally designed as a side project at Orbitz, Graphite is a monitoring tool that works equally well in the cloud or on cheaper physical IT infrastructure, so it is a popular choice for companies of all sizes. This solution is typically used for monitoring the performance of servers, applications, and websites.

With Graphite it is possible to gather and visualize multiple numerical time series data points or performance indicators.

5.Splunk:

Splunk is a widely used platform for searching, monitoring, and analyzing machine-generated data. It's not just a monitoring tool but also a comprehensive solution for log management and data analytics.

6.Grafana:

Grafana is an open-source platform for monitoring and observability. It can integrate with various data sources, including Prometheus, InfluxDB, and Graphite, to create customizable dashboards.

>>Wallet App Bussiness WorkFlows:

Wlallet App implements various business workflows to provide users with secure, convenient, and efficient financial and transactional services. Here are common business workflows implemented in a wallet application:

User Registration and Onboarding:

Allow users to register and create an account. Implement a secure onboarding process, including identity verification.

Profile Management:

Enable users to manage their profiles, including personal information, contact details, and preferences.

Wallet Creation and Initialization:

Allow users to create and set up their digital wallets. Initialize the wallet with basic details and preferences.

Transaction History:

Maintain a detailed transaction history, displaying both incoming and outgoing transactions. Include information such as transaction amounts, dates, and transaction types.

Bill Payments:

Allow users to recharge mobile balances and pay utility bills directly from the wallet application. Integrate with various service providers for bill payments.

Financial Planning and Investments:

Information on financial planning, goal setting, and investment options. Provide users with insights into investment performance.

Logout and Session Management:

Implement secure logout procedures to terminate user sessions. Manage session timeouts and ensure secure handling of user sessions.

These business workflows collectively contribute to the overall functionality and user experience of a wallet application, making it a versatile tool for managing financial transactions, payments, and related activities.

>>>>>**Observability Aspects:**

Achieving observability requires the implementation of several key practices such as logging, metrics, and tracing. Logging provides a record of system events, while metrics measure system performance. Tracing offers detailed information about the flow of data through a system. Together, these practices enable SRE teams to quickly diagnose and resolve issues, ensuring that systems are reliable and performant. Let us look at these factors individually:

1) System Metrics Monitoring

One of the most popular techniques to establish observability in SRE is through monitoring system metrics. Engineers can understand the behavior of the system and spot potential problems by gathering and analyzing data like CPU consumption, memory usage, and network latency.

For monitoring system metrics, a variety of tools are available, including Prometheus, Grafana, and Datadog. These tools give engineers the ability to gather and evaluate metrics in real time, giving them invaluable information about how systems behave.

2) Logging

Another crucial method for achieving observability in SRE is logging. Engineers can watch the behavior of the system over time and spot possible problems by logging system events and faults.

The logging frameworks Log4j, Logback, and Fluentd are just a few of the ones that are accessible. Engineers can log system events and faults using these frameworks, then analyze them with tools like Elasticsearch and Kibana.

3) Tracing

An additional method for achieving observability in SRE is tracing. Engineers can analyze the behavior of a system and spot potential problems by tracking requests and transactions through it.

Tracing frameworks like Zipkin, Jaeger, and OpenTelemetry are among the many that are accessible. Engineers may use tools like Grafana and Prometheus to evaluate requests and transactions as they move through a system thanks to these frameworks.

4) Distributed Tracing

A method for achieving observability in complicated, distributed systems is distributed tracing. Engineers can comprehend the behavior of the entire system and spot possible problems by tracking requests and transactions across various services.

Use Case on Observability aspects:

In banking, SRE observability can be used to monitor critical financial transactions such as wire transfers, ATM withdrawals, and online payments. By collecting and analyzing data from various systems and applications involved in these transactions, SREs can detect anomalies and quickly identify and resolve any issues that may arise. This not only helps to ensure the reliability and availability of these transactions but also enhances the security and compliance of banking operations. SRE observability also enables proactive capacity planning and optimization, allowing banks to efficiently scale their systems to meet the growing demand for digital financial services.
