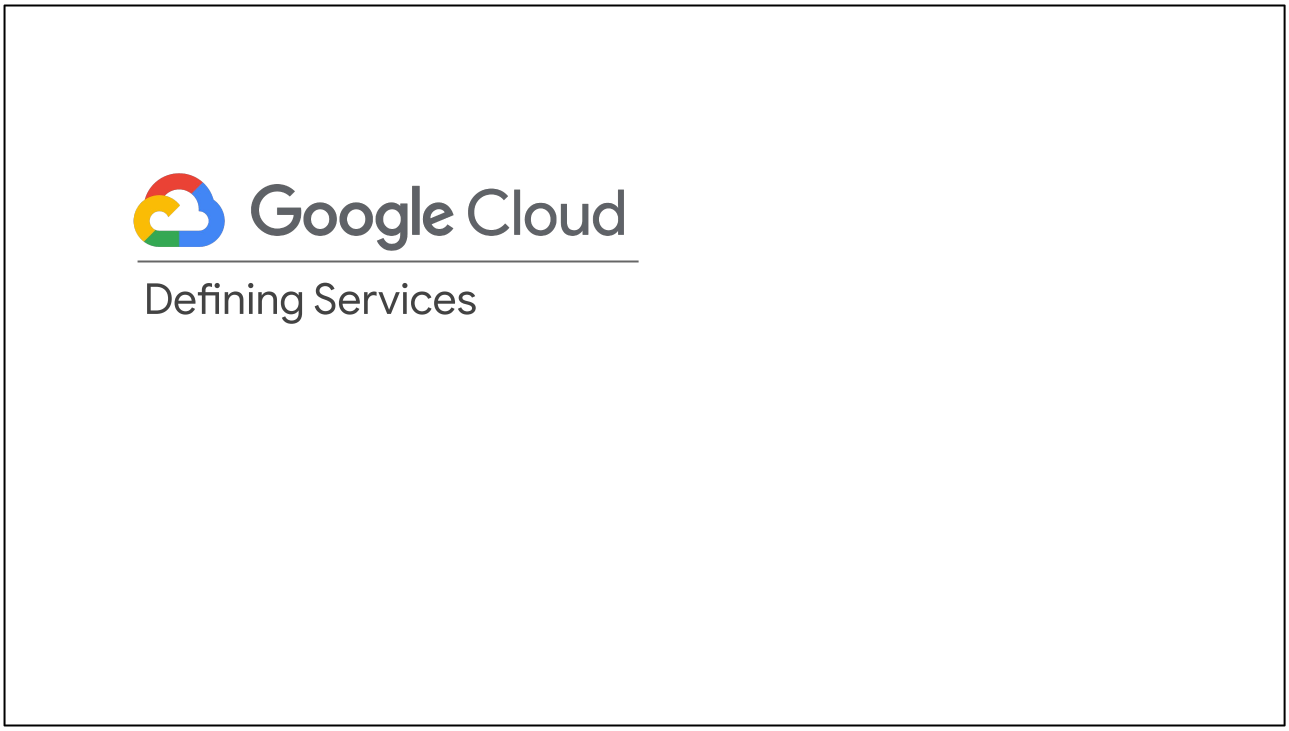
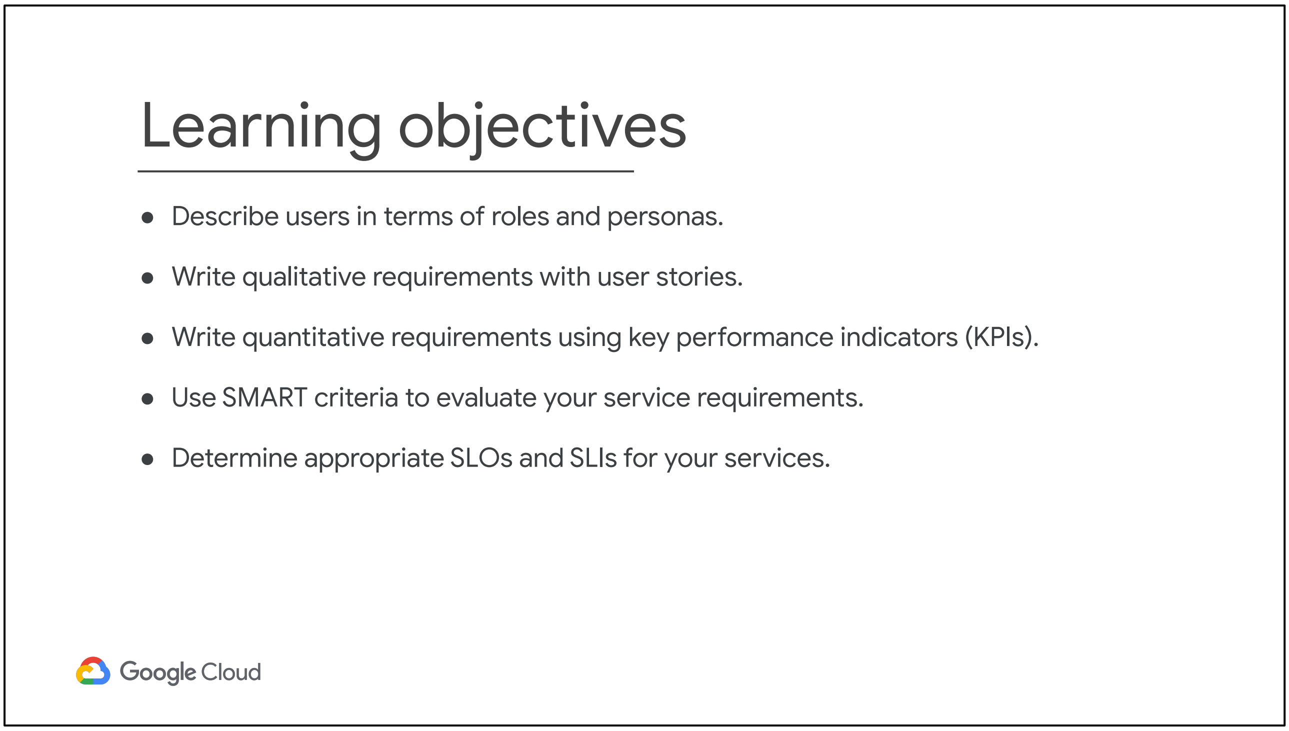


Google Cloud

Defining Services

A new development begins with the planning and design phases. These require information gathering, starting with business requirements. Once the requirements are defined, it is important to measure that they are providing business value. In this module, we will look at gathering requirements and then techniques for measuring the impact of the solutions.

Let's take a closer look at what we will cover.



Learning objectives

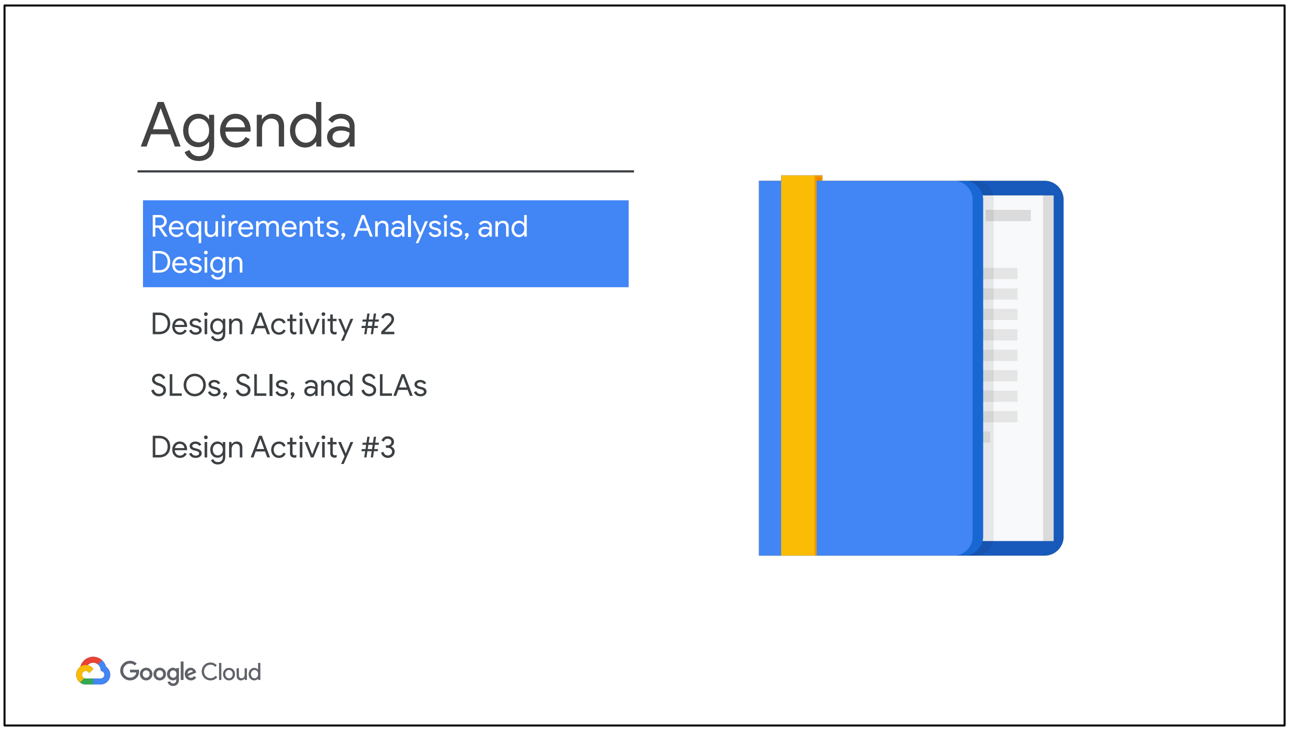
- Describe users in terms of roles and personas.
- Write qualitative requirements with user stories.
- Write quantitative requirements using key performance indicators (KPIs).
- Use SMART criteria to evaluate your service requirements.
- Determine appropriate SLOs and SLIs for your services.

Google Cloud

An architect starts the planning phase by collecting information, starting with business requirements. These provide a context for the architectural design and technical decisions. Examples of business requirements include accelerating the pace of software development, reducing capital expenditure, and reducing time to recover incidents.

These requirements may be financial, based on customer experience or operational improvements. These requirements are satisfied by a number of technical decisions. The technical requirements of a system are the functional and non-functional features required.

This module considers capturing the technical requirements and then discusses how these can be measured and evaluated.

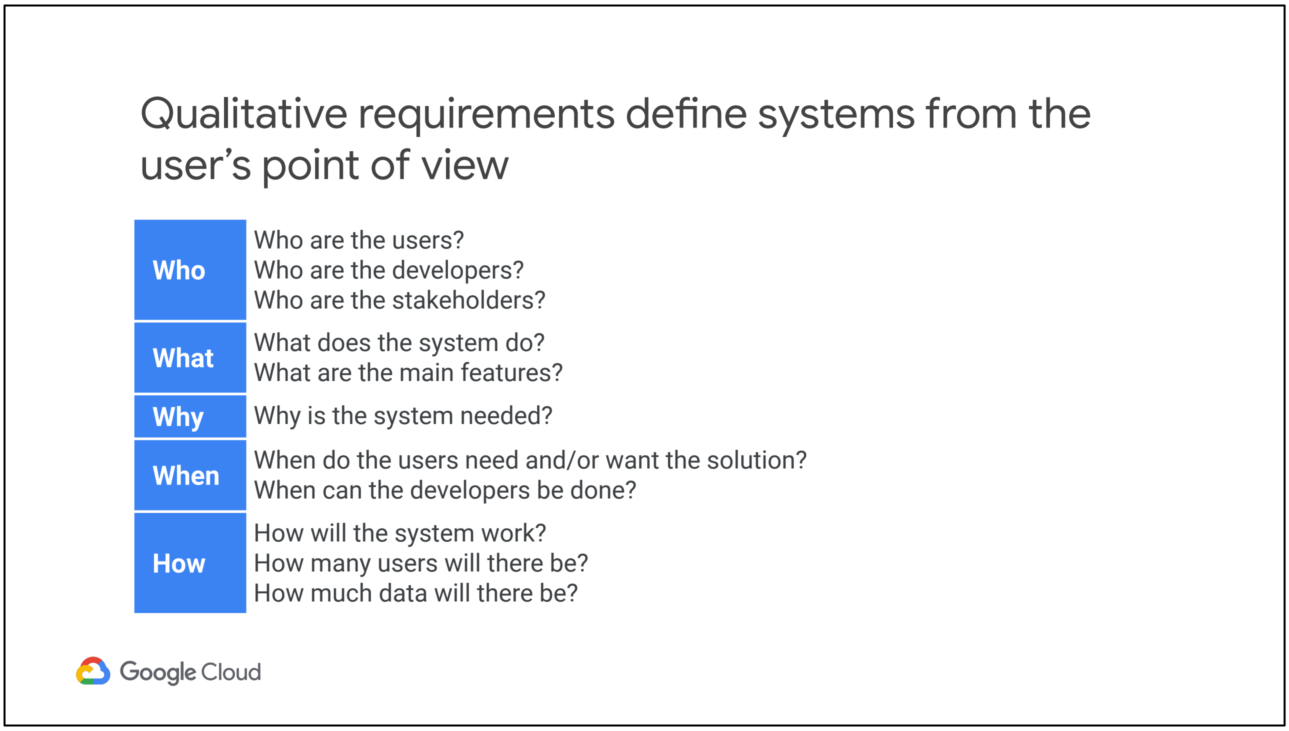


Agenda

- **Requirements, Analysis, and Design**
- Design Activity #2
- SLOs, SLIs, and SLAs
- Design Activity #3

Google Cloud

Let’s start talking about requirements, analysis, and design.

