DELIVERABLE

D2.3 System requirements

Project Acronym:	LAZARUS	
Project title:	pLatform for Analysis of Resilient and secUre Software	
Grant Agreement No.	101070303	
Website:	https://lazarus-he.eu/	
Contact:	info@lazarus-he.eu	
Version:	1.0	
Date:	30/06/2023	
Responsible Partner:	MOT	
Contributing Partners:	MAG, BNR, ICO, ARC, UCM, UNIPD, DC, LIST, APWG	
Reviewers:	Konstantinos Patsakis (ARC) Nikos Lykousas (DC)	
Dissemination Level:	Public	X
	Confidential – only consortium members and European Commission Services	

This project has received financial support from the European Union Horizon 2020 Programme under grant agreement no. 101070303.

Revision History

Revision	Date	Author	Organization	Description
0.1	20/02/2023	Miltos Anastasiadis	MOT	TOC
0.2	09/03/2023	Andrei Costin, Vadim Bogulean, Lelia Ataliani, Xaris Listis, Thanos Karantjias, Spyridon Papastergiou, Tareq Chihabi, Nikos Drosos, Mirko Fabbri, Alessandro Neri, Mauro Scarpa, Nikolaos Karaiskakis, Filippo Paganelli, Nikos Stratigakis, Panagiotis Markovits, Miltiadis Anastasiadis	MAG, BNR, ICO, MOT, DC	First initial input about systems specs and relation to T2.1 and T2.2
0.3	07/04/2023	Ana Lucila Sandoral, Sandra Perez, Luis Alberto Martinez, Fran Casino, Constantinos Patsakis, Mauro Conti, Alessandro Brighente, Yuejun Guo	WP3 partners, UNIPD, LIST, UCM, ARC	Contribution from wp3 regarding research tools and h/w systems specs in relation to user requirements
0.4	20/04/2023	Andrei Costin, Vadim Bogulean,	MAG, BNR, ICO, MOT, DC	Upgrade systems specs and relate to LAZARUS

		Lelia Ataliani, Xaris Listis, Thanos Karantjias, Spyridon Papastergiou, Tareq Chihabi, Nikos Drosos, Mirko Fabbri, Alessandro Neri, Mauro Scarpa, Nikolaos Karaiskakis, Filippo Paganelli, Nikos Stratigakis, Panagiotis Markovits, Miltiadis Anastasiadis		architecture and microservices
0.5	04/05/2023	Vadim Bogulean, Lelia Ataliani, Xaris Listis, Thanos Karantjias, Nikolaos Karaiskakis, Filippo Paganelli, Nikos Stratigakis, Miltiadis Anastasiadis, Fran Casino	BNR, ICO, MAG, ARC, DC, MOT	Strategic review of system specifications and their usage within LAZARUS project and product output
0.6	18/05/2023	Ana Lucila Sandoral, Sandra Perez, Luis Alberto Martinez, Fran Casino, Constantinos Patsakis, Mauro Conti, Alessandro Brighente, Yuejun Guo	WP3 partners, UNIPD, LIST, UCM, ARC, DC	Contribution from wp3 regarding research tools and final h/w systems specs and mapping to user requirements

0.7	12/06/2023	Andrei Costin, Vadim Bogulean, Miltos Anastasiadis, Panagiotis Markovits	BNR, MOT	Finalization of system requirements and their relation to use cases and user requirements
0.8	22/06/2023	Andrei Costin, Vadim Bogulean, Lelia Ataliani, Thanos Karantjias, Spyridon Papastergiou, Tareq Chihabi, Nikos Drosos, Mirko Fabbri, Alessandro Neri, Mauro Scarpa, Nikolaos Karaiskakis, Filippo Paganelli, Nikos Stratigakis, Panagiotis Markovits, Miltiadis Anastasiadis	MAG, BNR, ICO, MOT	Use cases and in turn user requirements extracted into final detailed system requirements
0.9	27/06/2023	Kostas Patsakis	ARC	Internal review
1.0	30/06/2023	Miltos Anastasiadis	MOT	Final version

Table of Contents

List of definitions & abbreviations

Abbreviation	Definition
UC	Use Case
UR	User Requirement
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial-of-Service
SDLC	Software Development Life Cycle
CFG	Control Flow Graph
EOL/EOS	end-of-life/end-of-sale
SBOM	Software Bill of Materials
DAST	Dynamic Application Security Testing
API	Application Programming Interface
VM	Virtual Machine

1 Executive Summary

The user requirements extracted in T2.2 (End-user requirements extraction) and provided in detail in D2.2 (Initial end-user requirements) are translated into system requirements and elaborated in depth in this deliverable. Our goal is to define specific requirements for all use cases and associate end-user requirements. Therefore, Initial functional and non-functional requirements provided by the users were analysed by partners and formed the operational specifications and conceptual technology stack of the LAZARUS platform. The outcome is D2.3 which will also serve as the foundation for D2.4, integrating the use case and system requirements to define a reference architecture.

To this end, D2.3 provides the first version of LAZARUS systems requirements and identifies and draws the main LAZARUS concept building blocks that will enhance DevSecOps, taking, among others, the following steps:

- It takes into account and builds upon the outcomes of D2.2, which provide the user requirements and defines the first version of the systems requirements scenarios that will be implemented in the project.
- It adopts a proper methodology (Volere [1]) that enables authors to better and in an organized manner describe the findings and outcomes of the designing phase.
- It paves the way for defining the LAZARUS architecture, focusing on the platform's main components, trying, on the one hand, to identify the abstract user journey within each one of the them, and on the other, approach the main data flows within the system
- For each LAZARUS system requirement, it introduces a concrete description, interfaces required for integration purposes, how these requirements are grouped and how these are mapped with the functionalities identified at the presentation of the core LAZARUS elements.
- Last but not least, it builds the mapping of identified services with the functional and non-functional technical requirements, as these are analysed in deliverable D2.2.

2 Introduction

For each use case scenario defined in task T2.1, T2.3 System Requirements defines the system specifications and related interface operation specifications of every module and IT system involved. The goal is to define specific system requirements for all respective user requirements comprising the defined LAZARUS use cases, analyse them and pave the way for D2.4 LAZARUS Architecture deliverable.

2.1 Purpose

The purpose of this document is to provide the System Requirements for the LAZARUS platform. The aim is to create a platform that meets the project's goals and user requirements (as described in deliverable D2.2). These are, in turn, to be met in the design of the System Architecture deliverable D2.4.

This approach satisfies both the needs of the end-users and, at the same time, ensures that the full potential of the project goals can be reached. It adopts a proper methodology (Volere) that enables authors to better and in an organized manner describe the findings and outcomes of the designing phase.

Based on the description of the project and the purpose of this document, system requirements derived and consist of information from:

- The use cases
- The user requirements
- The general project specifications
- The proposed conceptual technology stack
- The recommended operational specifications and recommended resource usage.

