**Software architecture document**

Table of contents:

[1. Introduction 3](#_Toc152800772)

[1.1 Purpose 3](#_Toc152800773)

[1.2 Scope 3](#_Toc152800774)

[1.3 Definitions, Acronyms, and Abbreviations 3](#_Toc152800775)

[1.3.1 Acronyms: 3](#_Toc152800776)

[1.4 References 3](#_Toc152800777)

[1.5 Document Overview 3](#_Toc152800778)

[2. Architectural Representation 3](#_Toc152800779)

[3. Architectural Goals and Constraints 4](#_Toc152800780)

[3.1 Architectural Goals 4](#_Toc152800781)

[3.2 Architectural Constraints 4](#_Toc152800782)

[4. Use-Case View 4](#_Toc152800783)

[4.1 Architecturally-Significant Use Cases 5](#_Toc152800784)

[4.2 Use-Case View Diagram 7](#_Toc152800785)

[5. Logical View 7](#_Toc152800786)

[5.1 Architecture Overview – Package and Subsystem Layering 7](#_Toc152800787)

[6. Process View 9](#_Toc152800788)

[6.1 Processes 9](#_Toc152800789)

[6.2 Process to Design Elements 10](#_Toc152800790)

[6.3 Process Model to Design Model Dependencies 11](#_Toc152800791)

[6.4 Processes to the Implementation 11](#_Toc152800792)

[7. Deployment View 13](#_Toc152800793)

[8. Size and Performance 14](#_Toc152800794)

[8.1 Data Size Details 15](#_Toc152800795)

[8.2 System Scalability Details 16](#_Toc152800796)

[8.3 Response Times Details 16](#_Toc152800797)

[8.4 Data Security and Encryption Details 17](#_Toc152800798)

[9. Quality 17](#_Toc152800799)

[9.1 Reliability Details 18](#_Toc152800800)

[9.2 Maintainability Details 19](#_Toc152800801)

[9.3 Usability Details 19](#_Toc152800802)

[9.4 Scalability Details 20](#_Toc152800803)

[9.5 Interoperability Details 20](#_Toc152800804)

[9.6 Performance Details 20](#_Toc152800805)

[10. Conclusion 21](#_Toc152800806)

# 1. Introduction

## 1.1 Purpose

The purpose of this Software Architecture Document (SAD) is to provide a comprehensive overview of the SmartSticker project's architecture. It aims to guide the development team, project stakeholders, and individuals involved in the design and implementation of the SmartSticker system.

## 1.2 Scope

The scope of this document encompasses the architectural representation, design decisions, and requirements of the SmartSticker system. It addresses both functional and non-functional aspects to ensure a clear understanding of the system's architecture.

## 1.3 Definitions, Acronyms, and Abbreviations

### 1.3.1 Acronyms:

SAD: Software Architecture Document

NFC: Near Field Communication

OS: Operating System

HTTPS: Hypertext Transfer Protocol Secure

MVC: Model-View-Controller

## 1.4 References

Software Requirements Specification (SRS) for SmartSticker, Version 0.1, prepared by Danylo Zemlianyi on 16.10.2023.

## 1.5 Document Overview

This document is structured to provide a clear and cohesive presentation of the SmartSticker system's architecture. It includes sections on architectural representation, design decisions, and detailed requirements.

# 2. Architectural Representation

The architectural representation of the SmartSticker system is captured through class diagrams, package diagrams, and use-case diagrams. These visual representations illustrate the key components, their interactions, and the overall structure of the system.

# 3. Architectural Goals and Constraints

The architectural goals and constraints of the SmartSticker system are established to guide the design and ensure the system meets the desired objectives.

## 3.1 Architectural Goals

Modularity: The system will employ a modular architecture to facilitate easier maintenance and future enhancements.

Scalability: The architecture will support horizontal scalability to accommodate a growing user base and increased demand.

Security: Strong security measures will be implemented, including encryption protocols, secure storage of sensitive data, and biometric authentication.

Cross-Platform Compatibility: The mobile application will be natively developed for both Android and iOS platforms, ensuring optimal performance and adherence to platform-specific guidelines.

Integration: The system will seamlessly integrate with other information systems within the enterprise, following industry standards for data exchange.

## 3.2 Architectural Constraints

Mobile Application Platforms: The system will be constrained to Android and iOS platforms for the mobile application.