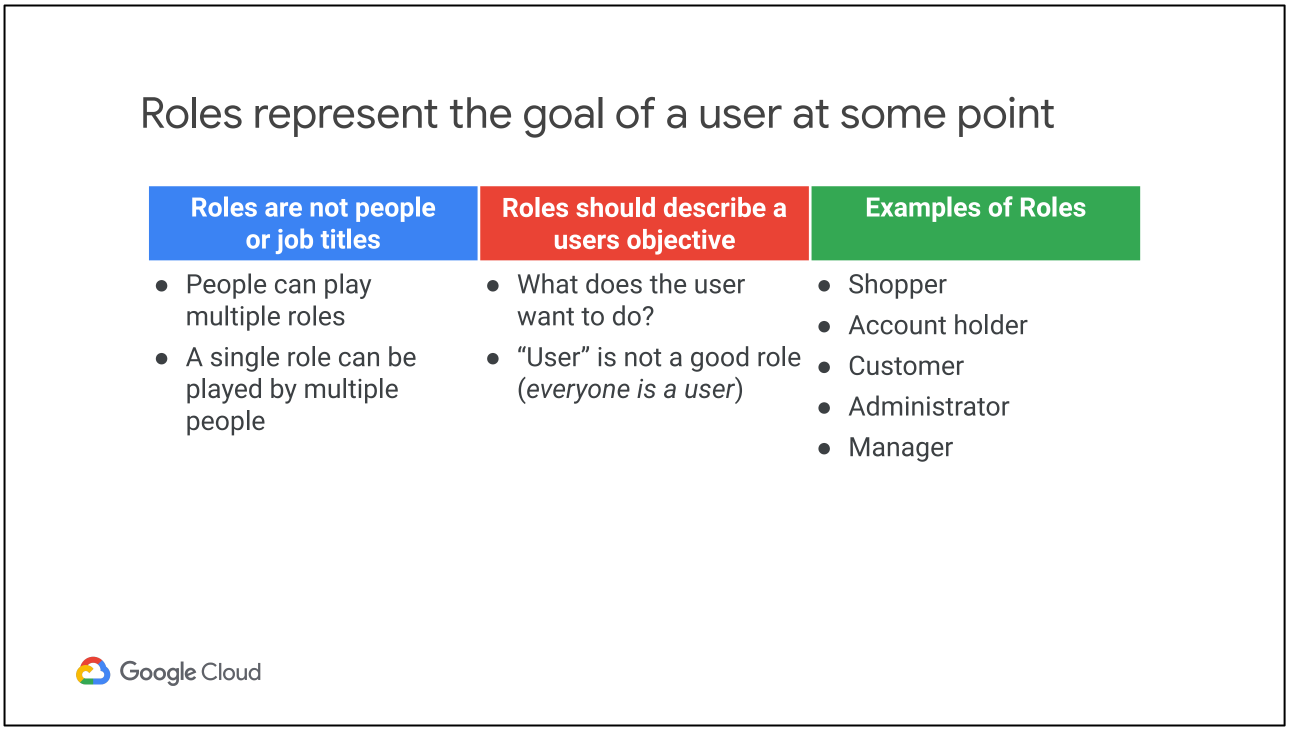


Qualitative requirements define systems from the user's point of view

Who	Who are the users? Who are the developers? Who are the stakeholders?
What	What does the system do? What are the main features?
Why	Why is the system needed?
When	When do the users need and/or want the solution? When can the developers be done?
How	How will the system work? How many users will there be? How much data will there be?

Google Cloud

Here we have questions that are a mixture of business and technical requirements. For example, "Why is the system needed" could be a business requirement. These are all high-level points, and requirements gathering would expand each question with more details.



Roles represent the goal of a user at some point

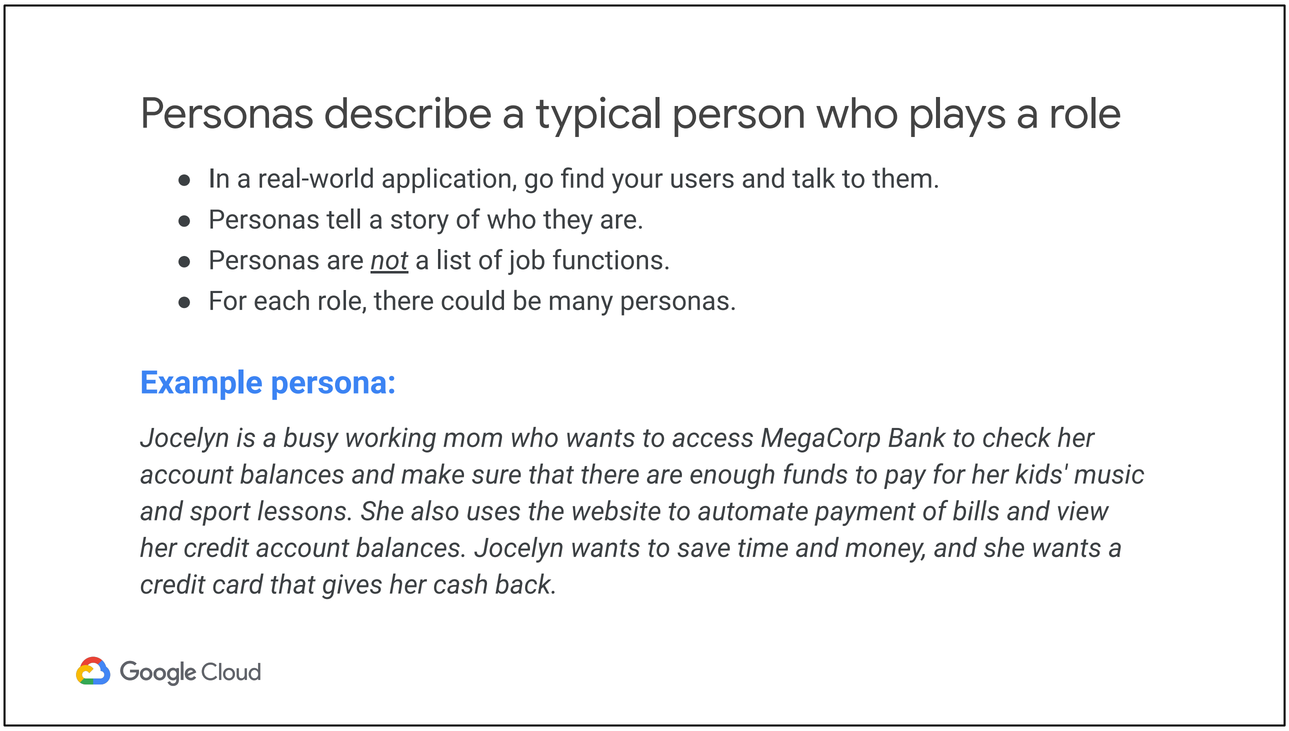
Roles are not people or job titles	Roles should describe a users objective	Examples of Roles
• People can play multiple roles • A single role can be played by multiple people	• What does the user want to do? • "User" is not a good role (*everyone is a user*)	• Shopper • Account holder • Customer • Administrator • Manager

Google Cloud

Roles help provide a context. They enable the analysis of a requirement in a particular context. It is important to note that a role is not necessarily a person. It is an actor on the system and could be another system such as a microservice client that is accessing another microservice.

To determine the roles, a typical process would be to:

- Brainstorm an initial set of roles: write as many roles as you can think of, with each role being a single user.
- Organize the initial set: here overlapping roles are identified and related roles are grouped together.
- Consolidate the roles: the aim here is to consolidate and condense the roles.
- Refine the roles including internal and external roles and the different usage patterns. Here extra information can be provided such as the users' level of expertise in the domain, or the frequency of use of the proposed software.



Personas describe a typical person who plays a role

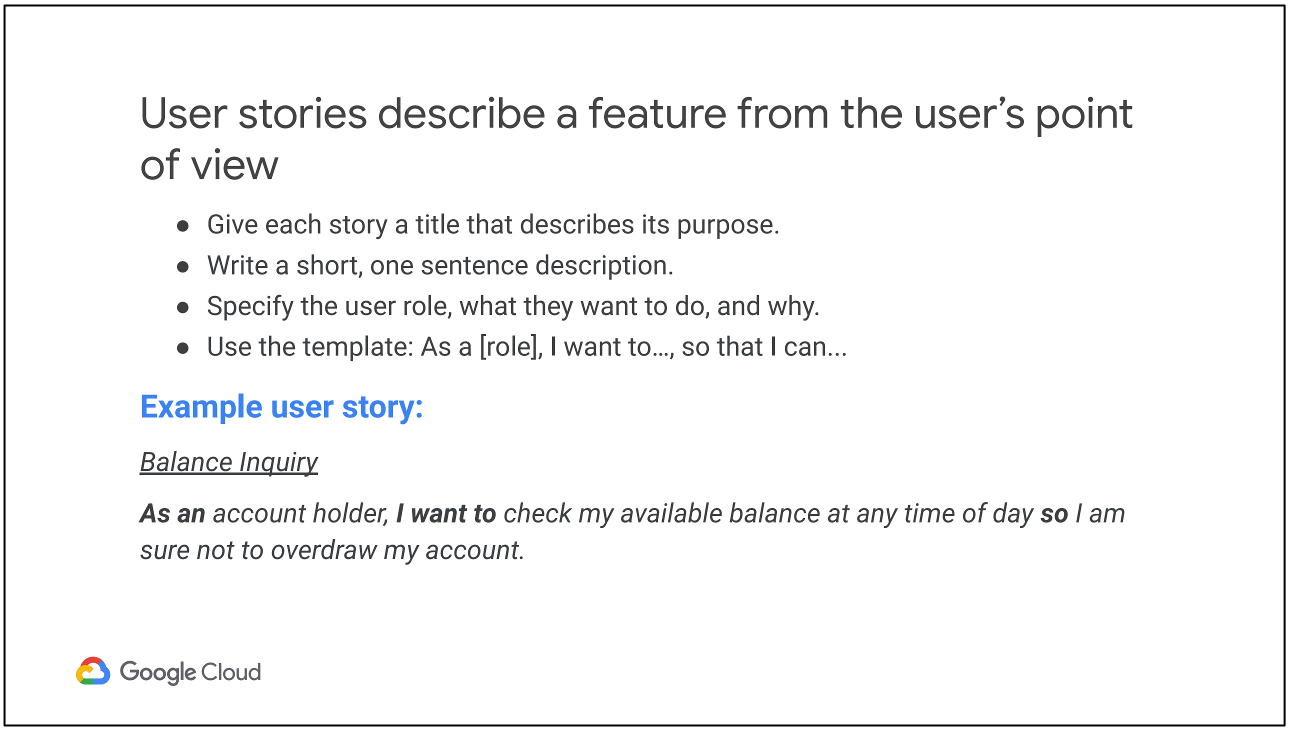
- In a real-world application, go find your users and talk to them.
- Personas tell a story of who they are.
- Personas are *not* a list of job functions.
- For each role, there could be many personas.