2.2 Definition

While ***User Requirements*** are statements (in a natural language plus diagrams) of what services the system is expected to provide to system users and the constraints under which it must operate and may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality, ***System Requirements*** **are more detailed descriptions of the software system's functions, services, and operational constraints**. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers to satisfy the stakeholder's needs. Different requirements are needed to communicate information about a system to different types of readers [2]. Two requirement categories derived from the User Requirements (D2.2), the *Functional Requirements* and the *Non-functional Requirements,* have been used to create a high-level architecture view and subsequently derive the system requirements from it.

Functional Requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out. On the other hand, **Non-functional Requirements** are constraints on the services or functions offered by the system. Their combination and consolidation result in a clear high-level representation of the processes and properties that the system implementation would entail, translated into a more formal and technical form through the System Requirements.

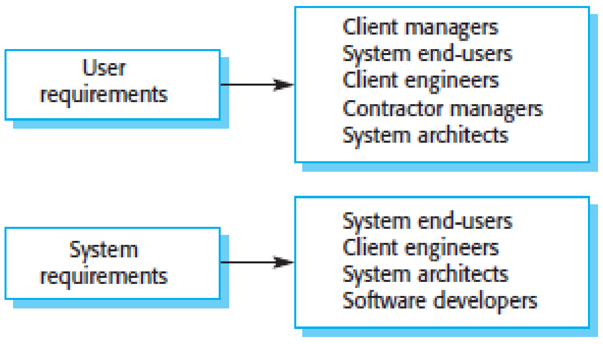

Figure 2.1: Readers of different types of requirements specification

2.3 Contribution to other Work Packages

As mentioned above, the purpose of this document is to identify and specify the list of system requirements required for the realization and evaluation and a pathway to an innovative product for DEVSECOPS of the LAZARUS platform.

Based on this and using it as a guideline, D2.3 will be used as a pathway towards:

- Setting up the LAZARUS architecture
- Detailing the services backing up LAZARUS as a platform.
- A proper integration in one platform for the provision of more advanced services that will realize the final LAZARUS system. The outcome of this process will be realized at the last (in order) task of each WP, which will provide the detailed specification of services resulting from its previous tasks.

At the same time, every effort has been made to ensure that the system requirements are aligned to and liaise with

- D2.4 - Reference Architecture
- D2.8 - Market Analysis,

which are worked on in parallel with D2.3, despite the fact that D2.4 is due later.

2.4 Structure of the document

The deliverable is divided into the following chapters:

1. **Executive Summary** – condensed information summarizing the deliverable contents
2. **Introduction** – a short introduction of the deliverable goals
3. **Methodology used** – brief presentation of the LAZARUS use cases and the associated user requirements, along with a description of the methodology used to extract them and the subsequent system requirements
4. **Use Cases - Requirements and associate system requirements** – a detailed presentation of the system requirements of the project
5. **Sub-system Integration & Operational Specifications –** a high-level description of the system integration approach and the operational specifications that will form the base for T2.4 and subsequent WP4 activities
6. **Conclusion** – conclusion of the deliverable based on previous chapters and outline of next steps
7. **References**

3 Used methodology

LAZARUS specifications follow the Volere requirements process [3], adapted to meet the LAZARUS research activities and allow agile refinement.

The Volere requirements specification process (“Volere” is the Italian word for “to wish” or “to want”) was developed to answer the need for a common language for discovering requirements and connecting them to solutions. The language needs to be understandable by businesspeople, customers, business analysts, engineers, designers, suppliers, testers or anyone else whose input is needed.

All these people have different skills and, not surprisingly, different views on what is important. A language intended for all these people must recognise the differences in people’s viewpoints and yet have a consistent way of communicating and tracing the relevant knowledge.

Since the introduction of the first version in 1995, the Volere requirements techniques have been used on projects in various domains, such as banking, air traffic control, retail, aviation, government, real-time control, business analysis, and manufacturing, just to name a few. The seemingly contradictory characteristics of rigour and flexibility have made the techniques popular as an aid to discovering, understanding, writing, and communicating requirements.

There are also management advantages like consistent input to estimating, risk management, monitoring, benefit analysis and the basis for reuse.

Volere uses established principles, models and practices and builds traceable connections between them. These connections provide a common thread between business or domain requirements, systems analysis models/deliverables, design models/components/deliverables, code and testing. The resulting framework is one that can be used regardless of modelling notation, methodology, degree of agility, development lifecycle or tool usage.

There are three groups of interconnected components:

- The Requirements Knowledge Structure is concerned with how different items of requirements knowledge relate to each other and how to trace requirements from one level to another.
- The Requirements Process is concerned with procedures and activities for how to discover, populate and disseminate the requirements for knowledge.
- The Requirements Stakeholders are the input for determining how much of the knowledge structure needs to be populated for each particular project. The project team uses the knowledge of the stakeholders to determine the order in which the work needs to be done and the appropriate level of detail.

The thinking requirements practitioner works to achieve the appropriate balance between requirements knowledge, process, and stakeholders. The purpose of Volere is to provide structure and guidance in achieving the appropriate balance for each project. The detailed contents of each one of these component groups are as follows:

- **Requirements Knowledge Structure**
    - a requirements knowledge model that acts as a filing system for requirements knowledge
    - a requirements template
    - atomic requirements structure
    - requirements traceability
    - levels of requirements
- **Requirements Process**
    - a generic requirements process for discovering requirements knowledge
    - how to get started depending on your project characteristics
    - requirements trawling/discovery techniques description and guidance
    - quality assessment techniques
    - goal analysis techniques
    - requirements estimation
    - requirements specification audit techniques
    - prioritisation techniques
- **Requirements Stakeholders**
    - stakeholder analysis techniques
    - role, knowledge, person analysis
    - roles, responsibilities and commitment
    - stakeholder feedback techniques
    - conflict resolution techniques
    - requirements viewpoints

One key-methodology technique is to follow the VOLERE Requirements Specification Template, which is intended for use as a basis for requirements specifications. It provides sections for each of the requirement types appropriate to today's software systems.

The VOLERE template is a compartmentalized container for requirements, and it categorizes the requirements into types that prove useful for the purpose of recognition and elicitation. These types are:

- Functional requirements: describe what the product has to do or which processing actions it must take
- Non-functional requirements: properties that the functions must have, such as performance and usability
- Project constraints: restrictions on the product due to the budget or the time available to build the product
- Design constraints: restrictions on how the product must be designed
- Project drivers: business-related forces
- Project issues: conditions under which the project will be carried out

The above categories are covered in related WP2 deliverables, such as D2.1, D2.2 and D2.8.
Whereas the template is a guide to what to write about, the requirements shell or snow card is a guide to how to write it. Individual requirements have a structure – a set of attributes, where each attribute

contributes something to understanding the requirement, to the precision of the requirement, and thereby to the accuracy of the product's development.