NFC Technology: The system relies on Near Field Communication (NFC) technology for communication between smartphones and NFC stickers affixed to physical objects.

Communication Protocols: The use of HTTPS for secure data transmission is a constraint to ensure encrypted communication between the application and server.

# 4. Use-Case View

Use-Case describes a set of scenarios that represent main system functionality that is important for architecture representation.

## 4.1 Architecturally-Significant Use Cases

The architecturally-significant use cases of the SmartSticker system are derived from the fully-dressed use cases presented earlier.

*4.1.1 User Registration*

Primary Actor: User

Description: Allow user to create their account, that will be later be used for user authentication, and getting access to objects.

*4.1.2 User Authentication*

Primary Actor: User

Description: The use case allows system to identify users and grant them access to system functionality. It generally uses username and password, but user can also choose another method of authentication – biometric.

*4.1.3 Biometric Authentication*

Primary Actor: User

Description: The use case is an extension of user authentication use-case. User can choose to use biometric data – fingerprint or face recognition, instead of username and password. But device must support this functionality and have relative identifier data stored.

*4.1.4 Object choosing (NFC scanning)*

Primary Actor: User

Description: The use case allows users to access object through NFC scanning. By using NFC technology of the smartphone users scan NFC sticker on the object they want to identify or change status of. NFC sticker will provide object identifier and help to retrieve object data from the database.

*4.1.5 Check allocated accesses*

Primary Actor: User

Description: The use case will allow users to check their allocated accesses through designated interface. Users can interact only with those objects they have access to, and this will help users to manage object easier. Allocated accesses data for a specific user will be retrieved from the database.

*4.1.6 Check object status*

Primary Actor: User

Description: The use case will allow user to check the status of the object they have scanned through NFC. It shows real-time status and if there are any changes in the database, it will be shown on users screen.

*4.1.7 Change object status*

Primary Actor: User

Description: The use case will allow users to change the status of the object they choose via NFC scanning. This changes will be displayed on server in realtime.

*4.1.8 Manage profile*

Primary Actor: User

Description: The use case allows users to manage their profile, change password and username, profile description and picture.

*4.1.9 Access allocation*

Primary Actor: System administrator

Description: This use case allows system administrators to allocate and deallocate accesses through server. Changes in statuses will be displayed in the database.

## 4.2 Use-Case View Diagram

Зображення, що містить текст, знімок екрана, схема, дизайн

Автоматично згенерований опис

# 5. Logical View

## 5.1 Architecture Overview – Package and Subsystem Layering

The logical view of the SmartSticker system is organized into packages and subsystems, representing the high-level structure of the software architecture.

*Smartphone Application Package:*

Description: This package encompasses all components related to the mobile application running on Android and iOS platforms.

Subsystems:

User Interface: Responsible for presenting information and receiving user input.

NFC Interaction: Manages the communication with NFC stickers.

User Account Management: Handles user registration, login, and profile management.

Object Control: Facilitates user interactions with allocated physical objects.

*Central Server Package:*

Description: The central server acts as the core backend component responsible for processing user requests, managing user accounts, and coordinating access to physical objects.

Subsystems:

User Authentication: Verifies user credentials and grants access.

Object Database: Stores information about physical objects, their identifiers, and permissions.

Access Control: Enforces access rights and allocates/deallocates objects to users.

