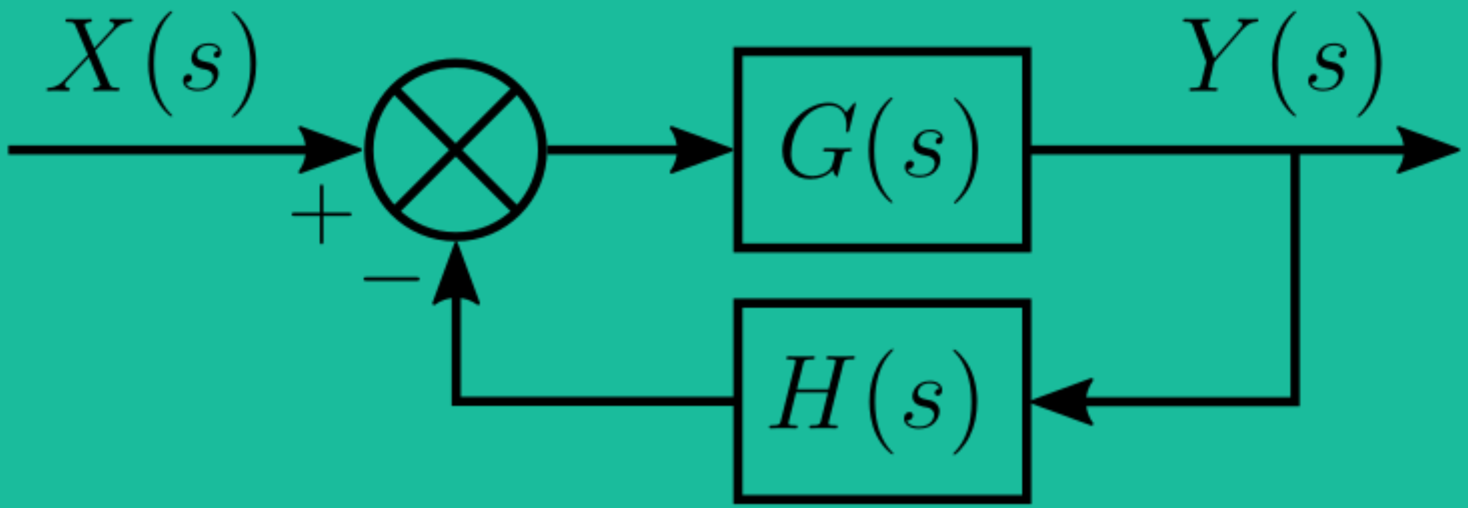
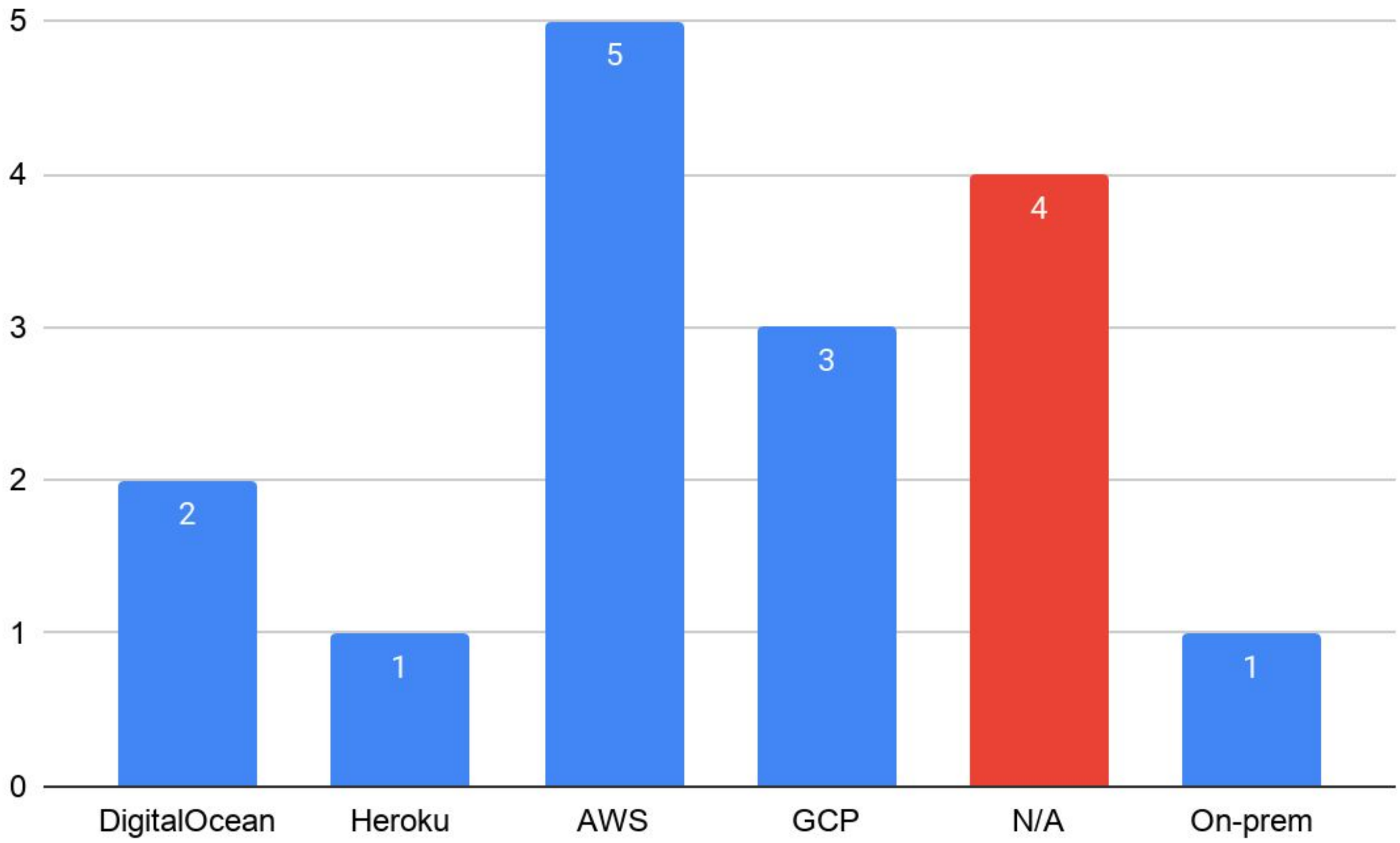