Based on this adapted methodology, each LAZARUS requirement shell specification is identified as follows:

Service Contract Name	**<Name>**
Specification ID	Unique identifier with the following taxonomy: <COMPONENT_ID>-<TYPE>-<NUMBER> Once attributed, an identifier cannot be changed or reused to represent a different concept.
Specification Type	Functional (F) Look and Feel (LF) Usability (U) Performance (P) Operational and Environmental (O) Maintainability and Support (M) Security (S) Compliance (C)
Category / Grouping	Main category of the requirement
Dependencies	Dependencies that the requirement has from other ones
Stakeholders	Based on the identified Stakeholders in paragraph 3.1.1)
User Requirement(s)	the user/stakeholder requirements from (D2.2)
Priority	(High, Medium, Low) based on the corresponding User Requirement(s)
Description and Rationale	Brief Description of the module / service / component
Tool(s) / Solution(s)	Project tool / solution that shall be used

Table 3.1: LAZARUS Service Specification Template

Towards this, for each LAZARUS service, its interface operation specification is identified as follows:

Operation Name	*Name*
Operation ID	<Specification ID> . <Operation ID>……
Description	(Provide a description for the operation – not more than one small paragraph), i.e. This operation will support the… For all these, it will be mandatory to provide…

Depended / Related Services	(Name the depended / Related services)
Input	(Describe if any)
Output	(Describe if any)

Table 3.2: LAZARUS End Service Interface Operation Specification

3.1 LAZARUS Stakeholders

To successfully identify the LAZARUS services and their characteristics, the project had at first to try to identify its different fictional user-types and characters. To do this, we performed the following:

- First of all, we tried to understand their needs, experiences, behaviours and goals, recognising that different people have different needs and expectations. Following this purpose, we mainly focus on the deliverable D2.2 and the user-requirements' gathering process.
- Secondly, we adopted the role-based perspective, which is goal-directed and it focuses on the user's behaviour. The different user types of role-based perspectives are massively data-driven and incorporate data from both qualitative and quantitative sources.

Therefore, within LAZARUS, we identify three different main user-types:

- The ***Customer***, who represents the end users of LAZARUS and generally users of the LAZARUS platform – product - and request access to the LAZARUS set of features and services
- The ***LAZARUS administrator,*** who represents one or more users that actually administer the LAZARUS platform and guarantee the proper operation of its services
- The ***Vendor / Supplier***, since LAZARUS will also be provided as a set of cloud services and tools.

Focusing on the Customer side, we can recognize the following different roles:

- The **Software Engineer**, who is the key role that makes full use of the DevOps/DevSecOps pipelines and actually develops the tested software product/service.
- The **Quality Assurance Professional**, who is responsible for thoroughly testing the deployed software so as to identify, categorize and report in detail possible bugs, security flaws and other functionality and design issues.
- The ***Security Analyst***, who, most of the time plays a vital role in keeping an organization's proprietary and sensitive information secure. Usually, he/she is responsible to identify and correct flaws in the organization's security systems, solutions, and programs while recommending specific measures that can improve its overall security posture.
- The ***Security Professional and Expert***, who search for vulnerabilities across the organization's systems, inspecting for any attacks and intrusions, recognizing potential threats, and designing various strategies and defensive systems against intruders.
- The ***Manager and/or Administrator***, who manage things in the IT department, make the strategy for how to run the department, design the best policies, give direction to the department, and lead from

the front.

3.2 Mapping of use cases and user requirements

In order to be consistent and concise and keep a flow between work and deliverables in WP2, the table below provides a global view of the LAZARUS use cases and the associated user requirements and priority status, that formed the basis for working and defining the system specifications.

Use Case	Use Case Title	User Requirement	User Requirement Code	User Requirement Title	MoSCoW Priority
General		UR-C-N-1	LZR-GR1	Compliance with existing security standards	MUST
		UR-C-F-1	LZR-GR2	Automated Compliance Checks	SHOULD
		UR-C-F-2	LZR-GR3	Standards-based Policy Enforcement	SHOULD
		UR-C-F-3	LZR-GR4	Mapping to Regulatory Frameworks	COULD
		UR-C-F-4	LZR-GR5	Risk Management	MUST
		UR-CE-N-1	LZR-GR6	Scalability and Adaptability	MUST
		UR-CE-N-2	LZR-GR7	Portability	SHOULD
		UR-C-F-5	LZR-GR8	Incident Management	MUST
		UR-C-F-6	LZR-GR9	Training and Awareness	COULD
		UR-C-F-7	LZR-GR10	Authentication	MUST
		UR-C-F-8	LZR-GR11	Authorization	MUST
		UR-C-F-9	LZR-GR12	Role-based access control	SHOULD
		UR-C-F-10	LZR-GR13	Module-specific access	COULD

		UR-C-F-11	LZR-GR14	Administrative Interfaces	SHOULD
		UR-C-F-12	LZR-GR15	Auditability	MUST
		UR-C-F-13	LZR-GR16	Verifiability	MUST
		UR-C-F-14	LZR-GR17	Documentation and Reporting	MUST
		UR-C-F-15	LZR-GR18	Machine-accessible Reporting	SHOULD
		UR-C-N-2	LZR-GR19	Source Code Confidentiality	MUST
		UR-C-N-3	LZR-GR20	Efficient Processing	MUST
		UR-C-N-4	LZR-GR21	System Stability	MUST
		UR-C-N-5	LZR-GR22	Multi-tenancy	MUST
UC-1	Issue detection regarding secrets management	UR-C-F-16	LZR-SM1	Reliable Hardcoded Secret Detection	MUST
		UR-C-F-17	LZR-SM2	Reliable Unencrypted Secret Detection	MUST
		UR-C-F-18	LZR-SM3	Reliable Stored Secret Detection	MUST
		UR-C-F-19	LZR-SM4	Secret Detection in Code History	COULD
UC-2	Code Linting	UR-CE-F-1	LZR-CL1	Reliable Vulnerability Detection in Source Code	MUST
		UR-C-F-20	LZR-CL2	Formatting and Styling Issue Detection	SHOULD

		UR-C-F-21	LZR-CL3	Coding and Deployment Best Practices Suggestions	COULD
		UR-C-F-22	LZR-CL4	Source Code Pattern-based Simulation	COULD
		UR-CE-F-2	LZR-CL5	Source Code Quality and Complexity Metrics	COULD
		UR-CE-F-3	LZR-CL6	Safety and Security Coding Standards Support	COULD
		UR-C-F-23	LZR-CL7	Out-of-the-box Certification	COULD
UC-3	Static Code Analysis	UR-CE-F-4	LZR-SA1	Source Code-based Data Flow Analysis	MUST
		UR-CE-F-5	LZR-SA2	Source Code-based Control Flow Graphs	MUST
		UR-C-F-24	LZR-SA3	Source Code-based Taint Analysis	COULD
		UR-C-F-25	LZR-SA4	Source Code-based Lexical Analysis	COULD
		UR-C-F-26	LZR-SA5	Cryptography-related Issue Detection	COULD
UC-4	SQL Injection Vulnerability Detection	UR-CE-F-6	LZR-SI1	DAST-based SQL Injection Vulnerability Detection	MUST
		UR-CE-F-7	LZR-SI2	Query Input Whitelisting Verification	SHOULD