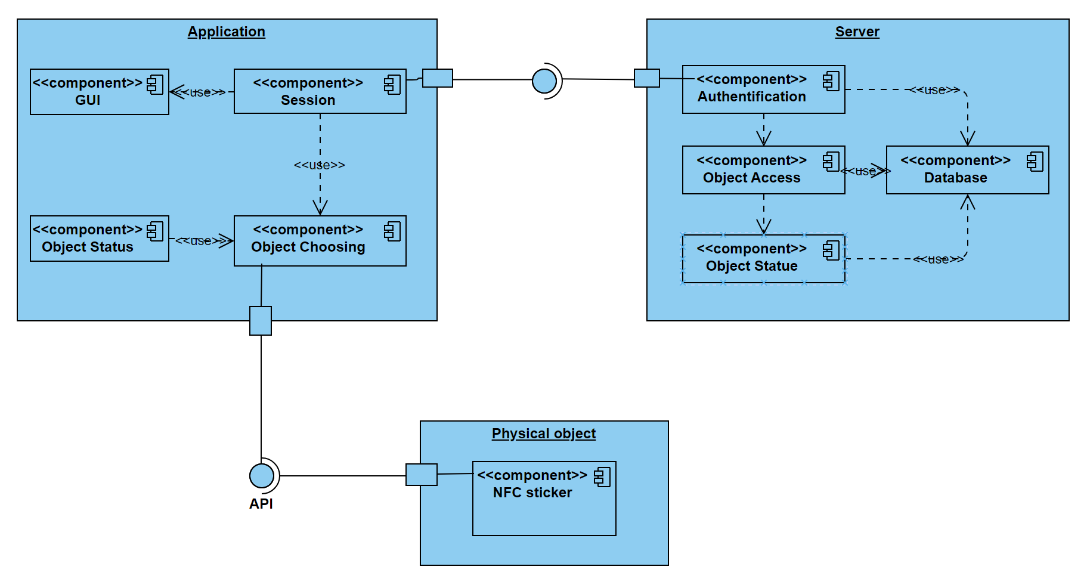
*External Interfaces Package:*

Description: This package handles interactions with external components, including smartphones with NFC capabilities and external databases.

Subsystems:

NFC Communication: Manages communication between smartphones and NFC stickers.

Database Interface: Facilitates data exchange with external databases.



# 6. Process View

## 6.1 Processes

The process view of the SmartSticker system outlines the key processes involved in the system's operation.

*User Registration Process:*

Description: This process involves the user registration workflow, where a user provides personal information to the administrator, and the administrator creates a user account and issues an NFC sticker.

Steps:

User provides personal information.

Administrator creates a user account.

Administrator issues an NFC sticker to the user.

*Room Access Process:*

Description: The room access process details how users use their NFC stickers to gain access to specific rooms within the enterprise.

Steps:

User presents NFC sticker to the access control system.

Access control system verifies the sticker's validity.

Access control system grants access to the user.

Зображення, що містить текст, схема, Паралель, ряд

Автоматично згенерований опис

## 6.2 Process to Design Elements

The processes identified in the process view are mapped to design elements, defining how these processes are implemented in the system's architecture.

*User Registration Process:*

Design Elements:

Mobile Application (User Interface): Facilitates user input for personal information.

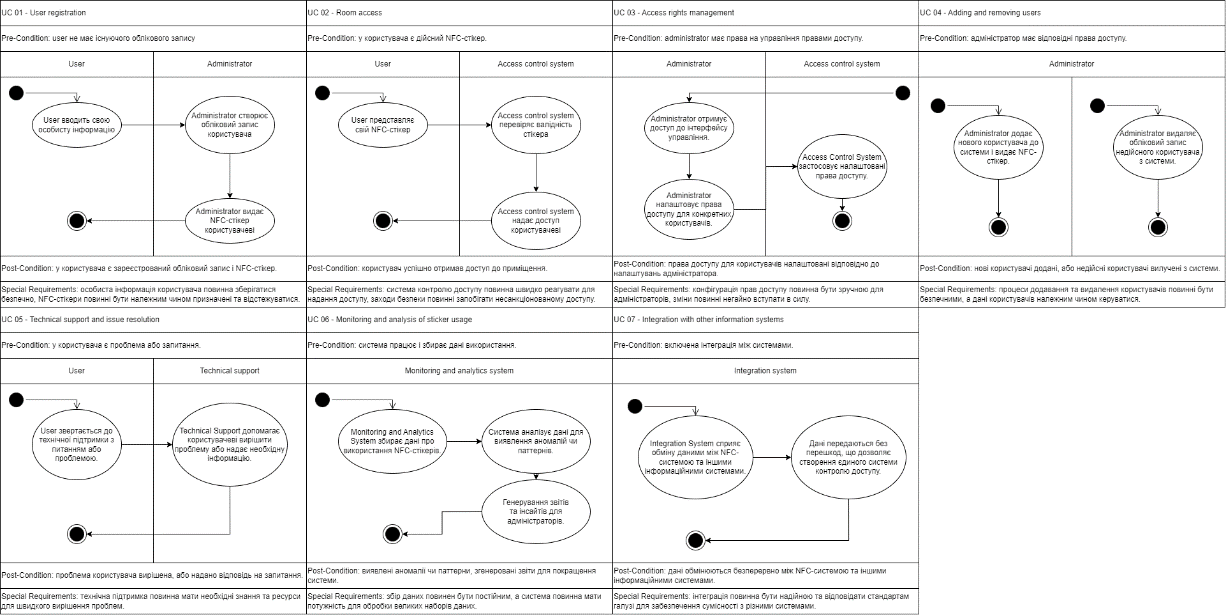
Central Server (User Account Management): Manages the creation of user accounts and issues NFC stickers.