Example persona:

Jocelyn is a busy working mom who wants to access MegaCorp Bank to check her account balances and make sure that there are enough funds to pay for her kids' music and sport lessons. She also uses the website to automate payment of bills and view her credit account balances. Jocelyn wants to save time and money, and she wants a credit card that gives her cash back.

Google Cloud

Identifying user roles is a useful technique as part of the requirements-gathering process. An additional technique, in particular for more important roles, can be to create a persona for the role. A persona is an imaginary representation of a user role. The aim of the persona is to help developers think about the characteristics of users by personalizing them. Often a role has multiple personas. In the example above, when a question arises from a developer they can better answer that question by thinking, “What would Jocelyn want here?



User stories describe a feature from the user's point of view

- Give each story a title that describes its purpose.
- Write a short, one sentence description.
- Specify the user role, what they want to do, and why.
- Use the template: As a [role], I want to..., so that I can...

Example user story:

Balance Inquiry

***As an** account holder, **I want to** check my available balance at any time of day **so** I am sure not to overdraw my account.*

Google Cloud

User stories describe one thing a user wants the system to do. They are written in a structured way, typically using the form:
As a [type of user] **I want to** [do something] **so that I can** [get some benefit]

Another commonly used form is:
Given [some context] **When I** [do something] **Then** [this should happen]

User stories provide a clear way of agreeing requirements with a customer/end user. The INVEST criteria are often used to evaluate good user stories. They are evaluated to ensure they are:

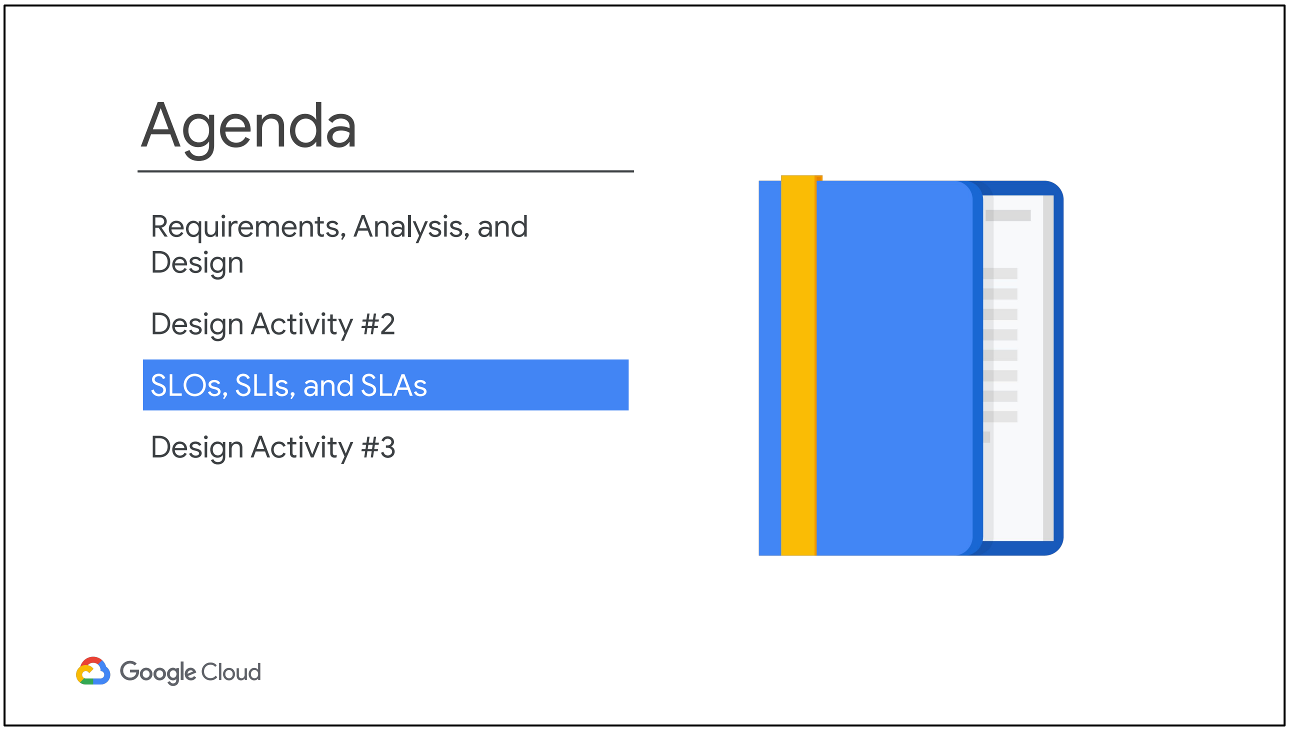
- **I**ndependent: A story should be independent to prevent problems with prioritization and planning.
- **N**egotiable: They are not written contracts, but are used to stimulate discussion between customer and developers until there is a clear agreement. They aid collaboration.
- **V**aluable: Stories should provide value to users.Think about outcomes and impact, not outputs and deliverables.
- **E**stimatable: The story must be estimable. If it is not, it often indicates missing details or the story is too large.
- **S**mall: Good stories should be small. This helps keep scope small and therefore less ambiguous and supports fast feedback from users.
- **T**estable: Stories must be testable so that developers can verify that the story has been implemented correctly and validate when the requirement has been met/is done.

Activity 2: Analyzing your case study

Refer to your Design and Process Workbook.

- Refine the roles you listed in Activity 1.
- Write personas for each role.
- Write user stories for the main features of your case study.

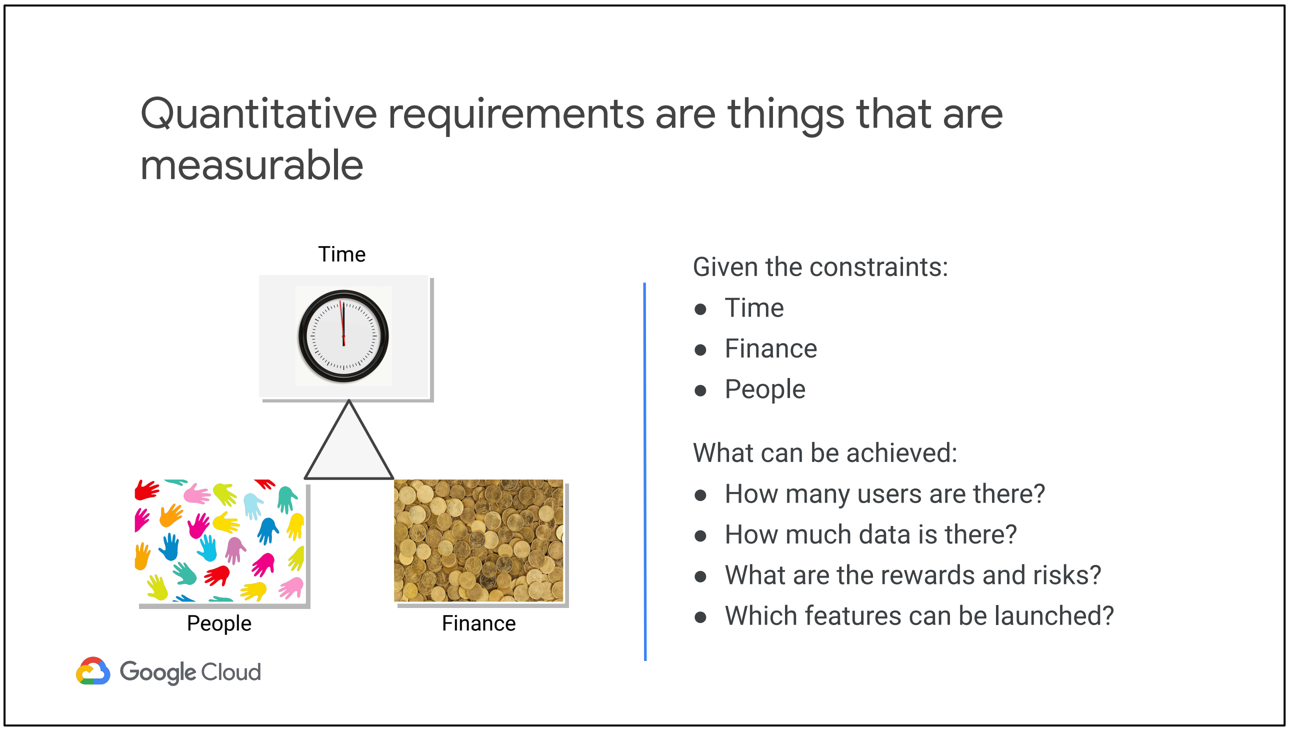
Google Cloud



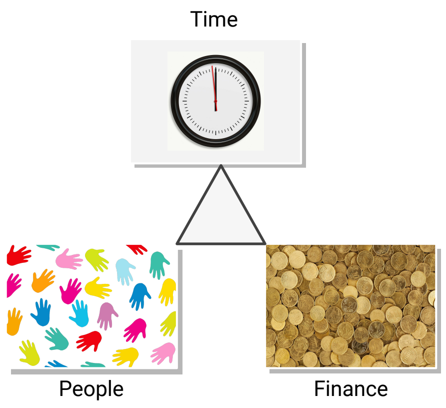
Agenda

Google Cloud

We now move on to consider how to measure whether the technical and business requirements have been met.



Quantitative requirements are things that are measurable



Given the constraints:

- Time
- Finance
- People

What can be achieved:

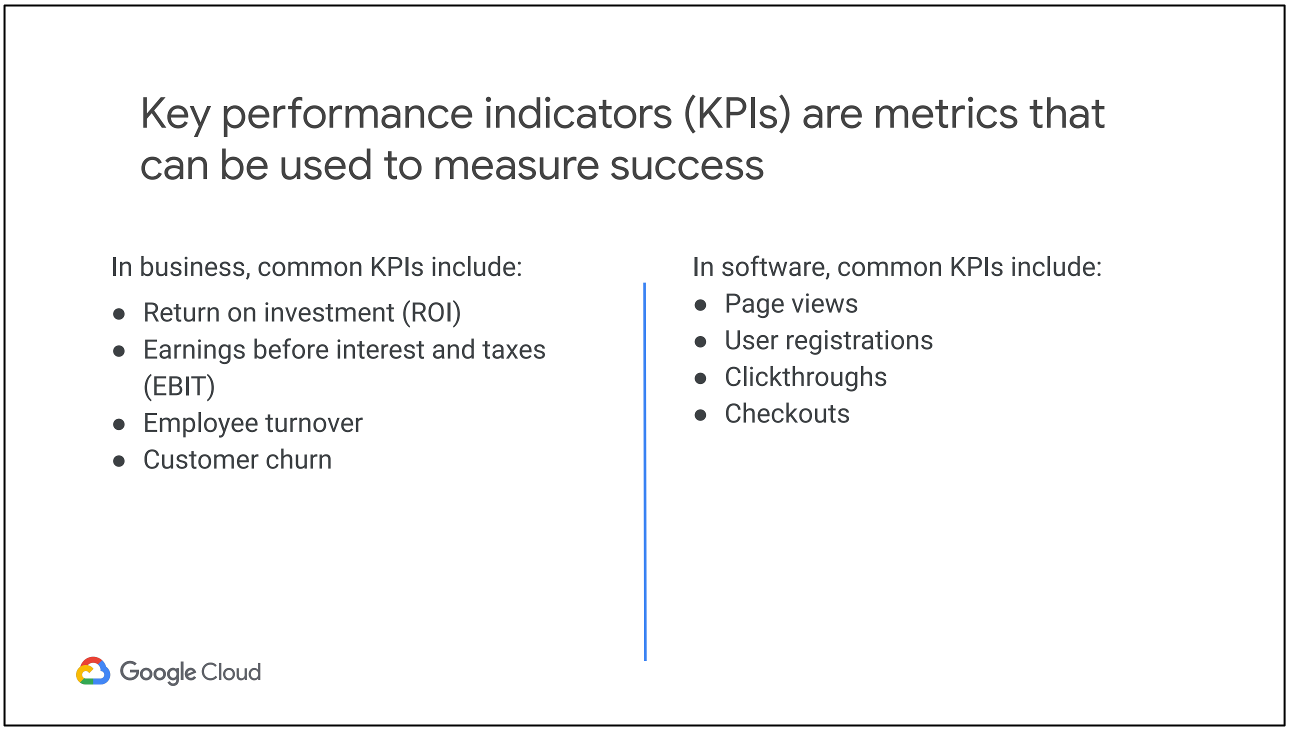
- How many users are there?
- How much data is there?
- What are the rewards and risks?
- Which features can be launched?