		UR-CE-F-8	LZR-SI3	Query Input Escape Verification	SHOULD
UC-5	Fuzzing	UR-CE-F-9	LZR-FZ1	Fuzzing Service Configurability	SHOULD
		UR-C-F-27	LZR-FZ2	Protocol Fuzzing Services	COULD
		UR-C-F-28	LZR-FZ3	File Format Fuzzing Services	COULD
UC-6	CVE Scan	UR-CE-F-10	LZR-CV1	CVE Scanning Reliability	MUST
		UR-CE-F-11	LZR-CV2	CVE Scanning Service Accessibility	MUST
		UR-CE-F-12	LZR-CV3	CVE Scanning Service Configurability	SHOULD
		UR-C-F-29	LZR-CV4	Unnecessary Direct and Transitive Dependencies Detection	COULD
		UR-C-F-30	LZR-CV5	Project Dependencies Health Check	COULD
UC-7	Container Vulnerability Scanning	UR-CE-F-13	LZR-CS1	Container Vulnerability Scanning Reliability	MUST
		UR-C-F-31	LZR-CS2	Container-based Compliance Validation Detection	SHOULD
		UR-C-F-32	LZR-CS3	Container-related Best Practice Suggestions	COULD
UC-8	Detection of Network Attacks & DDoS	UR-CE-F-14	LZR-NA1	Network Tool Reliability	MUST
		UR-CE-F-15	LZR-NA2	Network Tool Configurability	SHOULD

		UR-C-F-33	LZR-NA3	Incident Response and Recovery Policy Suggestions	COULD

Table 3.4: LAZARUS End User Requirements

4 Use Cases - Requirements and associated system requirements

A full analysis of the LAZARUS system requirements is presented in this Chapter.

4.1 System requirements

While the user requirements for a system should describe the functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge, system requirements are expanded versions of the user requirements that software engineers use as the starting point for the system design. They add detail and explain how the system should provide the user requirements. They may be used as part of the contract for the implementation of the system and should therefore be a complete and detailed specification of the whole system. Ideally, the system requirements should only describe the external behaviour of the system and its operational constraints. They should not be concerned with how the system should be designed or implemented.

However, at the level of detail required to specify a complex software system completely, it is neither possible nor desirable to exclude all design information. Therefore, a mixed approach has been adopted for this deliverable to provide in a concise and clear manner each specification and, at the same time, provide the required level of detail so that they can be used as a concrete development roadmap.

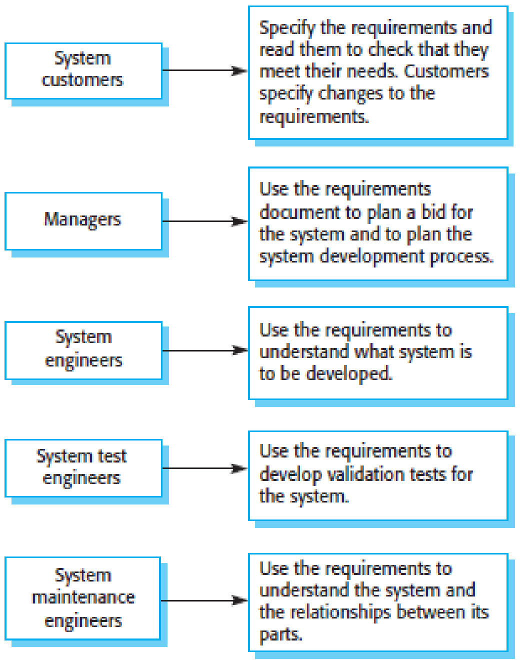
System customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers
Use the requirements to understand what system is to be developed.
System test engineers
Use the requirements to develop validation tests for the system.
System maintenance engineers
Use the requirements to understand the system and the relationships between its parts.

Figure 4.1: Requirements Purpose per Stakeholder Type

4.1.1 Summary

The following table summarizes the derived system specifications, their interface specifications and their mapping to the use cases and the user requirements:

Use Case	Use Case Title	User Requirement Title	System Specification	Interface Specifications	MoSCoW Priority
General		Compliance with existing security standards	Compliance with existing security standards	-	MUST
		Automated Compliance Checks	Application, Infrastructure & Configuration compliance checks	Application Compliance Check Infrastructure Compliance Check Configuration Compliance Check	SHOULD
		Standards-based Policy Enforcement	Definition and enforcement of ISO-based policies	Policy Definition Policy Enforcement	SHOULD
		Mapping to Regulatory Frameworks	Security controls and requirements mapping to specific laws and regulations	Security control Mapping	COULD
		Risk Management	Framework provision for identifying, assessing, and managing information security risks	Define Risk Manage Risk	MUST
		Scalability and Adaptability	Modular and flexible architecture	-	MUST
		Portability	Machine-independent platform implementation	-	SHOULD

	Incident Management	Incident Management Support	Define Incident Manage Incident	MUST
	Training and Awareness	User Training & Awareness Module	Train User in Security Management Train User in Security Awareness	COULD
	Authentication	User Account Management	User Registration Change Password Change Account Settings Delete Account	MUST
	Authorization	Authorization Checks	Authorization Check Role-based Access Check Module-based Access Check Permission-based Access Check	MUST
	Role-based access control			SHOULD
	Module-specific access			COULD
	Administrative Interfaces	Administrative Interfaces	Create User Modify User Delete User Assign Roles Grant Module Access Grant Permission	SHOULD
	Auditability	Auditability	Audit Trail Immutable Trail	MUST
	Verifiability	Verifiability	Record Action	MUST
	Documentation and Reporting	Documentation and Reporting	Produce Documentation	MUST
	Machine-accessible Reporting	Machine-accessible Reporting	Produce Machine-accessible Reporting	SHOULD
	Source Code Confidentiality	Source Code Confidentiality	Encrypt Source Code Copy Delete Source Code Copy After Processing	MUST

		Efficient Processing	Efficient Processing	Asynchronous Processing Configure Time Limitation for Synchronous Service	MUST
		System Stability	System Stability	No Data Loss Guarantee Change System State Gracefully	MUST
		Multi-tenancy	Multi-tenancy	Separate Tenants Isolate Processes	MUST
UC-1	Issue detection regarding secrets management	Reliable Hardcoded Secret Detection	Secret Detection	Scan for Secrets Scan Commit History Configure Secret Scan	MUST
		Reliable Unencrypted Secret Detection			MUST
		Reliable Stored Secret Detection			MUST
		Secret Detection in Code History			COULD
UC-2	Code Linting	Reliable Vulnerability Detection in Source Code	Vulnerability Detection in Source Code	Scan for Errors Scan for Styling Issues Suggest Coding Best Practices Simulate Patterns Provide Quality Metrics Check Code for Standards Compliance Certify Code	MUST
		Formatting and Styling Issue Detection			SHOULD
		Coding and Deployment Best Practices Suggestions			COULD
		Source Code Pattern-based Simulation			COULD
		Source Code Quality and Complexity Metrics			COULD