*Room Access Process:*

Design Elements:

Mobile Application (NFC Interaction): Manages the communication with NFC stickers.

Central Server (Access Control): Enforces access rights and verifies the validity of NFC stickers.



## 6.3 Process Model to Design Model Dependencies

The identified processes in the process view are crucial components of the overall system architecture. Each process has dependencies on specific design elements, illustrating how the high-level processes are implemented at the design level.

*User Registration Process:*

Dependencies:

Mobile Application (User Interface): Dependency for user input and information gathering.

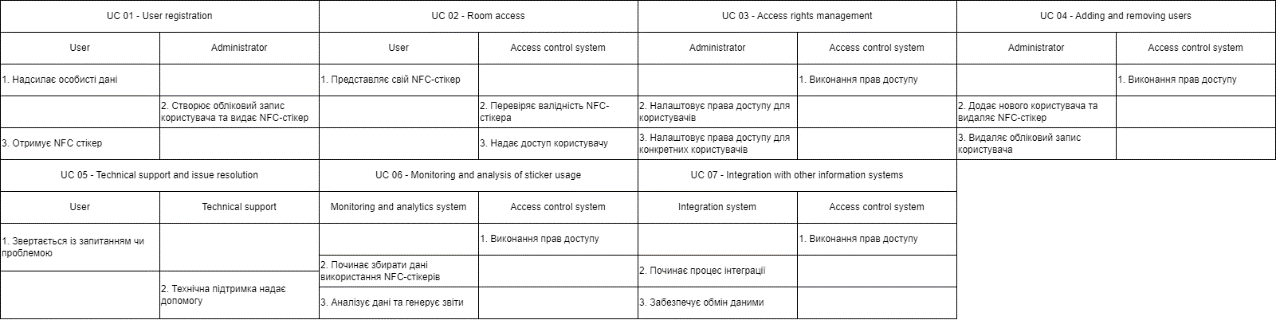
Central Server (User Account Management): Dependency for creating user accounts and issuing NFC stickers.

*Room Access Process:*

Dependencies:

Mobile Application (NFC Interaction): Dependency for managing communication with NFC stickers.

Central Server (Access Control): Dependency for enforcing access rights and verifying NFC sticker validity.



## 6.4 Processes to the Implementation

The processes identified in the process view are translated into actual implementation steps. These steps provide a detailed guide on how each process is executed within the system.

*User Registration Process:*

Implementation Steps:

User provides personal information through the mobile application's user interface.

Central Server's User Account Management processes the information and creates a new user account.

The Central Server issues an NFC sticker to the user through the Mobile Application.

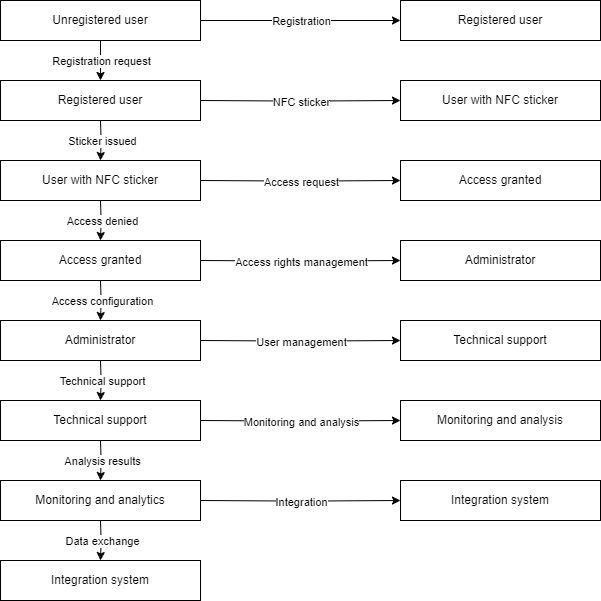
*Room Access Process:*

Implementation Steps:

User presents the NFC sticker to the Mobile Application's NFC Interaction module.