To manage a service well, it is important to understand which behaviors matter and how to measure and evaluate these behaviors.

The type of system being evaluated determines the data that can be measured. For example, for user-facing systems, was a request responded to? (availability), how long did it take to respond? (latency), how many requests can be handled? (throughput) are typically important factors.

For data storage systems, how long does it take to read and write data? (latency), is the data there when we need it? (availability), if there is a failure, do we lose any data (durability) are important factors.

The key to all these items is that the questions can be answered with data gathered from the services.



Key performance indicators (KPIs) are metrics that can be used to measure success

In business, common KPIs include:

- Return on investment (ROI)
- Earnings before interest and taxes (EBIT)
- Employee turnover
- Customer churn

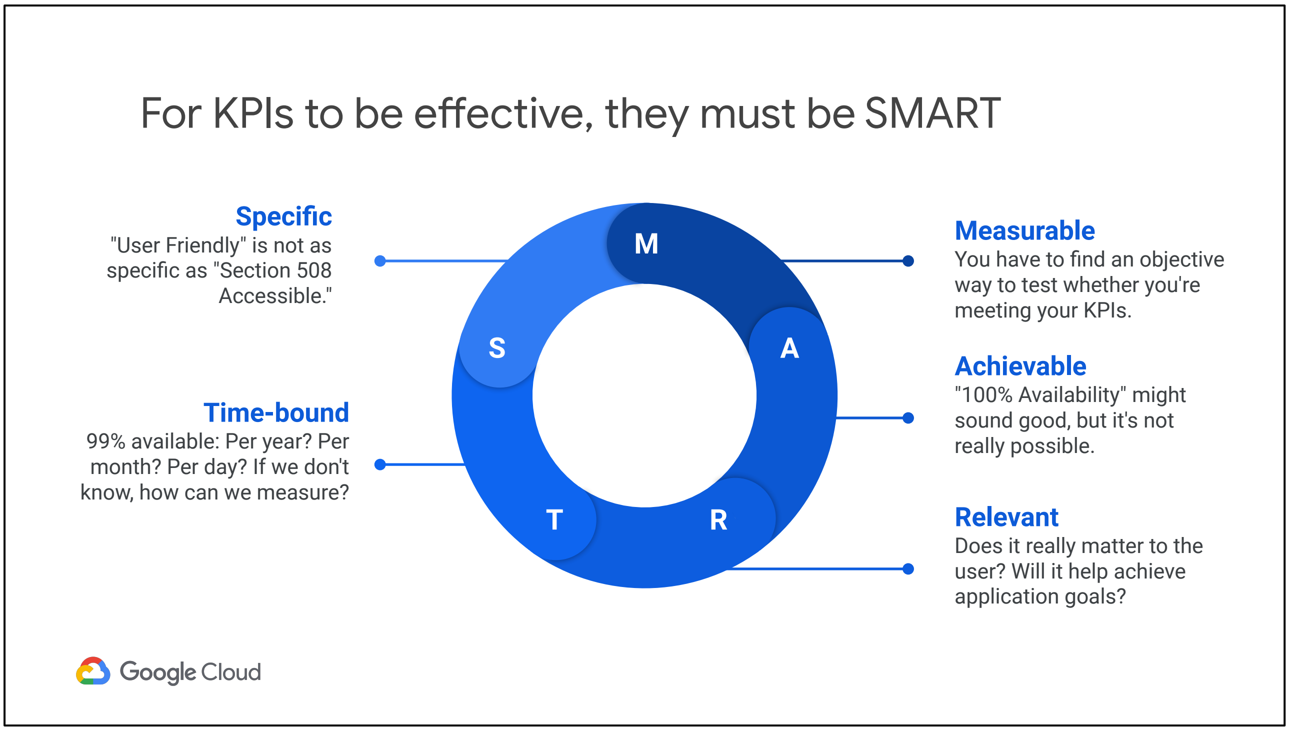
In software, common KPIs include:

- Page views
- User registrations
- Clickthroughs
- Checkouts

Google Cloud

Business decision makers want to measure the value of projects. This enables them to better support the most valuable projects and not waste resources on those that are not beneficial. A common way to measure success is to use KPIs. KPIs can be categorized as business KPIs and technical KPIs.

Business KPIs are a formal way of measuring what the business values, such as ROI, in relation to a project or service,. Software KPIs can consider aspects such as how effective the software is through user registrations, number of checkouts, etc. These KPIs should also be closely aligned with business objectives. As an architect, it is important that you understand how the business measures success of the systems that you design.



For KPIs to be effective, they must be SMART

Specific
"User Friendly" is not as specific as "Section 508 Accessible."

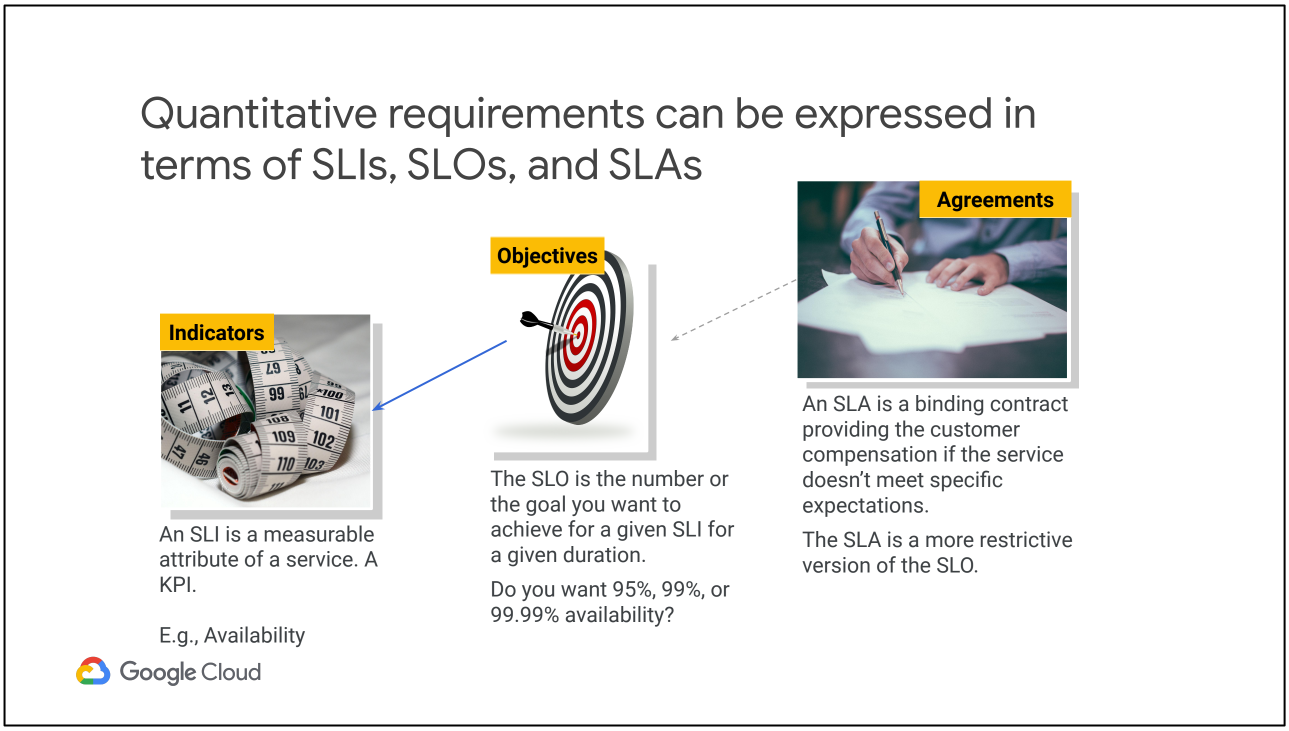
Measurable
You have to find an objective way to test whether you're meeting your KPIs.

Achievable
"100% Availability" might sound good, but it's not really possible.

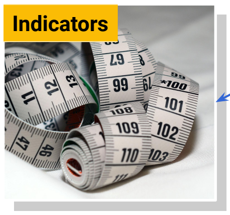
Relevant
Does it really matter to the user? Will it help achieve application goals?

Time-bound
99% available: Per year? Per month? Per day? If we don't know, how can we measure?

It is important to note that a KPI is not the same thing as a goal or objective. The goal is the outcome or result you want to achieve. The KPI is a metric that indicates whether you are on track to achieve the goal. To be the most effective, KPIs need an accompanying goal. This should be the starting point in defining KPIs. Then, for each goal define the KPIs that will allow you to monitor and measure progress. For each KPI, define targets for what success looks like. Monitoring KPIs against goals is important to achieving success and allows readjustment based on feedback.



Quantitative requirements can be expressed in terms of SLIs, SLOs, and SLAs



An SLI is a measurable attribute of a service. A KPI.

E.g., Availability

The SLO is the number or the goal you want to achieve for a given SLI for a given duration.

Do you want 95%, 99%, or 99.99% availability?

An SLA is a binding contract providing the customer compensation if the service doesn't meet specific expectations.

The SLA is a more restrictive version of the SLO.

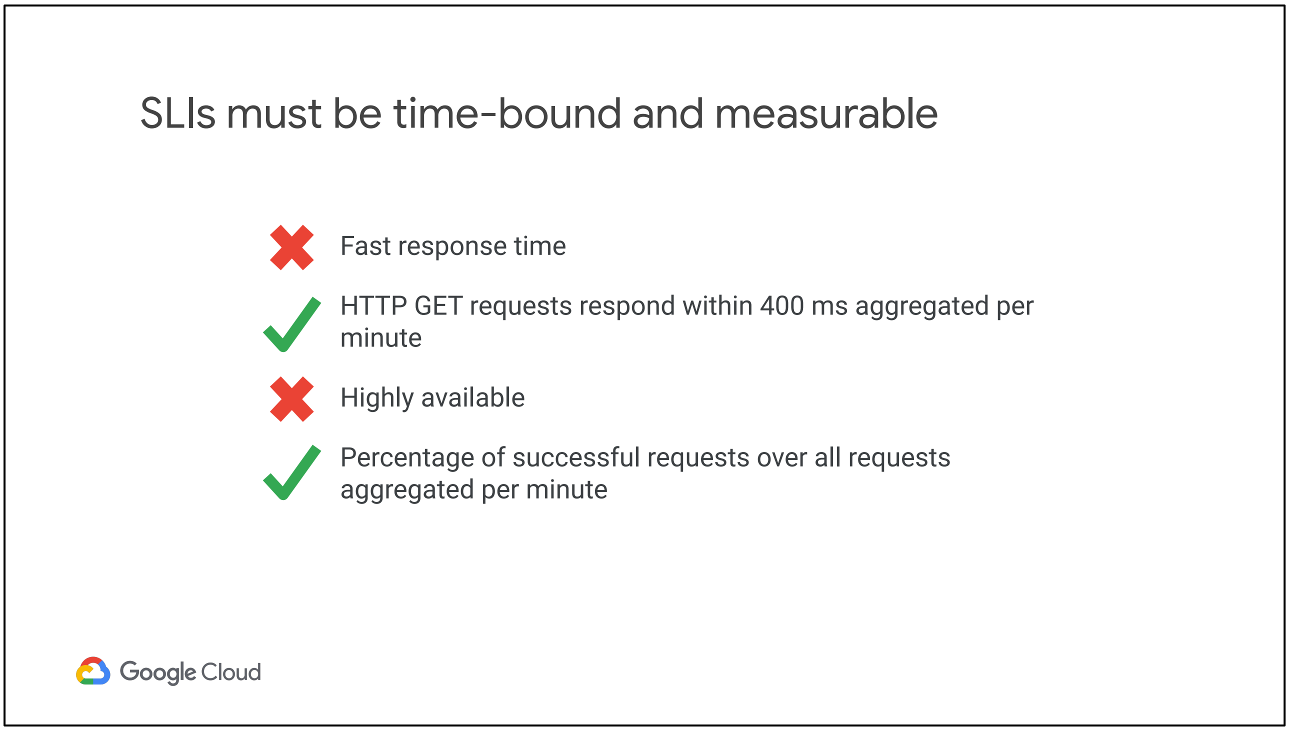
Google Cloud

Here we introduce service level terminology. To provide a given level of service to customers, it is important to define service level indicators (SLI), objectives (SLO), and agreements (SLA). These are measurements that describe basic properties of the metrics to measure, the values those metrics should read, and how to react if the metrics cannot be met.