		Safety and Security Coding Standards Support			COULD
		Out-of-the-box Certification			COULD
UC-3	Static Code Analysis	Source Code-based Data Flow Analysis	Static Code Analysis	Apply Data Flow Analysis Create CFG Apply Taint Analysis Apply Lexical Analysis Check for Cryptography-related Issues	MUST
		Source Code-based Control Flow Graphs			MUST
		Source Code-based Taint Analysis			COULD
		Source Code-based Lexical Analysis			COULD
		Cryptography-related Issue Detection			COULD
UC-4	SQL Injection Vulnerability Detection	DAST-based SQL Injection Vulnerability Detection	SQL Injection Vulnerability Detection	Check for SQL Injections Check Query Input	MUST
		Query Input Whitelisting Verification			SHOULD
		Query Input Escape Verification			SHOULD
UC-5	Fuzzing	Fuzzing Service Configurability	Fuzzing	Fuzz Target Apply Protocol Fuzzing Apply File Format Fuzzing Configure Fuzzing	SHOULD
		Protocol Fuzzing Services			COULD

		File Format Fuzzing Services			COULD
UC-6	CVE Scan	CVE Scanning Reliability	CVE Scan	Scan for CVEs Scan for Dependencies Check Project Dependencies Health Status Configure CVE Scan	MUST
		CVE Scanning Service Accessibility			MUST
		CVE Scanning Service Configurability			SHOULD
		Unnecessary Direct and Transitive Dependencies Detection			COULD
		Project Dependencies Health Check			COULD
UC-7	Container Vulnerability Scanning	Container Vulnerability Scanning Reliability	Container Vulnerability Scan	Scan Containers Scan Containers for Compliance Validations Provide Container-related Suggestions	MUST
		Container-based Compliance Validation Detection			SHOULD
		Container-related Best Practice Suggestions			COULD
UC-8	Detection of Network Attacks & DDoS	Network Tool Reliability	Detection of Network Attacks	Scan Network Suggest Best Practices Based on Network	MUST
		Network Tool Configurability			SHOULD

		Incident Response and Recovery Policy Suggestions			COULD

Table 4.1: LAZARUS System Requirements Summary

4.2 General Specifications

4.2.1 Compliance with existing security standards

4.2.1.1 Specification Overview

Service Contract Name	**Compliance with existing security standards**
Specification ID	GENERAL-C-10
Specification Type	Compliance (C)
Category / Grouping	Platform Security
Dependencies	Standards-based Policy Enforcement, Authentication, Authorization, Role-based access control, Module-specific access, Administrative Interfaces, Auditability, Verifiability, Source Code Confidentiality, System Stability, Multi-tenancy
Stakeholders	Vendor/Supplier, Software Engineers, Quality Assurance Professionals, Security Analysts and Security Experts
User Requirement(s)	LZR-GR1 (UR-C-N-1)
Priority	High
Description and Rationale	Compliance with existing security standards (such as ISO27001, ISO 27002, ISO 27005, ISO 27035) is a crucial property for the LAZARUS platform, as the platform itself is meant to provide secure DevSecOps services.
Tool(s) / Solution(s)	All project components will contribute towards this specification

4.2.1.2 Interface Specifications

None

4.2.2 Application, Infrastructure & Configuration compliance checks

4.2.2.1 Specification Overview

Service Contract Name	**Application, Infrastructure & Configuration compliance checks**
Specification ID	GENERAL-F-10
Specification Type	Functional (F)
Category / Grouping	Automated Compliance
Dependencies	Mapping to Regulatory Frameworks, Standards-based Policy Enforcement
Stakeholders	Quality Assurance Professionals, Security Analysts and Managers
User Requirement(s)	LZR-GR2 (UR-C-F-1)
Priority	Medium
Description and Rationale	Multi-level compliance checks will enable an organization to ensure and be able to prove the alignment of their applications and infrastructure with specific standards and regulations easily.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.2.2 Interface Specifications

Operation Name	*applicationComplianceCheck*
Operation ID	GENERAL-F-10.1
Description	Automated compliance check regarding application security controls, testing, change management and vulnerability scanning
Depended / Related Services	Infrastructure Compliance Check, Configuration Compliance Check, Standards-based Policy Enforcement, Mapping to Regulatory Frameworks
Input	Compliance Evidence
Output	Compliance Status Report

Operation Name	*infrastructureComplianceCheck*
Operation ID	GENERAL-F-10.2
Description	Automated compliance check regarding infrastructure security controls, testing, change management and vulnerability scanning
Depended / Related Services	Application Compliance Check, Configuration Compliance Check, Standards-based Policy Enforcement, Mapping to Regulatory Frameworks
Input	Compliance Evidence

Output	Compliance Status Report

Operation Name	*configurationComplianceCheck*
Operation ID	GENERAL-F-10.3
Description	Automated compliance check regarding application & infrastructure configuration
Depended / Related Services	Application Compliance Check, Infrastructure Compliance Check, Standards-based Policy Enforcement, Mapping to Regulatory Frameworks
Input	Compliance evidence in the form of configuration files
Output	Compliance Status Report

4.2.3 Definition and enforcement of ISO-based policies

4.2.3.1 Specification Overview

Service Contract Name	**Definition and enforcement of ISO-based policies**
Specification ID	GENERAL-F-20
Specification Type	Functional (F)
Category / Grouping	Automated Compliance
Dependencies	None
Stakeholders	Security Analysts and Managers
User Requirement(s)	LZR-GR3 (UR-C-F-2)
Priority	Medium
Description and Rationale	By enabling organizations to define and enforce policies based on the mentioned ISO standards (ISO 27001, ISO 27002, ISO 27005, and ISO 27035) and by incorporating these standards into the development pipeline, LAZARUS provides a guarantee of compliance with best practices and regulatory requirements.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.3.2 Interface Specifications

Operation Name	*policyDefinition*
Operation ID	GENERAL-F-20.1
Description	ISO-based policy definition
Depended / Related Services	Application Compliance Check, Infrastructure Compliance Check, Configuration Compliance Check, Mapping to Regulatory Frameworks
Input	Policy Rules through user input (UI) or XML/JSON file
Output	New Policy Entity

Operation Name	*policyEnforcement*
Operation ID	GENERAL-F-20.2
Description	ISO-based policy enforcement
Depended / Related Services	Application Compliance Check, Infrastructure Compliance Check, Configuration Compliance Check, Mapping to Regulatory Frameworks
Input	Defined policy entity and integration points with user environment
Output	Policy enforcement in assigned target(s) and alerting

4.2.4 Security controls and requirements mapping to specific laws and regulations

4.2.4.1 Specification Overview

Service Contract Name	**Security controls and requirements mapping to specific laws and regulations**
Specification ID	GENERAL-F-30
Specification Type	Functional (F)
Category / Grouping	Automated Compliance
Dependencies	None
Stakeholders	Quality Assurance Professionals, Security Analysts and Managers
User Requirement(s)	LZR-GR4 (UR-C-F-3)
Priority	Low
Description and Rationale	LAZARUS should have the capability to map security controls and requirements to specific laws and regulations, such as the NIS Directive, Directive 2002/21/EC, and

	Directive (EU) 2016/1148. This simplifies the compliance process and helps organizations demonstrate their adherence to the relevant legal frameworks
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.4.2 Interface Specifications