NFC Interaction communicates with the Central Server's Access Control module to verify the sticker's validity.

Access Control grants access to the user if the sticker is valid.



# 7. Deployment View

The following diagram shows the view of the various physical nodes that are needed to run the system.

Зображення, що містить схема, текст, ескіз, План

Автоматично згенерований опис

*7.1 Device*

Device is a user’s smartphone with NFC module and WiFi connection on which application is installed.

*7.2 Application*

Application is a software package installed on the user’s device, a client that contains a code realization of SmartSticker system.

*7.3 NFC*

NFC is a technology that device should have for application to work correctly. Application will use NFC module to communicate with NFC sticker through API.

*7.4 NFC sticker*

NFC sticker is an NFC tag with a chip, supplied by a company that specializes in NFC tag and supplies production. NFC sticker contains identifier, that smartphone with NFC can read.

*7.5 Object*

Object is a physical object that needs identification and access through smartphone. NFC sticker should be attached to it to allow communication with user’s device through NFC.

*7.5 Server*

Server is a system that processes request sent from the client (application), gets and processes data from the database and sends replies back to the client.

*7.6 Database*

Database is an organised collection of data, that can be stored, accessed and managed by the system. It is stored on server and also can be accessed from it. Database is used to store *user permissions* – users data and allocated accesses and *objects statuses* – object identifiers, data and realtime status.

# 8. Size and Performance

The size and performance considerations of the SmartSticker system play a crucial role in ensuring its efficiency and responsiveness. This section addresses aspects related to data size, system scalability, and response times.

Data Size:

The system handles user account information, object data, and access permissions.

User account data includes personal information, authentication credentials, and assigned NFC stickers.

Object data comprises information about physical objects, including identifiers and allocated user information.

Access permissions specify user rights to access specific objects.

System Scalability:

The architecture is designed to scale horizontally to accommodate a growing user base and increased object allocation.

The Central Server's architecture allows for the addition of server instances to distribute the processing load and enhance system responsiveness.

Response Times:

The system aims for a maximum response time of 2 seconds for typical user interactions.

Response times for critical processes, such as room access, are optimized for quick verification and access granting.

Data Security and Encryption:

Sensitive data, including user credentials and permissions, is stored using strong encryption in the server's database.

Data transmitted between the application and server, as well as communication with external systems, utilizes secure protocols like HTTPS.

## 8.1 Data Size Details

User Account Data:

Personal Information: Name, contact details, and user profile.

Authentication Credentials: Username, password, and biometric data.

NFC Sticker Information: Unique identifier and assignment status.

Object Data:

Object Identifier: Unique identifier for each physical object.

Allocation Information: User information allocated to the object.

Access Permissions:

User Permissions: Specific permissions assigned to users for accessing objects.

## 8.2 System Scalability Details

Horizontal Scaling:

The Central Server's architecture supports the addition of server instances to distribute the processing load.

Load balancing mechanisms ensure even distribution of user requests among server instances.

User Base Growth:

The system is designed to handle a growing user base by adding more resources dynamically.

The Registration Server and Central Server are optimized for efficient user account management and access control.

Object Allocation Handling:

The system scales to handle increased object allocations by optimizing database queries and access control processes.

Database sharding or partitioning may be implemented to enhance performance.

## 8.3 Response Times Details

Typical User Interactions:

Common user interactions, such as scanning NFC stickers and accessing object status, aim for a response time of less than 2 seconds.

The system prioritizes quick and seamless user experiences.

Critical Processes - Object Access:

The room access process is optimized for rapid verification of NFC sticker validity and granting access within milliseconds.

Multi-threading and parallel processing may be employed to enhance response times.

## 8.4 Data Security and Encryption Details

Storage Encryption:

Sensitive data, including user credentials and permissions, is stored using strong encryption algorithms.

Encryption keys are securely managed, and access to stored data is restricted.

Data Transmission:

Data transmitted between the application and server, as well as communication with external systems, uses secure protocols such as HTTPS.

End-to-end encryption is implemented to protect user data during transmission.

Access Control Mechanisms:

Role-based access control (RBAC) is employed to ensure that only authorized users, including administrators, can access specific system functionalities.

Authentication processes are designed to prevent unauthorized access.

Biometric Data Security:

Biometric data, such as fingerprints or facial recognition, is securely stored and processed.