Feedback Sessions #1: System Designs

Bilkent University | CS443 | 2020, Spring | Dr. Orçun Dayıbaş

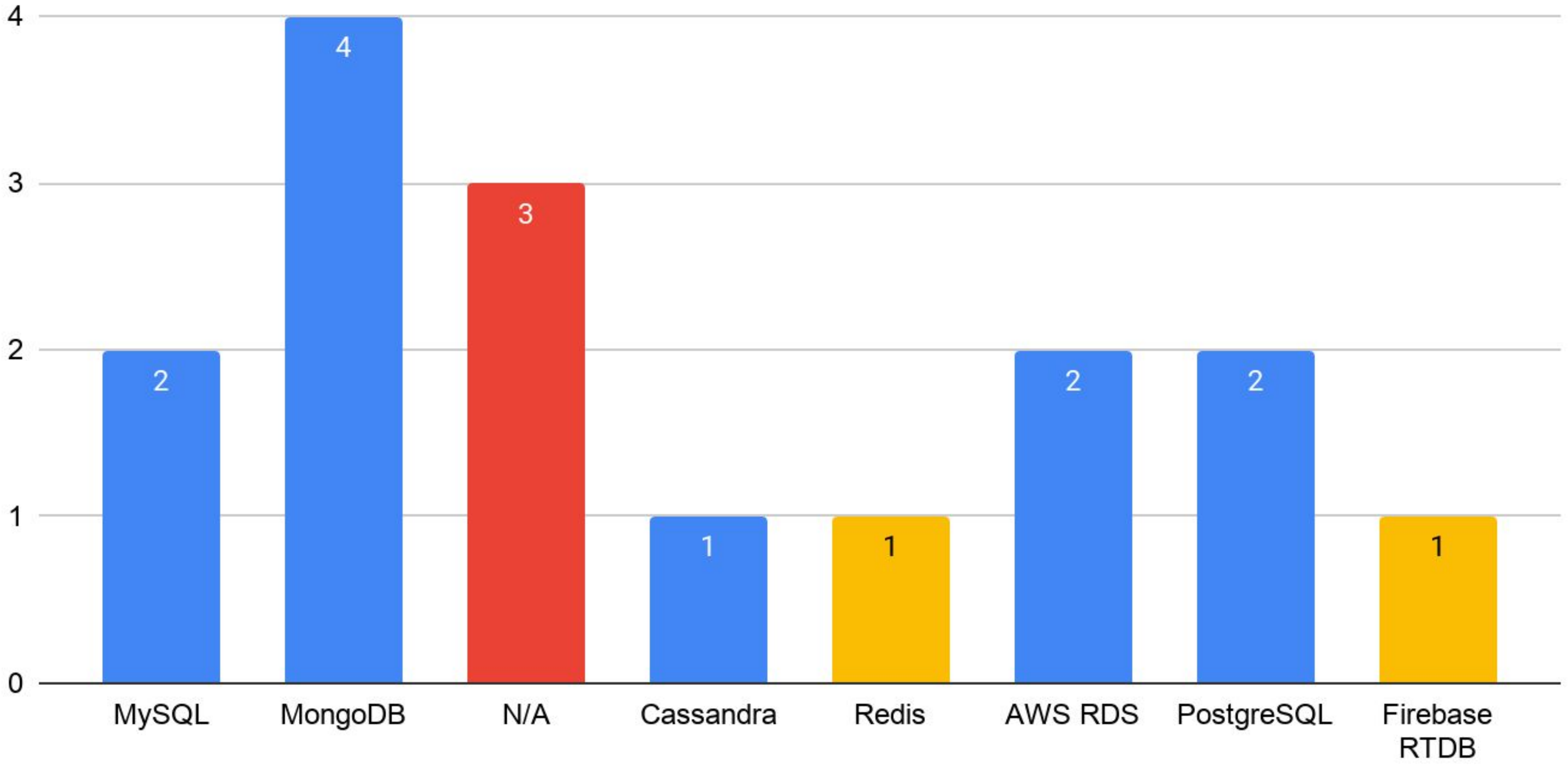
Tech Stack Decisions

- **Infra/Platform**



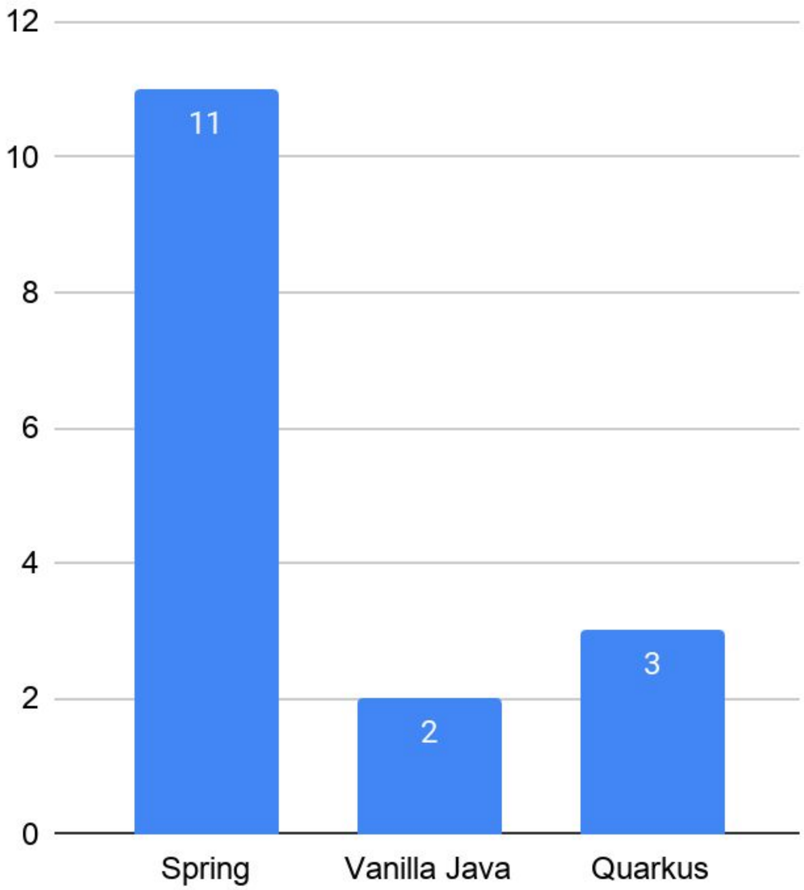
Tech Stack Decisions