Service level indicator is a quantitative measure of some aspect of the level of service being provided. Examples include throughput, latency, and error rate.

Service level objective is an agreed-upon target or range of values for a service level that is measured by an SLI. It is normally stated in the form SLI ≤ target or lower bound ≤ SLI ≤upper bound. An example of a SLO is that the average latency of HTTP requests for our service should be less than 100 milliseconds.

Service level agreement is an agreement between a service provider and consumer. They define responsibilities for delivering a service and consequences when these responsibilities are not met.



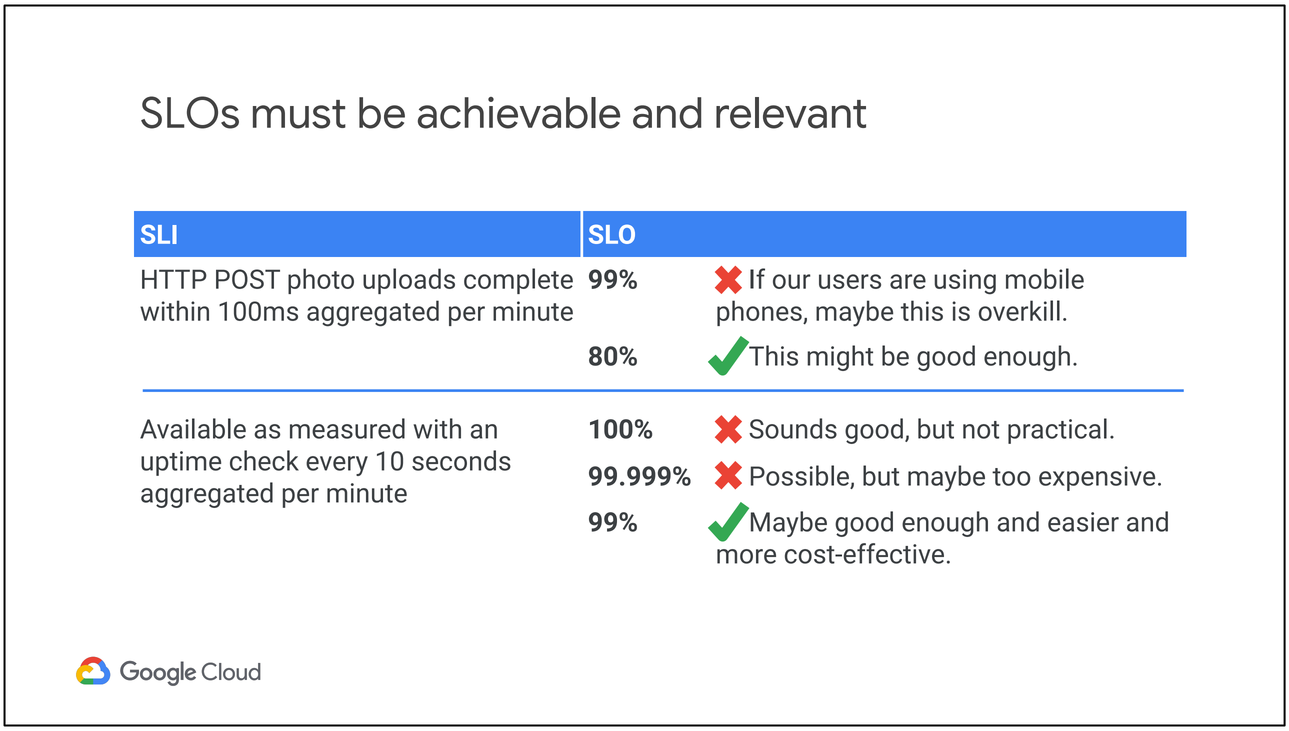
SLIs must be time-bound and measurable

- ✗ Fast response time
- ✓ HTTP GET requests respond within 400 ms aggregated per minute
- ✗ Highly available
- ✓ Percentage of successful requests over all requests aggregated per minute

Google Cloud

Understanding what users want from a service will help inform the selection of indicators. Indicators be measurable, and the way they are aggregated needs careful consideration. For example, consider requests per second to a service. How is the value calculated: by measurements obtained once per second or by averaging requests over a minute? The once-per-second measurement may hide high request rates that occur in bursts of a few seconds.

For example, consider a service that receives 1000 requests/s on even numbered seconds and 0 requests on odd numbered seconds. The average requests/s could be reported over a minute as 500. However the reality is that the load at times is twice as large as the average. Similar averages can mask user experience when used for metrics like latency. They can mask requests that take a lot longer to respond than the average. It is better to use percentiles for such metrics where a high order percentile, such as 99%, shows worst case values, while the 50th percentile will indicate a typical case.



SLOs must be achievable and relevant

SLI	SLO	
HTTP POST photo uploads complete within 100ms aggregated per minute	**99%**	✗ If our users are using mobile phones, maybe this is overkill.
	80%	✓ This might be good enough.
Available as measured with an uptime check every 10 seconds aggregated per minute	**100%**	✗ Sounds good, but not practical.
	99.999%	✗ Possible, but maybe too expensive.
	99%	✓ Maybe good enough and easier and more cost-effective.

Google Cloud

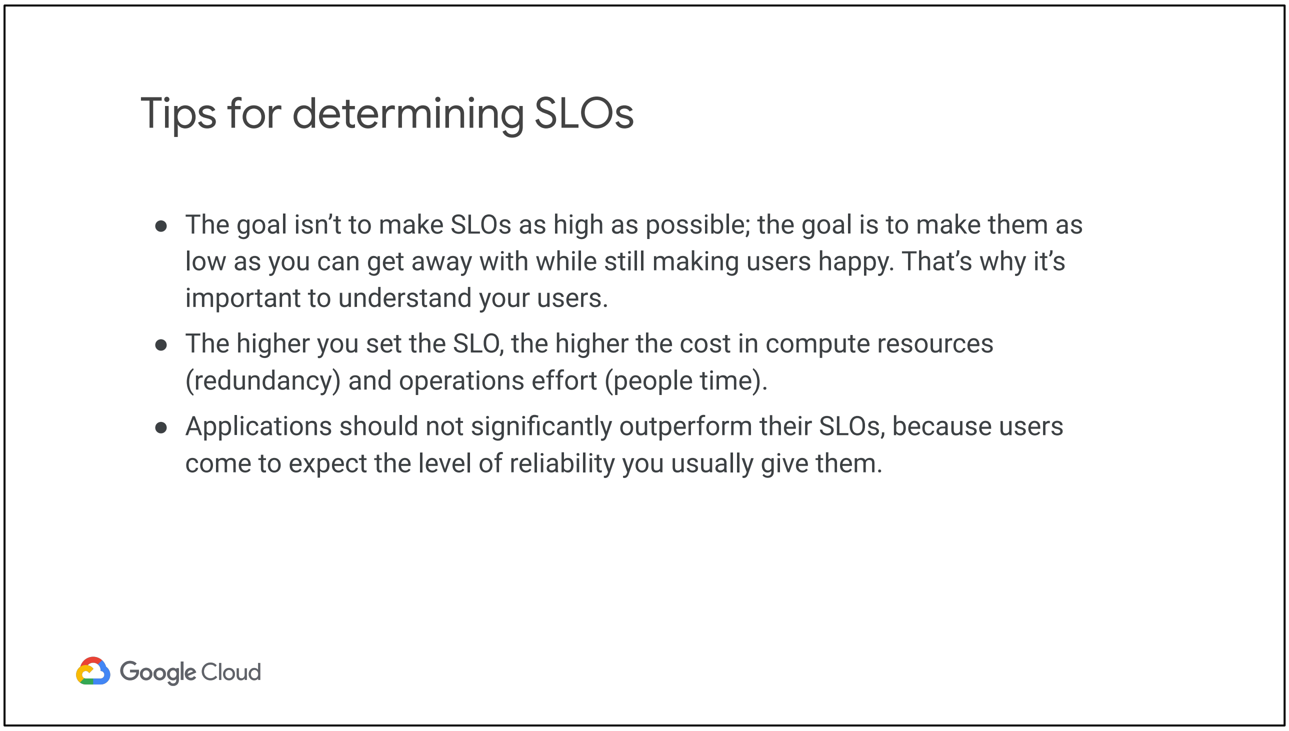
The relevancy of SLOs is vital: you want objectives that help or improve the user experience. It is easy to define SLOs based around what is easy to measure rather than what is useful. For clarity, SLOs should specify how they are measured and the conditions when they are valid. As the slide above states, it is unrealistic as well as undesirable to have SLOs with a 100% target. Such a target results in expensive, overly conservative solutions that are still unlikely to reach the SLO. It is better to track the rate at which SLOs are missed and work to improve this.

It is often ok to specify multiple SLOs. Consider the following:

99% of HTTP GET calls will complete in less than 100ms

This is a valid SLO but it may be the case that the shape of the performance curve is important. In this case, the SLO could be written as follows:

90% of HTTP GET calls will complete in less than 50ms
99% of HTTP GET calls will complete in less than 100ms
99.9% of HTTP GET calls will complete in less than 500ms



Tips for determining SLOs

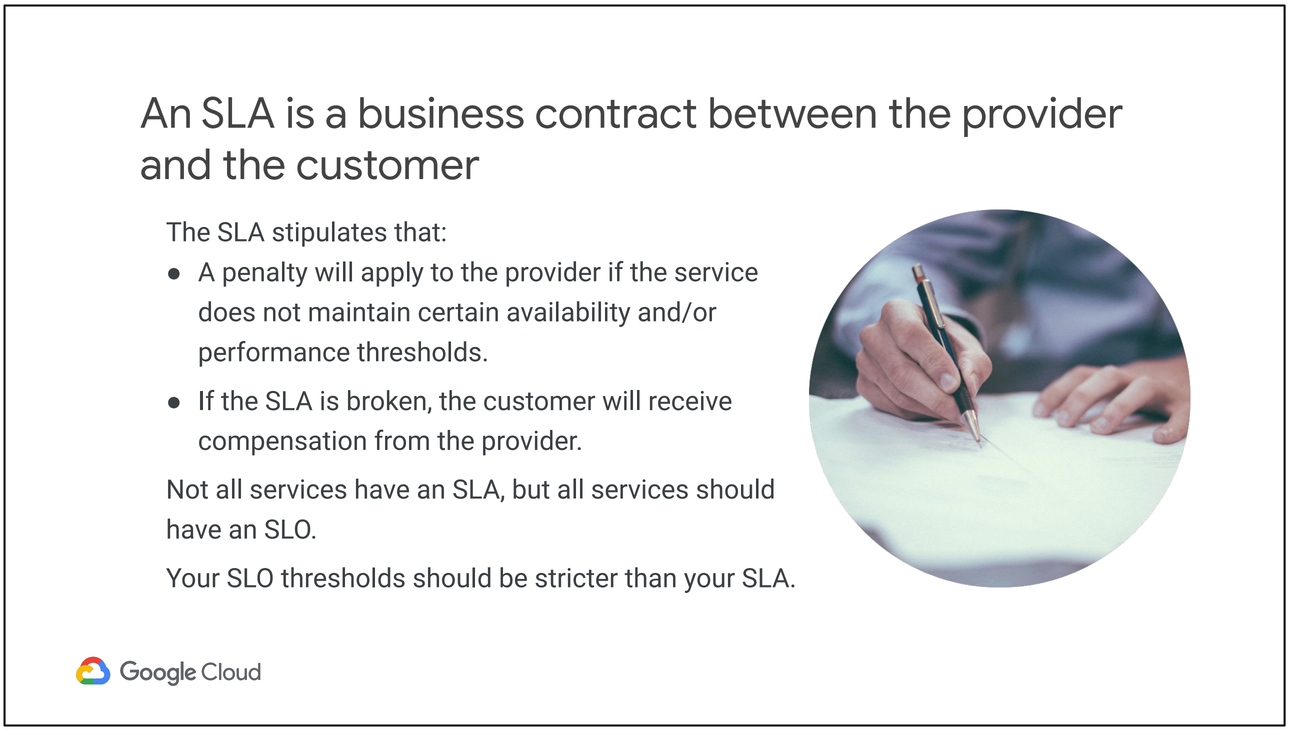
- The goal isn't to make SLOs as high as possible; the goal is to make them as low as you can get away with while still making users happy. That's why it's important to understand your users.
- The higher you set the SLO, the higher the cost in compute resources (redundancy) and operations effort (people time).
- Applications should not significantly outperform their SLOs, because users come to expect the level of reliability you usually give them.