Operation Name	*securityControlMapping*
Operation ID	GENERAL-F-30.1
Description	Mapping of security controls and requirements to specific laws and regulations to simplify the compliance process and demonstrate adherence to relevant legal frameworks.
Depended / Related Services	Application Compliance Check, Infrastructure Compliance Check, Configuration Compliance Check, Standards-based Policy Enforcement
Input	Integration points to security controls and user environment
Output	Status Report

4.2.5 Framework provision for identifying, assessing, and managing information security risks

4.2.5.1 Specification Overview

Service Contract Name	**Framework provision for identifying, assessing, and managing information security risks**
Specification ID	GENERAL-F-40
Specification Type	Functional (F)
Category / Grouping	Risk & Incident Management
Dependencies	None
Stakeholders	Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-GR5 (UR-C-F-4)
Priority	High

Description and Rationale	In line with the ISO 27005 standard, LAZARUS should facilitate risk management by providing a framework for identifying, assessing, and managing information security risks throughout the development life cycle.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.5.2 Interface Specifications

Operation Name	*defineRisk*
Operation ID	GENERAL-F-40.1
Description	Users must be able to define risk categories and specific information security risks and assess them.
Depended / Related Services	None
Input	User input
Output	Status Report

Operation Name	*manageRisk*
Operation ID	GENERAL-F-40.2
Description	Users must be able to re-evaluate defined risks and manage them in the case of their occurrence. All related decisions and information will be stored for future use.
Depended / Related Services	None
Input	User input
Output	Status Report

4.2.6 Modular and flexible architecture

4.2.6.1 Specification Overview

Service Contract Name	**Modular and flexible architecture**
Specification ID	GENERAL-M-10

Specification Type	Maintainability and Support (M)
Category / Grouping	System Architecture
Dependencies	None
Stakeholders	Security Analysts and Managers
User Requirement(s)	LZR-GR6 (UR-CE-N-1)
Priority	High
Description and Rationale	The LAZARUS system should be modular and flexible. As security laws and regulations evolve, a DevSecOps tool should be scalable and adaptable to accommodate new requirements and standards. This ensures that organizations can maintain compliance without significant disruptions to their development processes.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.6.2 Interface Specifications

None

4.2.7 Machine-independent platform implementation

4.2.7.1 Specification Overview

Service Contract Name	**Machine-independent platform implementation**
Specification ID	GENERAL-O-10
Specification Type	Operational and Environmental (O)
Category / Grouping	System Architecture
Dependencies	None
Stakeholders	Vendor/Supplier, Administrators and Managers
User Requirement(s)	LZR-GR7 (UR-CE-N-2)
Priority	Medium

Description and Rationale	The LAZARUS system should be device-agnostic (run on diverse operating systems and hardware) and able to replicate and be deployed across different infrastructures with low effort.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.7.2 Interface Specifications

None

4.2.8 Incident Management Support

4.2.8.1 Specification Overview

Service Contract Name	**Incident Management Support**
Specification ID	GENERAL-F-50
Specification Type	Functional (F)
Category / Grouping	Risk & Incident Management
Dependencies	None
Stakeholders	Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-GR8 (UR-C-F-5)
Priority	High
Description and Rationale	As per the ISO 27035 standard, the LAZARUS system should support incident management capabilities, including preparing for, identifying, assessing, responding to, and learning from information security incidents. This can help organizations minimize the impact of security incidents and ensure timely recovery.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.8.2 Interface Specifications

Operation Name	*defineIncident*
Operation ID	GENERAL-F-50.1
Description	Users must be able to define incident categories and specific incidents and prepare processes and procedures to address them when they occur. All related information will be stored for future use.
Depended / Related Services	None
Input	User input
Output	Status Report

Operation Name	*manageIncident*
Operation ID	GENERAL-F-50.2
Description	Users must be able to manage incidents when they occur by applying or modifying their defined processes and procedures and learning from the outcome. All related decisions and information will be stored for future use.
Depended / Related Services	None
Input	User input
Output	Status Report

4.2.9 User Training & Awareness Module

4.2.9.1 Specification Overview

Service Contract Name	**User Training & Awareness Module**
Specification ID	GENERAL-M-20
Specification Type	Maintainability and Support (M)
Category / Grouping	Risk & Incident Management
Dependencies	None
Stakeholders	Security Analysts, Security Experts, Administrators and Managers
User Requirement(s)	LZR-GR9 (UR-C-F-6)
Priority	Low

Description and Rationale	A DevSecOps tool can include training and awareness modules to help educate developers and other stakeholders on security best practices and the importance of compliance. This can help foster a security-conscious culture and reduce the likelihood of security incidents due to human error.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.9.2 Interface Specifications

Operation Name	*trainUserInSecurityManagement*
Operation ID	GENERAL-M-20.1
Description	Users will be trained through a special program regarding security management and compliance.
Depended / Related Services	None
Input	User progress in training program
Output	Training Proof & Results Report

Operation Name	*trainUserInSecurityAwareness*
Operation ID	GENERAL-M-20.2
Description	Users will be trained through a special program regarding security awareness.
Depended / Related Services	None
Input	User progress in training program
Output	Training Proof & Results Report

4.2.10 User Account Management

4.2.10.1 Specification Overview

Service Contract Name	**User Account Management**
Specification ID	GENERAL-S-10
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization

Dependencies	Administrative Interfaces
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR10 (UR-C-F-7)
Priority	High
Description and Rationale	The LAZARUS system shall require users to authenticate themselves before accessing any of its modules. To achieve this, various related user account management functionalities need to be implemented.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.10.2 Interface Specifications

Operation Name	*registerUser*
Operation ID	GENERAL-S-10.1
Description	User registration and approval by the responsible administrator(s).
Depended / Related Services	None
Input	User registration form
Output	User Account Creation

Operation Name	*changePassword*
Operation ID	GENERAL-S-10.2
Description	Change password feature
Depended / Related Services	None
Input	Change password request by user
Output	Password Change

Operation Name	*changeAccountSettings*
Operation ID	GENERAL-S-10.3

Description	Ability of user to change the settings of their account
Depended / Related Services	None
Input	Change setting request by user
Output	User Settings change

Operation Name	*deleteAccount*
Operation ID	GENERAL-S-10.4
Description	Ability of user to delete their account
Depended / Related Services	None
Input	Delete account request by user and approval by the responsible administrator(s).
Output	User Account Deletion

4.2.11 Authorization Checks

4.2.11.1 Specification Overview

Service Contract Name	**Authorization Checks**
Specification ID	GENERAL-S-20
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization
Dependencies	Administrative Interfaces
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR11 (UR-C-F-8), LZR-GR12 (UR-C-F-9), LZR-GR13 (UR-C-F-10)
Priority	High
Description and Rationale	Only authorized access to modules/resources should be allowed, along with role-based and module-based authorization checks and fine-grained permissions.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.11.2 Interface Specifications

Operation Name	*authorizedAccess*
Operation ID	GENERAL-S-20.1
Description	Users should only be allowed access to resources they are allowed to access.
Depended / Related Services	None
Input	User access request to resources
Output	Access Approval or Denial