- **Data (Persistency)**



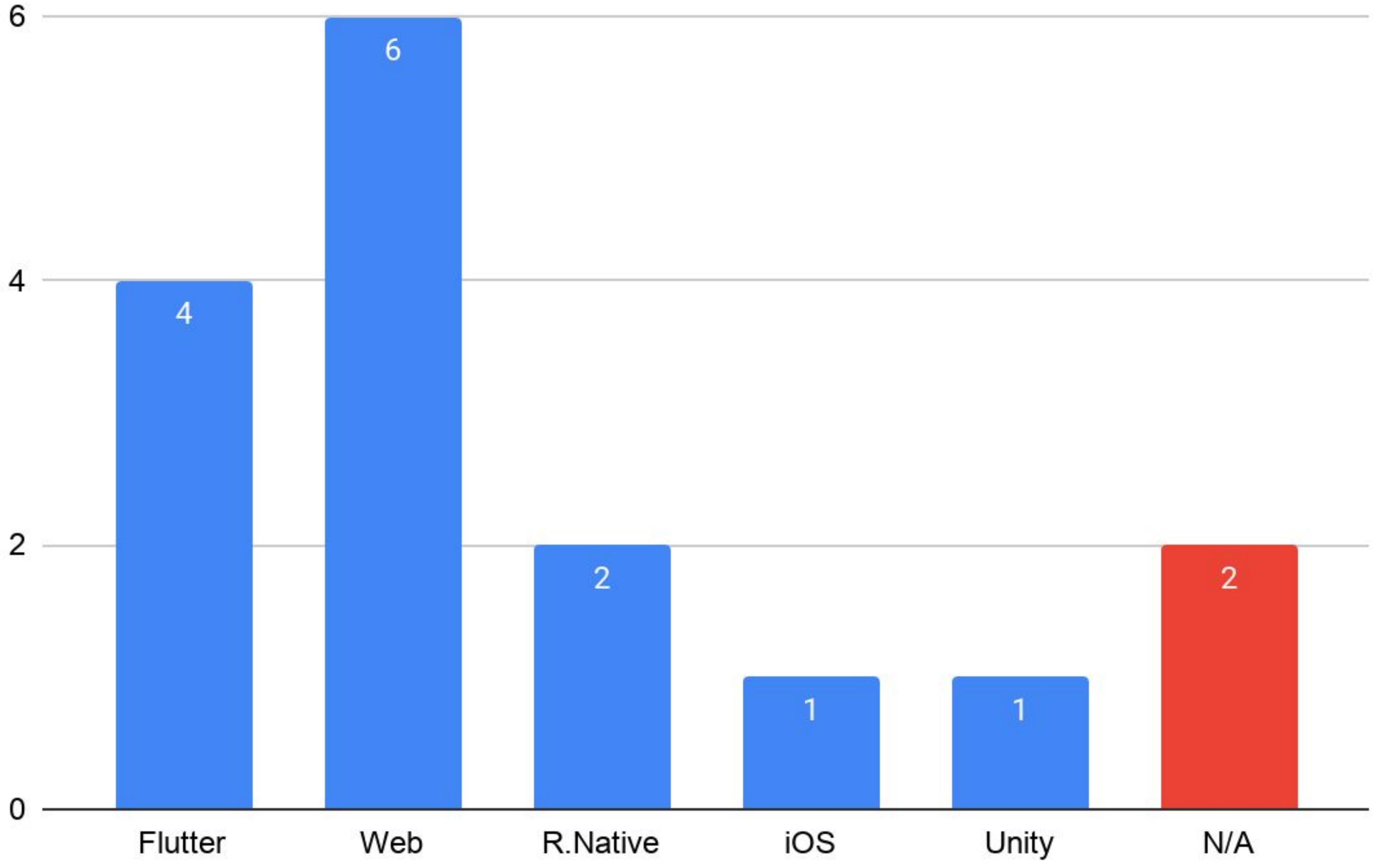
Tech Stack Decisions

- **Business Logic**



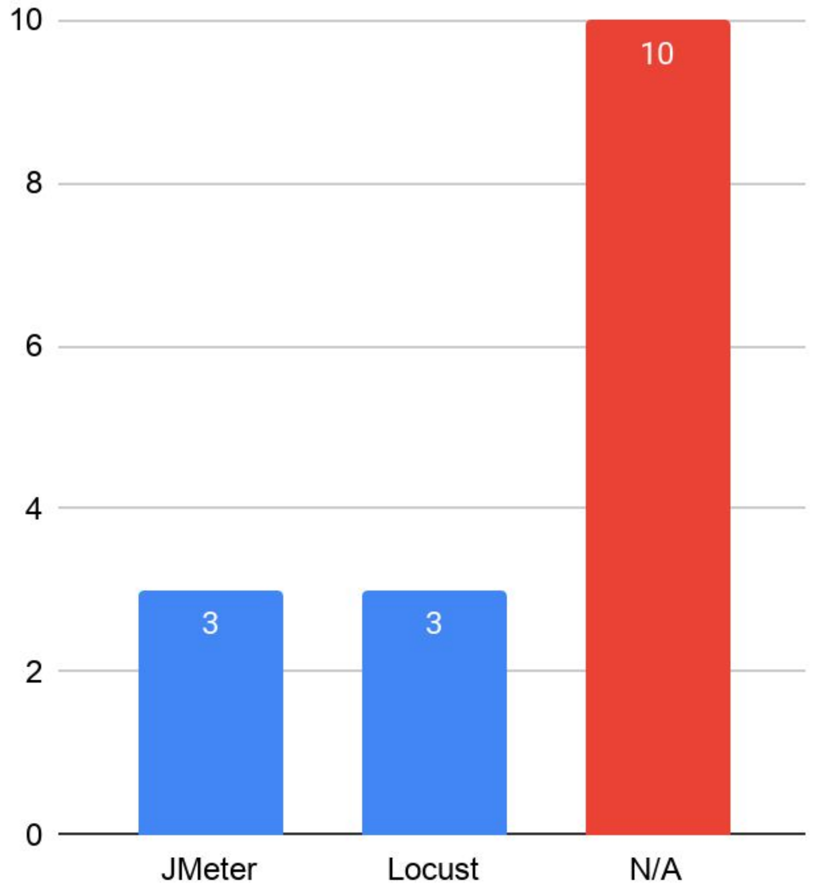
Tech Stack Decisions

- **Mobile**



Tech Stack Decisions

- **Test**



High Level System Design