Google Cloud

Selecting SLOs has both product and business implications. Often tradeoffs need to be made based on constraints such as staff, time to market, and funding. As the slide states, the aim is to keep users happy, not to have an SLO that requires heroic efforts to maintain. Some tips on selecting SLOs include:

- Keep them simple: More complex SLIs can obscure important changes in performance.
- Avoid absolute values: To have an SLO that states 100% availability is unrealistic. Such an SLO increases the time to build, complexity, and cost to operate, and in most cases is highly unlikely to be required.
- Minimize SLOs: A common mistake is to have too many SLOs. The recommendation is to have just enough SLOs to give coverage of the key system attributes.
- Do not make them too high: It is better to have lower SLOs to begin with and tighten them over time as you learn about the system instead of defining those that are unattainable and require a significant effort and cost to try and achieve.

In summary, good SLOs should reflect what the users care about. They work as a forcing function for development teams. A poor SLO will result in a significant amount of wasted work if it is too ambitious, or a poor product if it is too relaxed.



An SLA is a business contract between the provider and the customer

The SLA stipulates that:

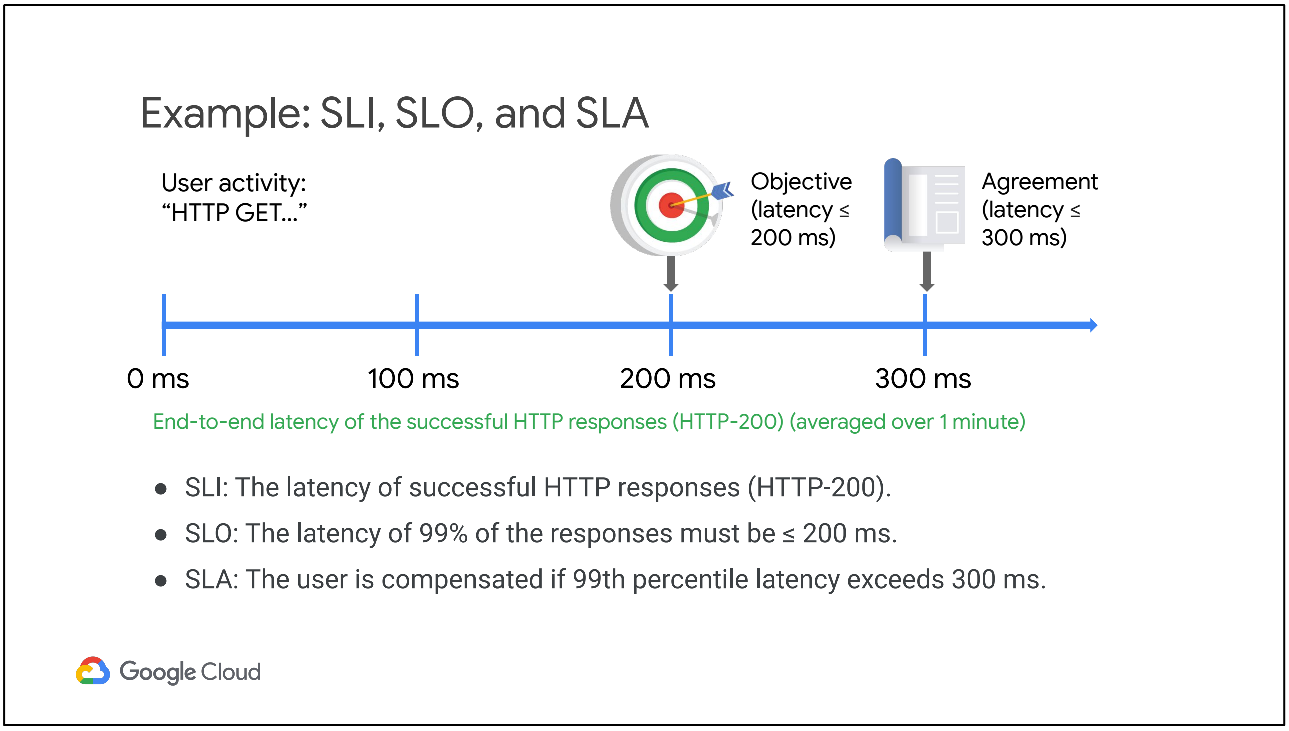
- A penalty will apply to the provider if the service does not maintain certain availability and/or performance thresholds.
- If the SLA is broken, the customer will receive compensation from the provider.

Not all services have an SLA, but all services should have an SLO.

Your SLO thresholds should be stricter than your SLA.

Google Cloud

As with SLOs, it is better to be conservative with SLAs because it is difficult to change or remove SLAs that offer little value or cause a large amount of work. In addition, they can have a financial implication through compensation to the customer. To provide protection and some level of safety, an SLA should have a threshold that is lower than the SLO.



Example: SLI, SLO, and SLA

User activity: "HTTP GET..."
Objective (latency ≤ 200 ms)
Agreement (latency ≤ 300 ms)
0 ms
100 ms
200 ms
300 ms
End-to-end latency of the successful HTTP responses (HTTP-200) (averaged over 1 minute)

- SLI: The latency of successful HTTP responses (HTTP-200).
- SLO: The latency of 99% of the responses must be ≤ 200 ms.
- SLA: The user is compensated if 99th percentile latency exceeds 300 ms.

Google Cloud

In the above example for the SLO, it is important that the percentiles are monitored as if only the average is considered. Then to an SRE team, the SLO could be met but the SLA is not.

Activity 3: Defining SLIs and SLOs

Refer to your Design and Process Workbook.

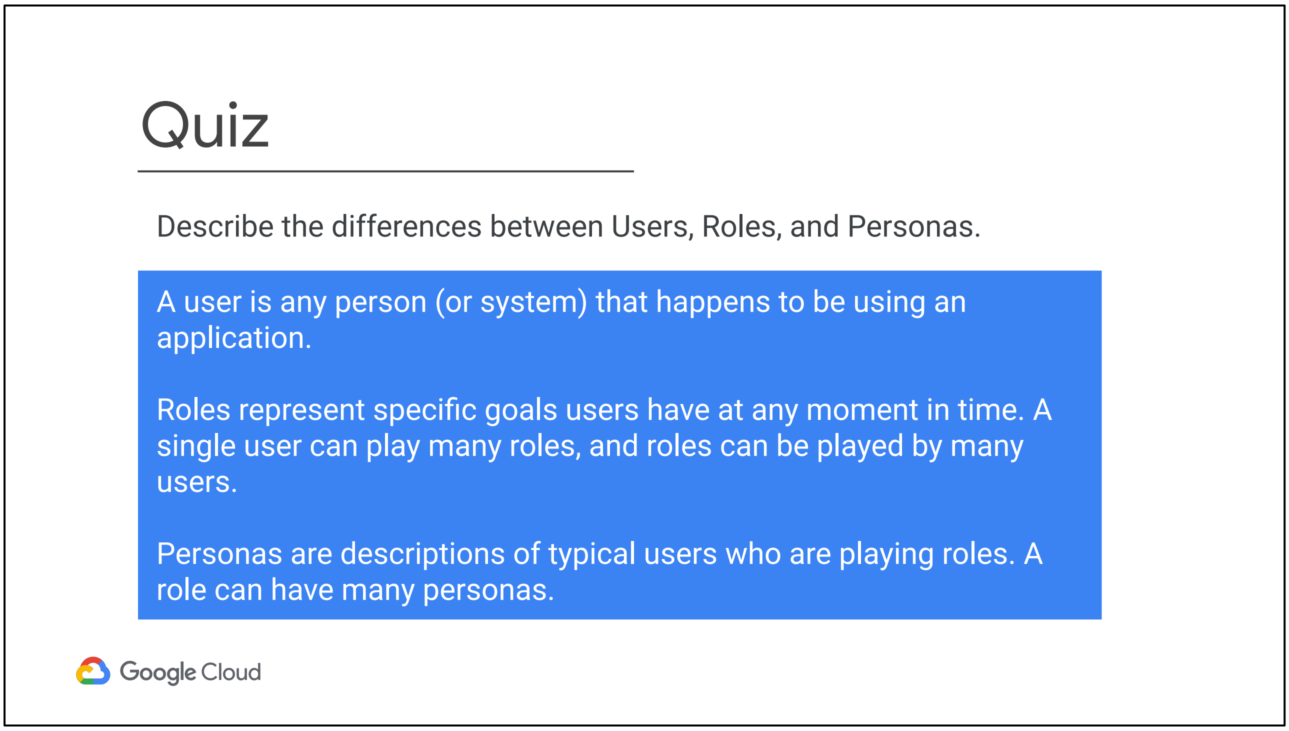
- Write SLIs and SLOs for your case study features.

Google Cloud

Quiz

Describe the differences between Users, Roles, and Personas.

Google Cloud


Quiz

Describe the differences between Users, Roles, and Personas.

A user is any person (or system) that happens to be using an application.

Roles represent specific goals users have at any moment in time. A single user can play many roles, and roles can be played by many users.

