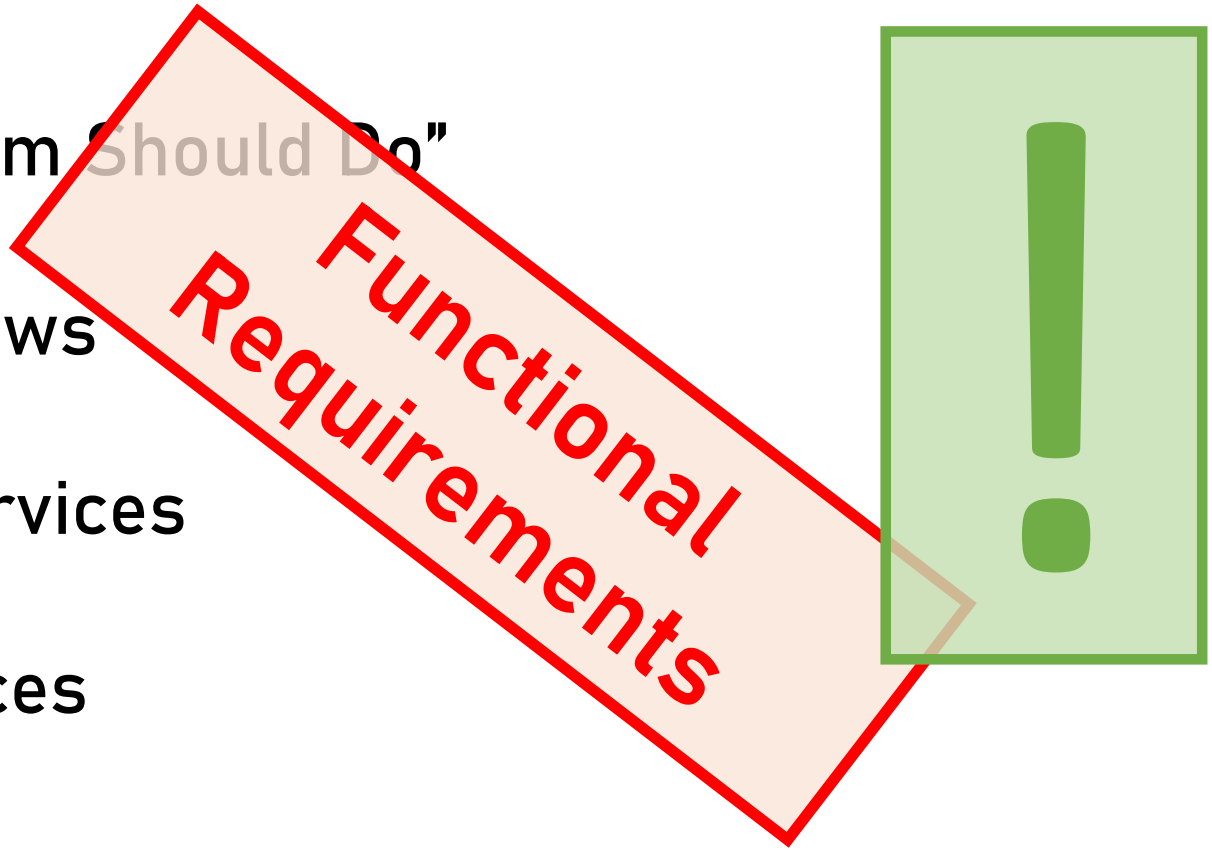
Understanding System Requirements

Memi Lavi
www.memilavi.com

Two Types of Requirements

- "What the System Should Do"
  - Business Flows
  - Business Services
  - User Interfaces



The Other Kind…

- “What Should the System Deal With”
  - Performance
  - Load
  - Data Volume
  - Concurrent Users
  - SLA