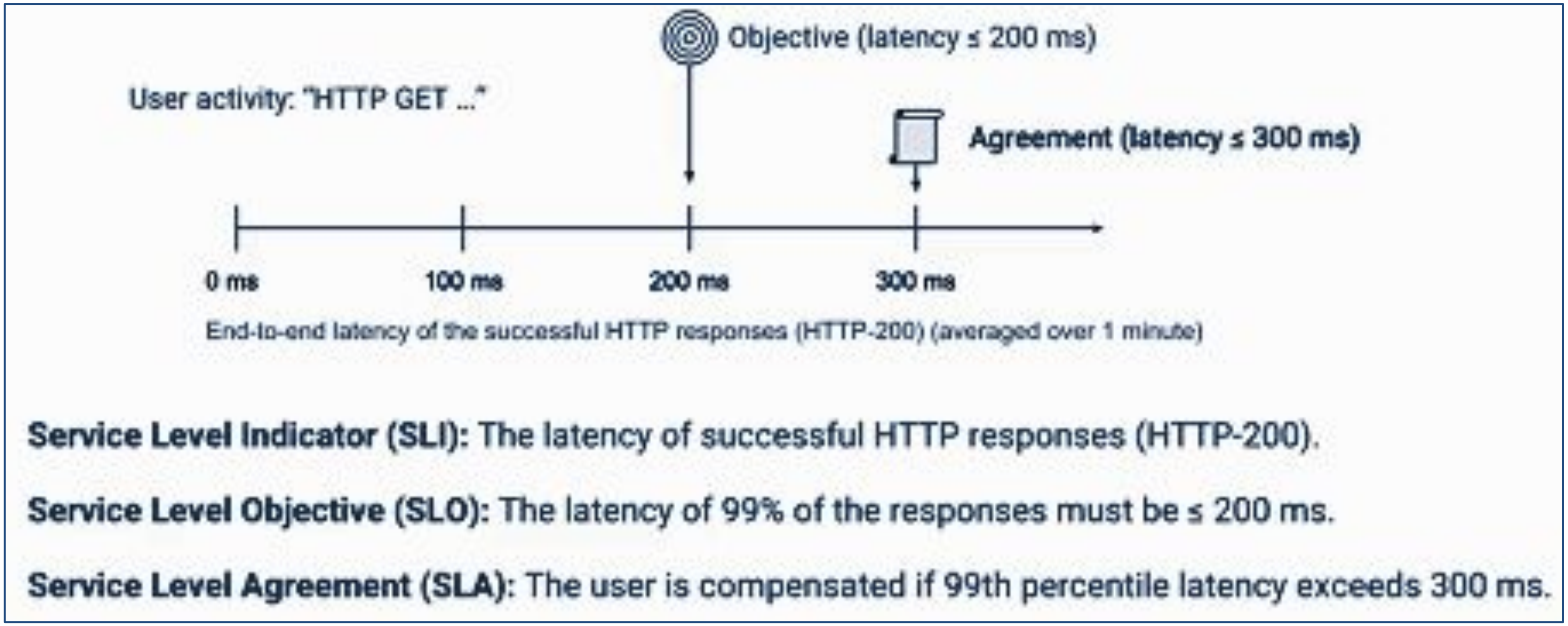
- **Scalable Design**
    - Not scalable
        - Ex: A solution with monolithic architecture & fixed resources
    - Vertically scalable
        - Ex: Using “only” VM solutions of cloud vendors
    - Horizontally scalable
        - There is no out-of-the-box solution; you have to handle it
            - Using microservices approach “can” help
            - Using kubernetes “can” help
            - Using <put_any_hype_here> “only can” help...

