Why is software architecture important

Without a proper architecture the system might be

- Difficult to build
- Difficult to test
- Difficult to maintain
- Difficult to change
- Difficult to understand
- Difficult to reuse
- Difficult to integrate with other systems

Difference software architecture and software design

- Software architecture oversees the whole while software design mostly focusses on one design problem
- Software architecture needs to make sure that all requirements are met
- Software architecture is often more at a higher level than software design
- Architecture is design, but design does not need to be architecture

What makes software architecture so difficult?

- Many possibilities
- Many frameworks, libraries, styles, patterns, databases, etc
- Every possibility has advantages and disadvantages
- Everything is a tradeoff
- Everything changes all the time
- Many decisions to take

Explain clearly the main differences of software architecture in a traditional waterfall project and software architecture in an agile project.

Traditional: 1 person has the role architect, but this person does not code at all. The architect makes all architecture decisions.

The architect writes an architecture document, and then often leaves the project

Agile: One developer takes on the role architect and oversees the whole, but this architect also writes code. We make architecture decisions together with the team.

We do not write a large architecture document

Suppose you need to define the architecture for a large expensive system, and it is important that this system is future proof because this system will be used for at least 20 years. Explain how you can design a future proof system.

Make sure your architecture implements most of the important principles:

- Keep it simple
- Keep it flexible
- Loose coupling
- High cohesion, low coupling
- Separation of concern
- Information hiding
- Principle of modularity
- Open-closed principle

For each of the following qualities, give at least 1 technique that you know to increase this quality:

1. Performance
    a) Use a cache
    b) Minimize remote calls
2. Availability
    a) Use a cluster of multiple instances of a certain application
3. Resilience (against failure)
    a) Implement backup functionality in case something fails
4. Reusability
    a) Divide the system into smaller reusable parts
5. Maintainability
    a) Test automation
    b) Clean code