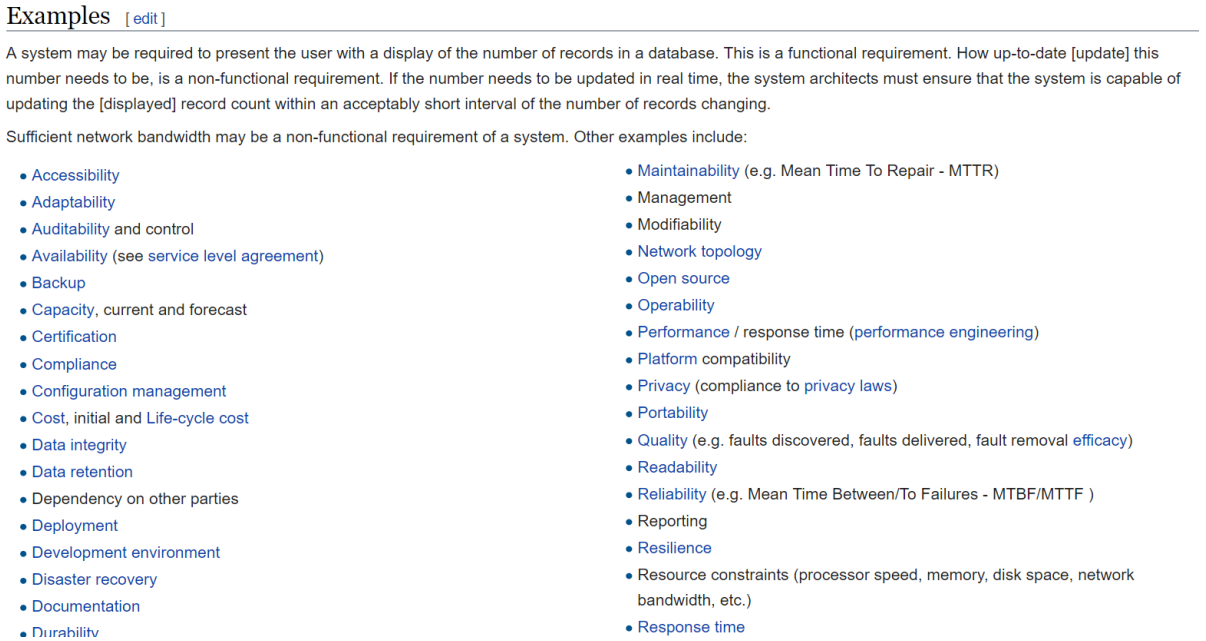
Non-Functional Requirements

Remember!

- Non-Functional Requirements are Important
- BUT - Do Not Ignore Functional Requirements!
  - Part of the Business

Non-Functional Requirements

- "What the System Should Deal With"
  - Performance
  - Load
  - Data Volume
  - Concurrent Users
  - SLA



Examples [edit]

A system may be required to present the user with a display of the number of records in a database. This is a functional requirement. How up-to-date [update] this number needs to be, is a non-functional requirement. If the number needs to be updated in real time, the system architects must ensure that the system is capable of updating the [displayed] record count within an acceptably short interval of the number of records changing.

Sufficient network bandwidth may be a non-functional requirement of a system. Other examples include:

- Accessibility
- Adaptability
- Auditability and control
- Availability (see service level agreement)
- Backup
- Capacity, current and forecast
- Certification
- Compliance
- Configuration management
- Cost, initial and Life-cycle cost
- Data integrity
- Data retention
- Dependency on other parties
- Deployment
- Development environment
- Disaster recovery
- Documentation
- Durability
- Maintainability (e.g. Mean Time To Repair - MTTR)
- Management
- Modifiability
- Network topology
- Open source
- Operability
- Performance / response time (performance engineering)
- Platform compatibility
- Privacy (compliance to privacy laws)
- Portability
- Quality (e.g. faults discovered, faults delivered, fault removal efficacy)
- Readability
- Reliability (e.g. Mean Time Between/To Failures - MTBF/MTTF)
- Reporting
- Resilience
- Resource constraints (processor speed, memory, disk space, network bandwidth, etc.)
- Response time

https://en.wikipedia.org/wiki/Non-functional_requirement

Performance

What is the required performance for this system?

Fast

Performance

- Always talk in numbers
- Latency & Throughput

Performance

- Always talk in numbers

Icon made by freepik.com

Latency

> How much time does it take to perform a single task?