Operation Name	*roleBasedAccessCheck*
Operation ID	GENERAL-S-20.2
Description	Users should only be allowed access to resources allowed by their assigned role(s).
Depended / Related Services	None
Input	User access request to resources
Output	Access Approval or Denial

Operation Name	*moduleBasedAccessCheck*
Operation ID	GENERAL-S-20.3
Description	Users should only be allowed access to modules they are allowed to access.
Depended / Related Services	None
Input	User access request to module
Output	Access Approval or Denial

Operation Name	*permissionAccessCheck*
Operation ID	GENERAL-S-20.4
Description	Users should only be allowed to execute actions and access resources allowed by their fine-grained permissions.
Depended / Related Services	None

Input	User access request to resources/execute action
Output	Access/Action Approval or Denial

4.2.12 Administrative Interfaces

4.2.12.1 Specification Overview

Service Contract Name	**Administrative Interfaces**
Specification ID	GENERAL-F-60
Specification Type	Functional (F)
Category / Grouping	Authentication & Authorization
Dependencies	Administrative Interfaces
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR14 (UR-C-F-11)
Priority	Medium
Description and Rationale	LAZARUS should provide administrative interfaces for managing user accounts, roles, and module-level access controls. The interfaces shall enable administrators to create, modify, and delete user accounts, assign roles, and grant module-level access permissions to users and roles.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.12.2 Interface Specifications

Operation Name	*adminCreateUser*
Operation ID	GENERAL-F-60.1
Description	An administrator creates a new user account.
Depended / Related Services	None
Input	Administrator request for new user account creation
Output	User Account Creation

Operation Name	*adminModifyUser*
Operation ID	GENERAL-F-60.2
Description	An administrator modifies a user account.
Depended / Related Services	None
Input	Administrator request for user account modification
Output	User Account Modification

Operation Name	*adminDeleteUser*
Operation ID	GENERAL-F-60.3
Description	An administrator deletes a user account.
Depended / Related Services	None
Input	Administrator request for user account deletion
Output	User Account Deletion

Operation Name	*adminAssignRoles*
Operation ID	GENERAL-F-60.4
Description	An administrator assigns roles to a user account.
Depended / Related Services	None
Input	Administrator request for role assignment to user
Output	Role assignment to user

Operation Name	*adminGrantModuleAccess*
Operation ID	GENERAL-F-60.5
Description	An administrator grants module access to a user account.
Depended / Related Services	None
Input	Administrator request for a user to acquire access to a module
Output	User acquires access to module

Operation Name	*adminGrantPermission*
Operation ID	GENERAL-F-60.6
Description	An administrator grants a fine-grained permission to a user account.
Depended / Related Services	None
Input	Administrator request for a user to acquire a fine-grained permission
Output	User acquires permission

4.2.13 Auditability

4.2.13.1 Specification Overview

Service Contract Name	**Auditability**
Specification ID	GENERAL-S-30
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR15 (UR-C-F-12)
Priority	High
Description and Rationale	The LAZARUS system should maintain an audit trail of all authentication and authorization events, including login attempts, module accesses, and any changes made to user accounts, roles, or module-level access controls. The audit trail shall be accessible only to authorized users and shall be protected against unauthorized access, modification, or deletion.
Tool(s) / Solution(s)	LAZARUS Main Platform, Blockchain

4.2.13.2 Interface Specifications

Operation Name	*auditTrail*
Operation ID	GENERAL-S-30.1
Description	Processes and/or relevant technology must be set up that will maintain an audit trail of all authentication and authorization events, including login attempts, module accesses, and any changes made to user accounts, roles, or module-level access controls.
Depended / Related Services	None
Input	Authentication/Authorization Event
Output	Immutable event log added to audit trail

Operation Name	*immutableTrail*
Operation ID	GENERAL-S-30.2
Description	Processes and/or relevant technology must be set up that guarantee that the audit trail shall be accessible only to authorized users and shall be protected against unauthorized access, modification, or deletion.
Depended / Related Services	None
Input	Event Log
Output	Guarantee of immutability

4.2.14 Verifiability

4.2.14.1 Specification Overview

Service Contract Name	**Verifiability**
Specification ID	GENERAL-S-40
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR16 (UR-C-F-13)
Priority	High

Description and Rationale	All data generated by LAZARUS must be verifiable.
Tool(s) / Solution(s)	LAZARUS Main Platform, Blockchain

4.2.14.2 Interface Specifications

Operation Name	*recordAction*
Operation ID	GENERAL-S-40.1
Description	Executed process or action is recorded, along with any related information, e.g. actor and timestamp.
Depended / Related Services	None
Input	Action Event
Output	Immutable record of action

4.2.15 Documentation and Reporting

4.2.15.1 Specification Overview

Service Contract Name	**Documentation and Reporting**
Specification ID	GENERAL-M-30
Specification Type	Maintainability and Support (M)
Category / Grouping	Reporting
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR17 (UR-C-F-14)
Priority	High
Description and Rationale	The LAZARUS platform should support comprehensive documentation and reporting capabilities, enabling organizations to demonstrate their compliance with the mentioned security standards and regulations. This should include generating

	audit-ready reports, tracking remediation efforts, and providing evidence of security controls and risk management practices.
Tool(s) / Solution(s)	LAZARUS Main Platform, Blockchain

4.2.15.2 Interface Specifications

Operation Name	*produceDocumentation*
Operation ID	GENERAL-M-30.1
Description	Creation of detailed, relevant, comprehensive and specific (as requested) documentation.
Depended / Related Services	None
Input	Documentation procurement request
Output	Documentation file(s)

4.2.16 Machine-accessible Reporting

4.2.16.1 Specification Overview

Service Contract Name	**Machine-accessible Reporting**
Specification ID	GENERAL-U-10
Specification Type	Usability (U)
Category / Grouping	Reporting
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR18 (UR-C-F-15)
Priority	Medium

Description and Rationale	The reporting provided by LAZARUS should provide the option to output any report in a standard machine-readable format (JSON/XML) so it can be parsed by an automated tool/dashboard.
Tool(s) / Solution(s)	LAZARUS Main Platform, Blockchain

4.2.16.2 Interface Specifications

Operation Name	*ProduceMachineAccessibleReporting*
Operation ID	GENERAL-U-10.1
Description	Creation of machine-accessible (XML/JSON format) reports
Depended / Related Services	None
Input	Request for relevant machine-accessible report
Output	XML/JSON file(s)

4.2.17 Source Code Confidentiality

4.2.17.1 Specification Overview

Service Contract Name	**Source Code Confidentiality**
Specification ID	GENERAL-S-50
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR19 (UR-C-N-2)
Priority	High
Description and Rationale	Modules that have access to a repository must use the repository source code solely for the purpose of their functionality and must not export, send, or otherwise use any external tools or services that would expose the source code outside of the

	system without explicit authorization. Any use of the source code must be restricted to the specific context and scope of the module, and the source code must not be exposed to unauthorized users or systems.
Tool(s) / Solution(s)	LAZARUS Main Platform