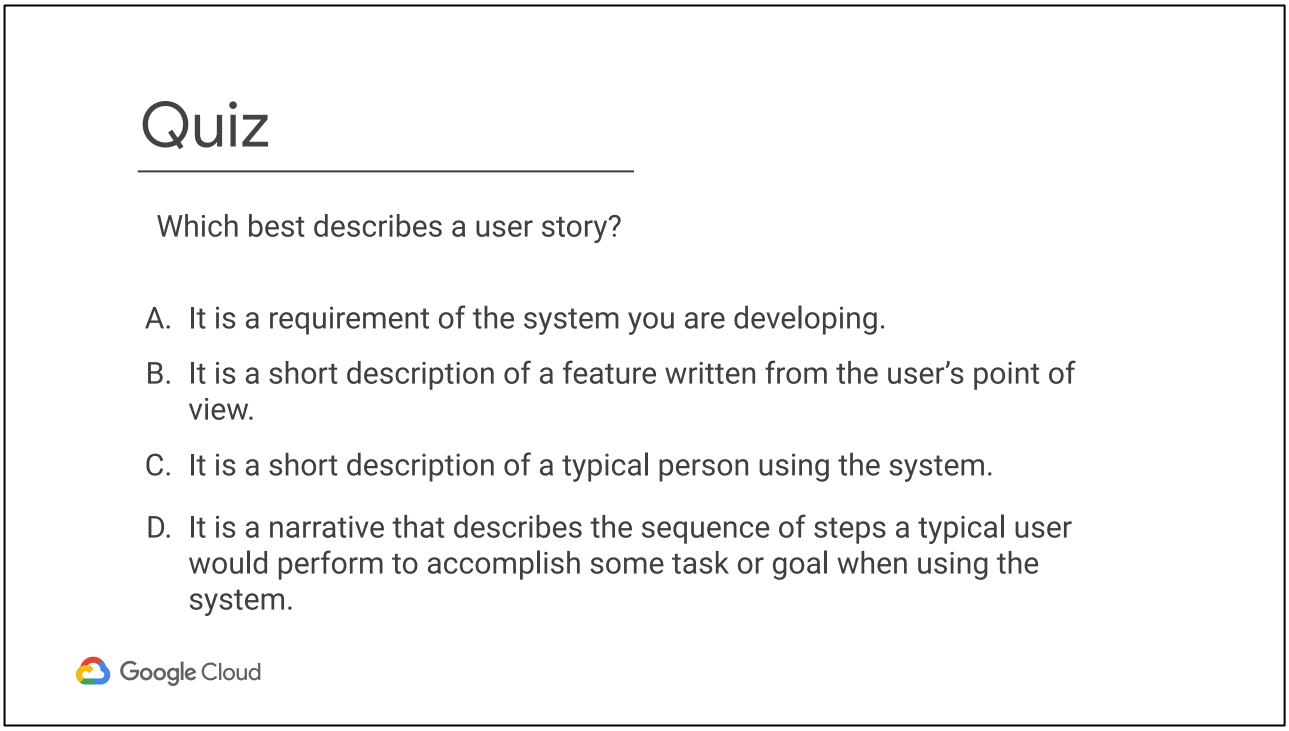
Personas are descriptions of typical users who are playing roles. A role can have many personas.

Google Cloud

A user is any person (or system) that happens to be using an application.

Roles represent specific goals users have at any moment in time. A single user can play many roles, and roles can be played by many users.

Personas are descriptions of typical users who are playing roles. A role can have many personas.



Quiz

Which best describes a user story?

A. It is a requirement of the system you are developing.

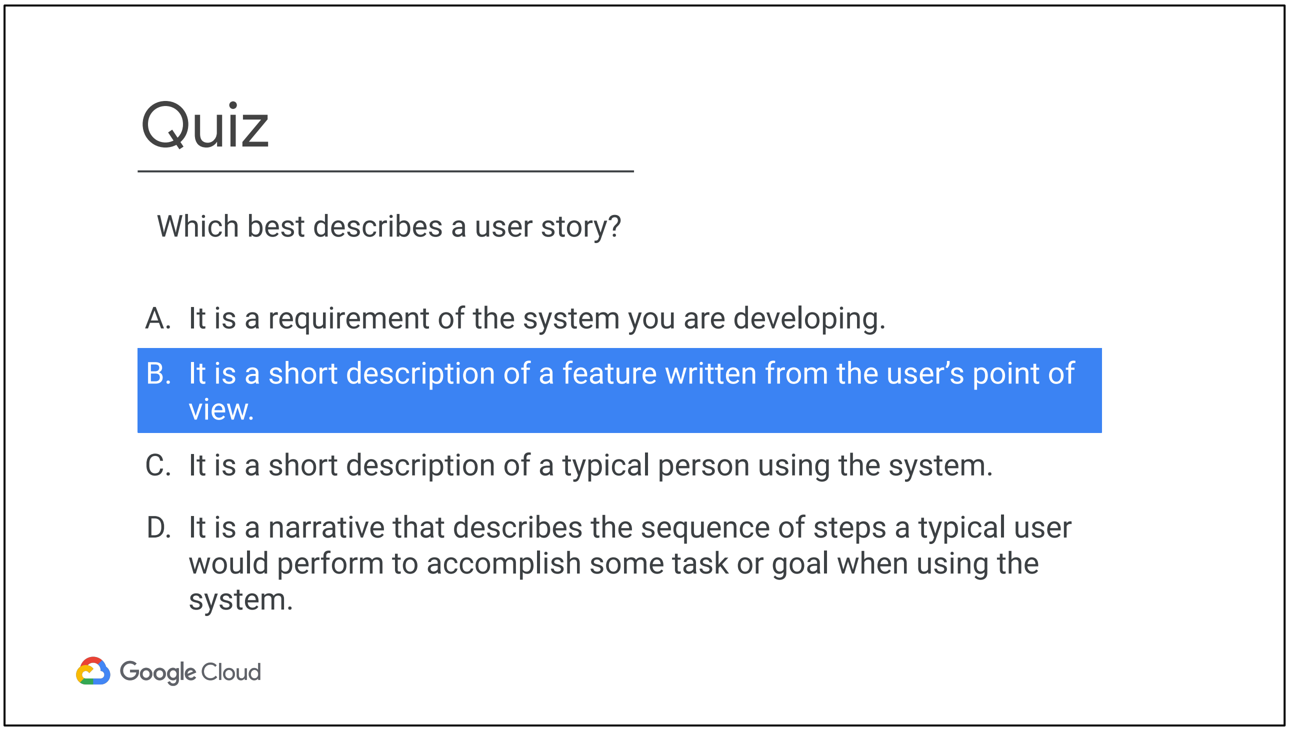
B. It is a short description of a feature written from the user's point of view.

C. It is a short description of a typical person using the system.

D. It is a narrative that describes the sequence of steps a typical user would perform to accomplish some task or goal when using the system.

Google Cloud

What best describes a user story? Choose the best answer from the following choices.



Quiz

Which best describes a user story?

A. It is a requirement of the system you are developing.

B. [✓ It is a short description of a feature written from the user's point of view.]

C. It is a short description of a typical person using the system.

D. It is a narrative that describes the sequence of steps a typical user would perform to accomplish some task or goal when using the system.

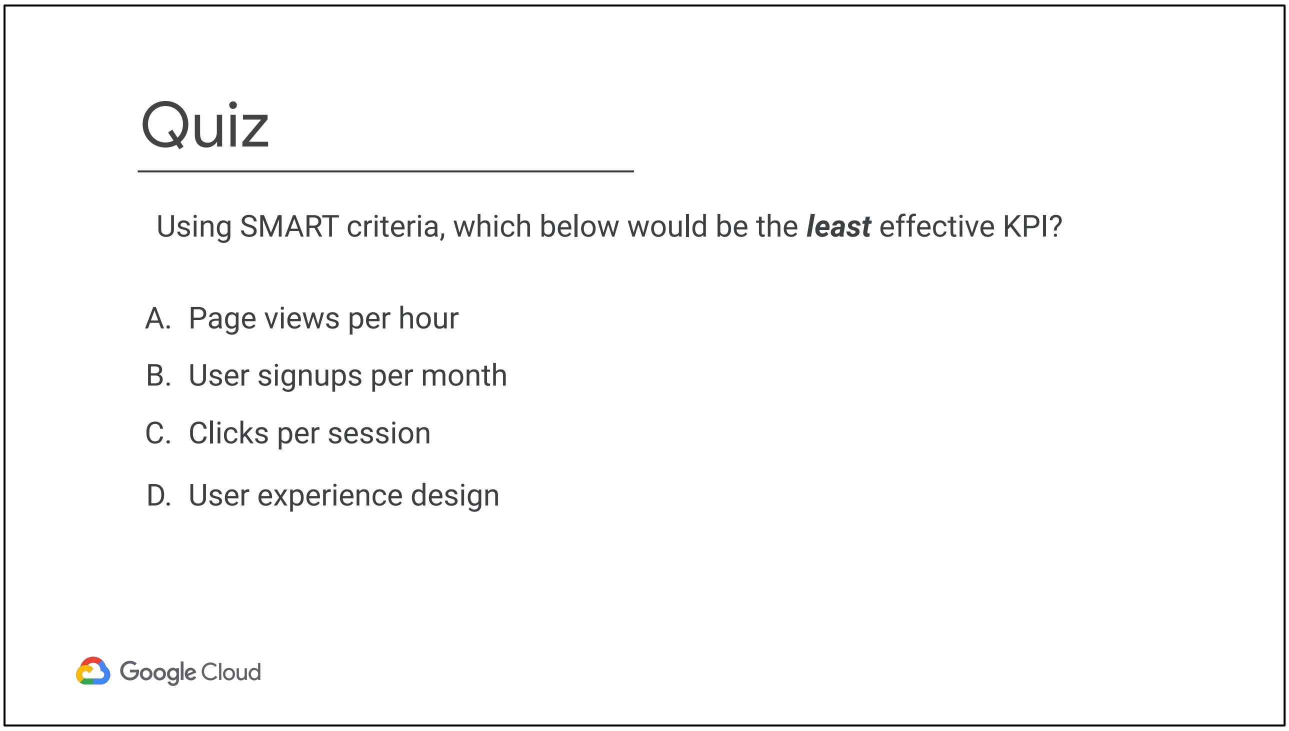
Google Cloud

Answer A is not correct because, while a user story is a requirement, this answer is more general than the correct answer and makes no mention of the requirement being from the user's point of view.

B is the correct answer because it describes a feature from the user's point of view.

Answer C is not correct because this answer mentions a typical person, but a user story always describes a user-facing feature. This answer more accurately describes a role.

Answer D is not correct. A user story describes the *what,* not the *how*. A description of a sequence of steps is more about describing how something should be achieved rather than what is required.



Quiz

Using SMART criteria, which below would be the ***least*** effective KPI?

A. Page views per hour

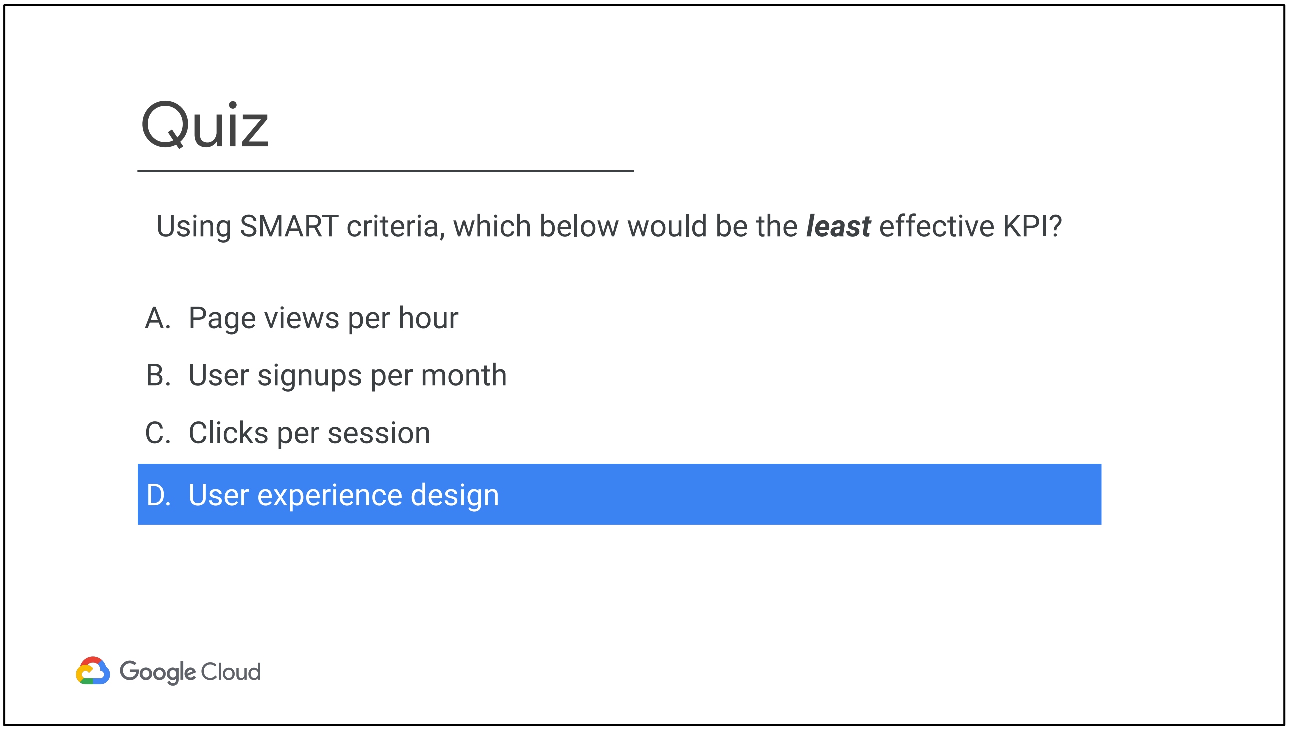
B. User signups per month

C. Clicks per session

D. User experience design

Google Cloud

Using SMART criteria, which answer below would be the least effective KPI?