Encryption is applied to biometric templates to protect user privacy.

# 9. Quality

Ensuring the quality of the SmartSticker system involves addressing various aspects, including reliability, maintainability, and usability. This section outlines the quality considerations integrated into the software architecture.

Reliability:

The system aims for high reliability to ensure consistent and secure access control.

Redundancy mechanisms are implemented in critical components to mitigate the impact of potential failures.

Maintainability:

The architecture prioritizes maintainability to facilitate efficient system updates and modifications.

Modularity and clear subsystem delineation contribute to ease of maintenance.

Usability:

User interfaces, particularly the mobile application, adhere to platform design guidelines for Android and iOS, ensuring a familiar and intuitive user experience.

User feedback mechanisms and error messages are designed to be user-friendly and guide users through issue resolution.

Scalability:

The system's architecture is scalable, allowing for the addition of resources to accommodate a growing user base and increased object allocation.

Performance:

The system prioritizes responsiveness, with a maximum response time of 2 seconds for typical user interactions.

Continuous monitoring and optimization processes are in place to maintain optimal performance.

## 9.1 Reliability Details

Redundancy Measures:

Critical components, such as the Central Server, may have redundant instances to ensure continuous operation in case of a failure.

Automated failover mechanisms redirect traffic to operational instances.

Data Backup and Recovery:

Regular data backups are performed to prevent data loss in the event of system failures.

Robust recovery procedures are in place to restore system functionality quickly.

## 9.2 Maintainability Details

Modularity:

The system architecture is modular, with clearly defined components and subsystems.

Each module encapsulates specific functionality, facilitating independent updates and modifications.

Documentation:

Comprehensive documentation is provided for system components, APIs, and data structures.

Code comments and inline documentation support developers in understanding and maintaining the codebase.

Version Control:

The use of version control systems, such as Git, ensures traceability and facilitates collaborative development.

Tagging and branching strategies are employed to manage different software versions.

Continuous Integration/Continuous Deployment (CI/CD):

CI/CD pipelines are implemented to automate the testing and deployment processes.

Automated testing suites ensure that changes do not compromise system stability.

## 9.3 Usability Details

Platform Guidelines Adherence:

The mobile application adheres to platform design guidelines for Android and iOS to provide a familiar and intuitive user experience.

User Feedback Mechanisms:

The application incorporates user-friendly error messages and guidance to assist users in resolving issues effectively.

Accessibility Features:

The application considers accessibility standards to ensure usability for users with diverse needs.

## 9.4 Scalability Details

Dynamic Resource Allocation:

The architecture supports dynamic resource allocation to handle a growing user base.

Resources, including server instances, are added or removed dynamically based on demand.

Object Allocation Optimization:

The system scales to handle increased object allocations by optimizing database queries and access control processes.

Load balancing mechanisms ensure even distribution of user requests among server instances.

## 9.5 Interoperability Details

Integration Interfaces:

Well-defined integration interfaces are established to facilitate seamless data exchange with external systems.

## 9.6 Performance Details

Continuous Monitoring:

Continuous monitoring processes are in place to track system performance metrics.

Anomalies or performance degradation triggers proactive optimization efforts.

Optimization Strategies:

Performance optimization strategies include database indexing, query optimization, and caching mechanisms.

Regular performance assessments guide ongoing optimization initiatives.

# 10. Conclusion

The SmartSticker software architecture is a robust and well-designed solution that addresses the complex requirements of controlled access to physical objects using Near Field Communication (NFC) technology. This document has provided a comprehensive overview of the architecture, covering various aspects, including:

Introduction:

Purpose, scope, definitions, acronyms, and references.

Architectural Representation:

Visualization of the system's architecture through diagrams and models.

Architectural Goals and Constraints:

Defined goals and constraints influencing the architecture.

Use-Case View:

Identification and elaboration of architecturally-significant use cases.

Logical View:

Overview of the package and subsystem layering within the architecture.

Process View:

Description of key processes and their interactions, along with dependencies between process and design models.

Deployment View:

View of the various physical nodes that are needed to run the system.

Size and Performance:

Details on data security, encryption, reliability, maintainability, usability, scalability, interoperability, performance, security, and non-functional requirements.

Quality:

In-depth coverage of reliability, maintainability, usability, scalability, interoperability, performance, security, and non-functional requirements.