Mobile & API Design

- **We have discussed API design details in the class**
  - Refer “Developer Experience (DX)” slides to design “good one”
  - Easiest way to increase quality: adopting a standard/quasi-standard (do not reinvent the wheel)
    - The links are provided in the course website (e.g. Microsoft, Zalando) but it is not an exhaustive
- **Mobile**
  - Project definition states that UX is important
    - web vs. native-app

SLA Definitions

- **What is your objective?**
  - Explicit obj.: “satisfy your users’ need”
    - Implicit obj.: generating money (it has to be profitable & cost-effective)
    - Implicit obj.: user vs. client (it has to be sellable & sustainable)
  - SLA includes “precise definition” of the indicator & objective
    - Your “internal goal” may be different than the agreement → error budget
      - but it has to be defined in your design documents anyway


Objective (latency ≤ 200 ms)
User activity: "HTTP GET ..."
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)

Service Level Indicator (SLI): The latency of successful HTTP responses (HTTP-200).

Service Level Objective (SLO): The latency of 99% of the responses must be ≤ 200 ms.

Service Level Agreement (SLA): The user is compensated if 99th percentile latency exceeds 300 ms.

Testing Strategy

- **Why do you need a "strategy"?**
  - "Strategy" is a high level plan to achieve one or more goals under conditions of uncertainty.
  - SLA definitions + Requirements (func.&non-func.) → Indicators & Goals
  - Monitoring → Status/behaviour information
  - Use these inputs to formulate your testing strategy
    - Our plan is to do XXX to justify our claim on availability/scalability/etc.
  - Ex: Designing a system which has 5-nines availability for 100 MAUs for a year
    - Definition of "active user": uncertain
    - Common user behaviour: uncertain
- **It is also valid argument in general**
  - This is why you need assumptions/estimations in this project (actually, "in every meaningful cloud project")