Examples:

- How much time will it take for the API to save the user data in the database?
- How much time will it take to read a single file from the file system?

Throughput

> How many tasks can be performed in a given time unit?

Examples:

- How many users can be saved in the database in a minute?
- How many files can be read in a second?

Let's Crunch Numbers...

Saving User Data

Latency	1 Second

Let's Crunch Numbers…

Saving User Data

Latency	1 Second
Throughput	?

Let's Crunch Numbers…

Saving User Data

Latency	1 Second
Throughput	Well designed app: >1000

Let's Crunch Numbers…

Saving User Data

Latency	1 Second
Throughput	Well designed app: >1000
	Badly designed app: < 60

Load

- Quantity of Work Without Crashing
- Example: In WebAPI – How Many Concurrent Requests

  Without Crashing

Load vs Throughput

Throughput	
Load	

Load vs Throughput

Throughput	100 requests / sec
Load	

Load vs Throughput

Throughput	100 requests / sec
Load	500 requests without crashing

Always plan for extreme cases!

Data Volume

- How Much Data the System Will Accumulate Over Time
- Helps With:
  - Deciding on Database Type
  - Designing Queries
  - Storage Planning

Data Volume

- Two Aspects:
  - Data Required on “Day One”
  - Data Growth

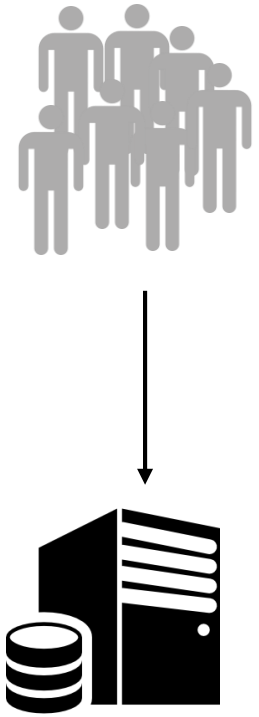
Data Volume

- Example:

Day One	500 MB
Annual Growth	2 TB

Concurrent Users

- How Many Users Will Be Using the System Simultaneously



Concurrent Users vs Load

Concurrent Users	Including “Dead Times”
Load	

Concurrent Users vs Load

Concurrent Users	Including “Dead Times”
Load	Actual Requests

Rule of Thumb:

Concurrent = Load X 10

SLA (Service Level Agreement)

- Required Uptime for the System

SLA for Azure Cosmos DB

Last updated: November 2017

Azure Cosmos DB is Microsoft's globally distributed multi-model database service. It offers turnkey global distribution across any number of Azure regions by transparently scaling and replicating your data wherever your users are. The service offers comprehensive 99.99% SLAs which covers the guarantees for throughput, consistency, availability and latency for the Cosmos DB Database Accounts scoped to a single Azure region configured with any of the five Consistency Levels or Database Accounts spanning multiple Azure regions, configured with any of the four relaxed Consistency Levels. Furthermore, independent of the choice of a Consistency Level, Cosmos DB offers 99.999% SLA for read availability for Database Accounts spanning two or more Azure regions.

Source: https://azure.microsoft.com/en-us/support/legal/sla/cosmos-db/v1_1/

SLA (Service Level Agreement)

- 99.99% Equals To:

24 X 365 = 8760 hrs / year

8760 X 99.99% = 8759.12

8760 – 8759.12 = 0.88 hrs downtime / year

SLA (Service Level Agreement)

- Manage Client's Expectations
- 99.999% Uptime Is Not a Realistic Goal…

Non-Functional Requirements

Never

start working before setting them!

Who Defines Non-Functional Requirements?

Defining Non-Functional Requirements

What is the SLA for the system?

Always!

Defining Non-Functional Requirements

What is the required response time for the API?

10 ms!

Defining Non-Functional Requirements

- Architect's Roles:
  - Framing the requirements' boundaries
  - Discuss numbers

Non-Functional Requirements Conclusion

- Define what the system will have to deal with
  - Performance, SLA, Load, etc.
- Client won't be able to define them
- Never begin working on a system without Non-Functional Requirements in place