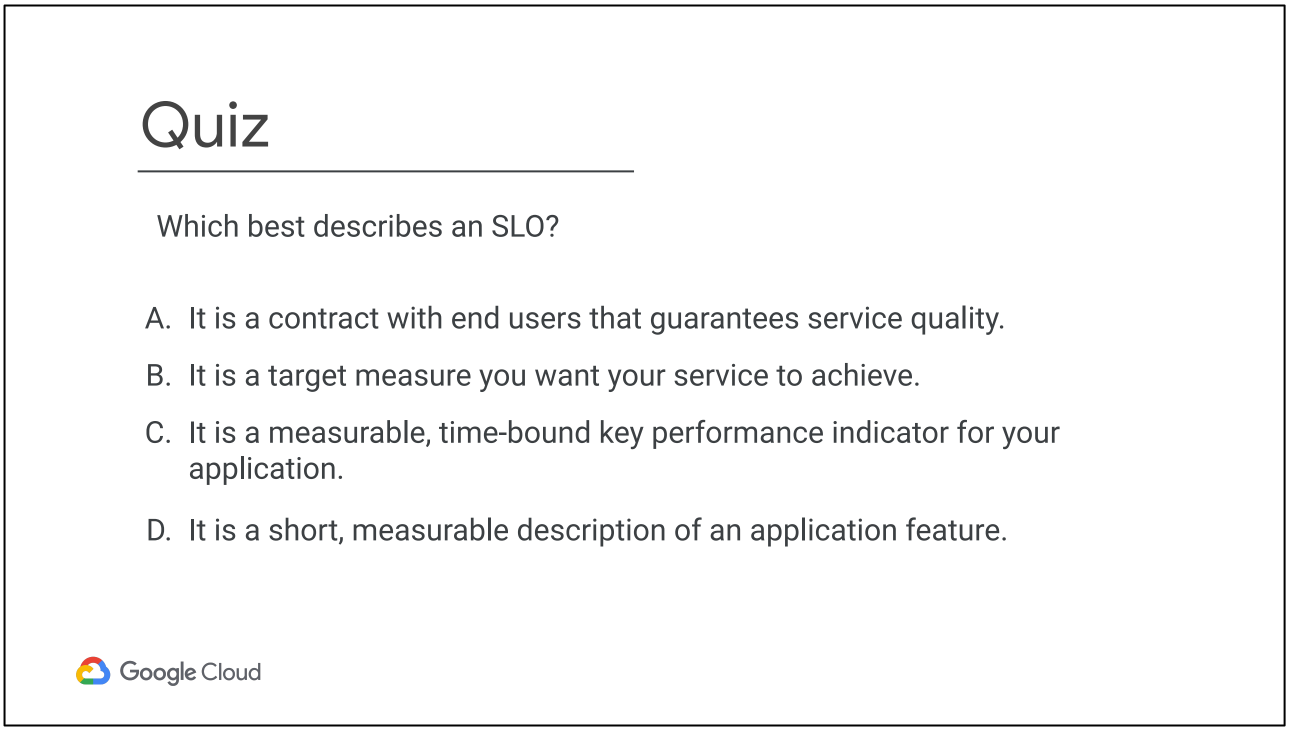
Quiz

Using SMART criteria, which below would be the ***least*** effective KPI?

A. Page views per hour
B. User signups per month
C. Clicks per session
[✓ D. User experience design]

Google Cloud

Answers A to C all meet the five SMART criteria. The correct answer is D. User experience design is not measurable or time bound and so would not make a relevant KPI.



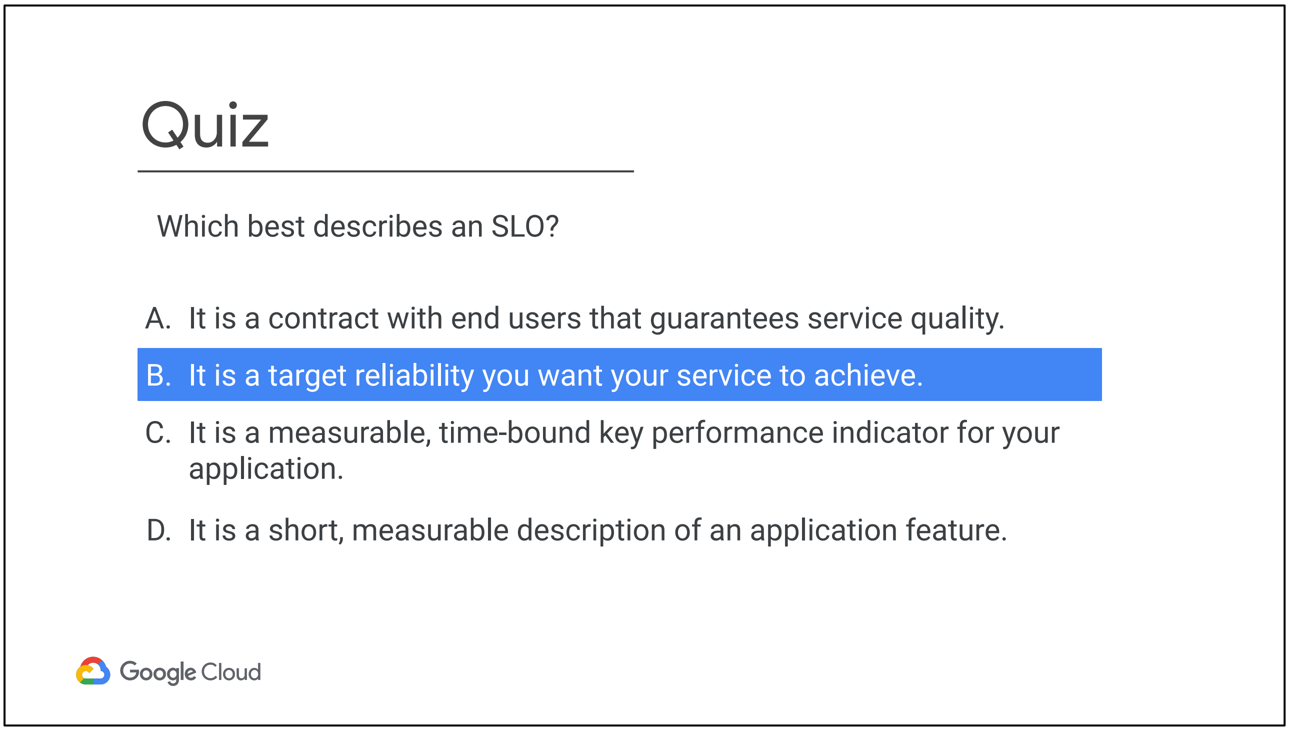
Quiz

Which best describes an SLO?

A. It is a contract with end users that guarantees service quality.
B. It is a target measure you want your service to achieve.
C. It is a measurable, time-bound key performance indicator for your application.
D. It is a short, measurable description of an application feature.

Google Cloud

Which of the answers best describes an SLO?



Quiz

Which best describes an SLO?

A. It is a contract with end users that guarantees service quality.

[✓ B. It is a target reliability you want your service to achieve.]

C. It is a measurable, time-bound key performance indicator for your application.

D. It is a short, measurable description of an application feature.

Google Cloud

Answer A is not correct. The description is more of an SLA; although with the SLA there are still no guarantees, just penalties if the agreement is not met.

The correct answer is B. An SLO is the agreed-upon target for a measurement or range of values for a service. Reliability could be one of these.

C is not correct. This answer describes a KPI which is the metric that is used in the SLO.

D is not correct. It is a generic phrase that is ambiguously related to SLO, SLI, SLA.

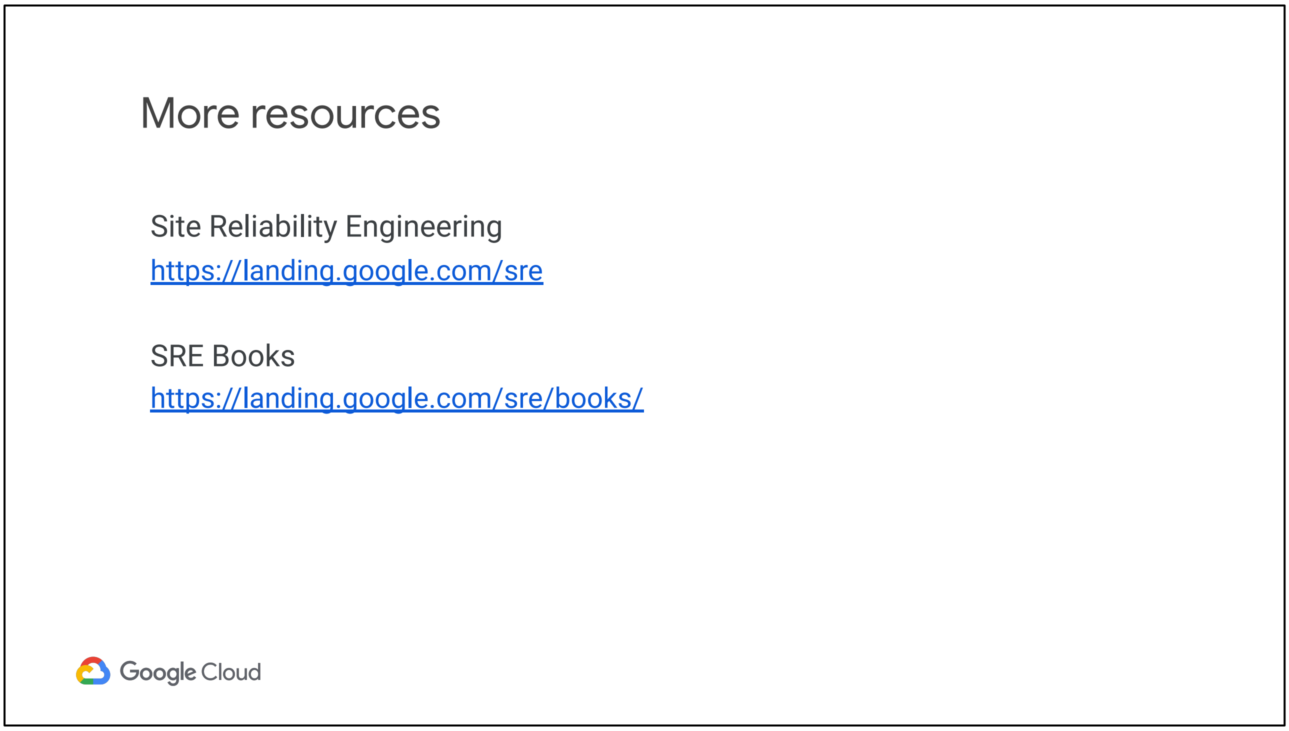
Review

Defining Services

Google Cloud

In this module we learned about qualitative and quantitative requirements. Qualitative requirements are things that the user cares about, like features. We can express qualitative requirements in the form of user stories. In order to understand our users better, we should write personas.

Quantitative requirements are things we can measure. We can express these as key performance indicators, or KPIs. KPIs in software are things like user signups, clicks per session, completed purchases, or customer retention. We can also express quantitative requirements as SLOs and SLIs. These are lower-level metrics, things like latency, availability, or response time.



More resources

Site Reliability Engineering
https://landing.google.com/sre

SRE Books
https://landing.google.com/sre/books/

Google Cloud

Take a look at these links for more information about Site Reliability Engineering and defining SLIs, SLOs, and SLAs. We've published two books on the topic: Site Reliability Engineering and The Site Reliability Workbook. These books can be read online for free, or you can purchase hard copy or Kindle versions.

Google Cloud