Comments

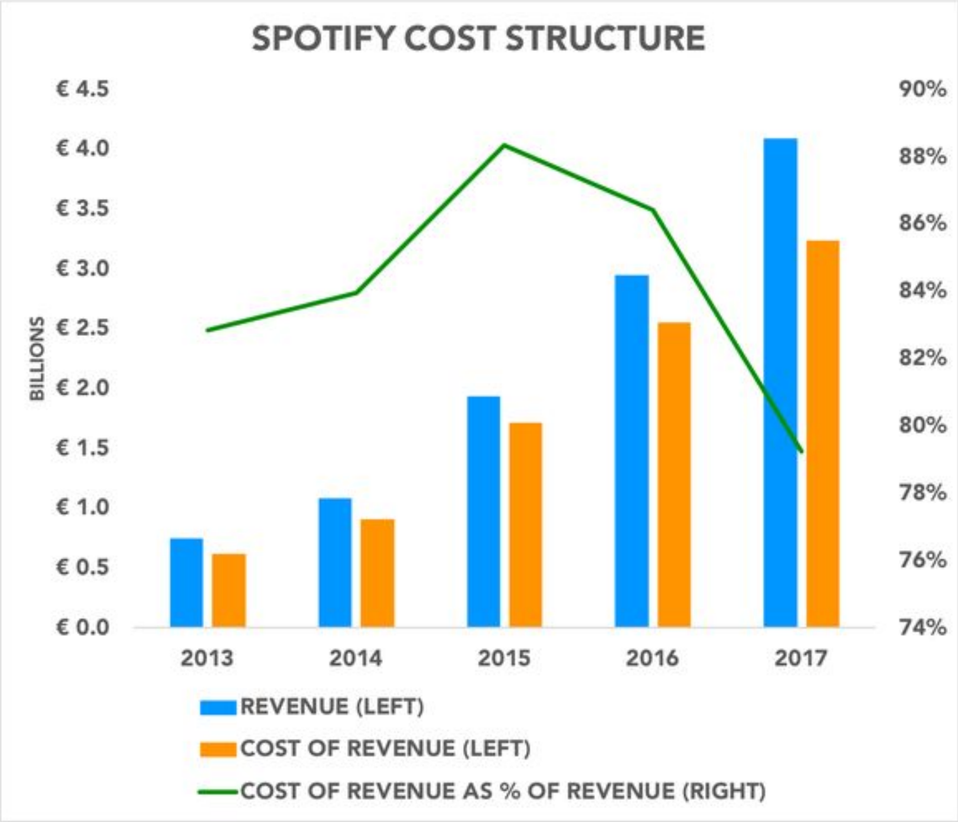
- **Final Report/Presentation**
    - Be sure to include;
        - **Mobile demo**
        - **Contributions of each team member**
            - Be precise & concise (e.g. a table w/ cols: name, task(s), % of all workload)
        - **Access details (at least for web part)**
            - URL, sign in details, etc.
        - **Proofs for SLAs (defined in your design report)**
            - Be precise & concise (SLA → Indicator/metric → Testing results)
            - No need to able to hold your promise but you need to be able to calculate (testing strategy) current status of your system
            - Discuss explicitly (include “aha moments”), if you have “non-realistic goals” in the design report; you have a chance to catch and fix it in your final presentation (do not try to sweep it under the rug; I assure you, it is not that hard to identify that kind of goals...)

Comments

- **Final Report/Presentation (cont.)**
    - Be sure to include;
        - **Limits of the system (computation → usage/user)**
            - It has “certain limits” whether you are aware of it or not (Free/On-prem/etc.)
                - Show/prove “you are aware of it”
            - Be precise & concise
                - Status
                    - “The system can host …. concurrent link accesses” (include proofs)
                    - “The system can host …. MAU for …. months” (include proofs)
                - Projection for scale
                    - We need …. $ to host 1K/1M/10M MAU” (include proofs and where to spend the resource, MAU: Monthly Active Users)
        - **Scaling in&out/up&down (methods)**
            - How your system expand/shrink (if it’s capable of)
            - Demonstrate (both in report & during the presentation)

Comments

- **Ask yourself**
  - Why “all” startups need investment?
  - What/when would you want from a VC if that project was your startup idea?



https://finance.yahoo.com/news/spotify-apos-truly-disruptive-potential-232400400.html

Comments

- **Availability is not provided by your CSP**
- **Scalability is not provided by buzzwords**
  - Microservices, Docker, Kubernetes, etc.
- **Uncertainty requires a strategy**
  - Design/plan for uncertainty
- **Tools/Frameworks(*) do not generate “a good” solution by themselves**

I NEED TO KNOW WHY MOVING OUR APP TO THE CLOUD DIDN'T AUTOMATICALLY SOLVE ALL OUR PROBLEMS.
@ScottAdamsSays
Dilbert.com
YOU WOULDN'T LET ME RE-ARCHITECT THE APP TO BE CLOUD-NATIVE.
JUST PUT IT IN CONTAINERS.
11-08-17 © 2017 Scott Adams, Inc./Dist. by Andrews McMeel
YOU CAN'T SOLVE A PROBLEM JUST BY SAYING TECHY THINGS.
KUBERNETES.

(*) Your grades will not be affected from the tech stack decisions as long as there is no effect on project goals

Q/A