4.2.17.2 Interface Specifications

Operation Name	*encryptSourceCodeCopy*
Operation ID	GENERAL-S-50.1
Description	Acquired source code copy should be encrypted and used only in the context of the request operation(s).
Depended / Related Services	None
Input	Source Code Copy
Output	Encrypted Source Code Copy

Operation Name	*deleteSourceCodeCopyAfterProcessing*
Operation ID	GENERAL-S-50.2
Description	Acquired source code copy should be securely deleted immediately after the relevant requested operations have finished.
Depended / Related Services	None
Input	Encrypted Source Code Copy
Output	Secure Source Code Copy Deletion

4.2.18 Efficient Processing

4.2.18.1 Specification Overview

Service Contract Name	**Efficient Processing**
Specification ID	GENERAL-P-10
Specification Type	Performance (P)

Category / Grouping	Service Quality
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Administrators
User Requirement(s)	LZR-GR20 (UR-C-N-3)
Priority	High
Description and Rationale	Long-running or synchronous processes in modules provided by LAZARUS and commonly used in SDLC "software development life cycle" should be avoided to provide high performance and usability.
Tool(s) / Solution(s)	All project components will contribute towards this specification

4.2.18.2 Interface Specifications

Operation Name	*asynchronousProcessing*
Operation ID	GENERAL-P-10.1
Description	Provided services should be implemented and executed in an asynchronous way by default.
Depended / Related Services	None
Input	None
Output	None

Operation Name	*configureTimeLimitationForSynchronousService*
Operation ID	GENERAL-P-10.2
Description	Users should be able to configure time limitations for the services that run in a synchronous manner.
Depended / Related Services	None
Input	None
Output	None

4.2.19 System Stability

4.2.19.1 Specification Overview

Service Contract Name	**System Stability**
Specification ID	GENERAL-U-20
Specification Type	Usability (U)
Category / Grouping	Service Quality
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR21 (UR-C-N-4)
Priority	High
Description and Rationale	The system shall provide a fail-safe configuration. i.e. in case of an unexpected event or error, the system shall go to a safe state.
Tool(s) / Solution(s)	All project components will contribute towards this specification

4.2.19.2 Interface Specifications

Operation Name	*guaranteeNoDataLoss*
Operation ID	GENERAL-U-20.1
Description	Processes and/or relevant technology must be set up that guarantee the integrity of data in case of unexpected events or errors.
Depended / Related Services	None
Input	Unexpected Event/Error
Output	Intact User Data

Operation Name	*changeSystemStateGracefully*
Operation ID	GENERAL-U-20.2

Description	Processes and/or relevant technology must be set up that guarantee a graceful state change to a safe system state in case of unexpected events or errors.
Depended / Related Services	None
Input	Unexpected Event/Error
Output	Graceful state change to a safe system state

4.2.20 Multi-tenancy

4.2.20.1 Specification Overview

Service Contract Name	**Multi-tenancy**
Specification ID	GENERAL-S-60
Specification Type	Security (S)
Category / Grouping	Authentication & Authorization
Dependencies	None
Stakeholders	Vendor/Supplier, LAZARUS Administrator, Software Engineers, Quality Assurance Professionals, Administrators
User Requirement(s)	LZR-GR22 (UR-C-N-5)
Priority	High
Description and Rationale	When integrating resources that are used by multiple tenants/users (e.g. cloud environment), those shared resources shall support tenant separation / process isolation.
Tool(s) / Solution(s)	All project components will contribute towards this specification

4.2.20.2 Interface Specifications

Operation Name	*separateTenants*
Operation ID	GENERAL-S-60.1

Description	Explicit mechanisms should be set up that ensures that each tenant's resources are isolated, even if they run on shared infrastructure.
Depended / Related Services	None
Input	Multiple Tenants
Output	Guarantee of Tenant Separation

Operation Name	*isolateProcesses*
Operation ID	GENERAL-S-60.2
Description	Explicit mechanisms should be set up that limit the communication between processes so that one process cannot directly modify the executing code or memory of another process.
Depended / Related Services	None
Input	Multiple Processes
Output	Guarantee of Process Isolation

4.3 Use Case Specifications

4.3.1 Secret Detection

4.3.1.1 Specification Overview

Service Contract Name	**Secret Detection**
Specification ID	SM-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-SM1 (UR-C-F-16), LZR-SM2 (UR-C-F-17), LZR-SM3 (UR-C-F-18), LZR-SM4 (UR-C-F-19)
Priority	High

Description and Rationale	The LAZARUS system should be able to reliably identify various forms of secrets in the provided source code.
Tool(s) / Solution(s)	Secret Detection Tools

4.3.1.2 Interface Specifications

Operation Name	*scanForSecrets*
Operation ID	SM-F-10.1
Description	Provided source code is scanned for hardcoded, unencrypted or stored secrets.
Depended / Related Services	None
Input	Source Code[1]
Output	Scanning Report

[1] Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**

Operation Name	*scanCommitHistory*
Operation ID	SM-F-10.2
Description	Commit history is scanned for inadvertent secrets.
Depended / Related Services	None
Input	Source Code History[1]
Output	Scanning Report

[1] Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**

Operation Name	*configureSecretScan*
Operation ID	SM-F-10.3
Description	The user configures scanning options/parameters according to their specific needs.
Depended / Related Services	None
Input	User chooses specific scanning options/parameters
Output	Scanning options/parameters are applied to subsequent scans

4.3.2 Vulnerability Detection in Source Code

4.3.2.1 Specification Overview

Service Contract Name	**Vulnerability Detection in Source Code**
Specification ID	CL-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-CL1 (UR-CE-F-1), LZR-CL2 (UR-C-F-20), LZR-CL3 (UR-C-F-21), LZR-CL4 (UR-C-F-22), LZR-CL5 (UR-CE-F-2), LZR-CL6 (UR-CE-F-3), LZR-CL7 (UR-C-F-23)
Priority	High
Description and Rationale	The LAZARUS system should be able to reliably detect potential vulnerabilities in provided source code.
Tool(s) / Solution(s)	Code Linting Tools

4.3.2.2 Interface Specifications

Operation Name	*scanForErrors*
Operation ID	CL-F-10.1
Description	Provided source code is scanned for errors and indicates in the resulting report whether they can lead to security vulnerabilities.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Scanning Report

Operation Name	*scanForStylingIssues*

Operation ID	CL-F-10.2
Description	Provided source code is scanned for formatting or styling issues.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Scanning Report

Operation Name	*suggestCodingBestPractices*
Operation ID	CL-F-10.3
Description	Provided source code is scanned, and the relevant best practices are suggested subsequently in a report format, as well as best practice tips for software and hardware deployment.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Best Practices and Suggestions Report

Operation Name	*simulatePatterns*
Operation ID	CL-F-10.4
Description	Pattern-based simulation is applied to provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Simulation Output Report

Operation Name	*provideQualityMetrics*
Operation ID	CL-F-10.5
Description	Quality and complexity metrics are calculated based on provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Quality and Complexity Metrics Report

Operation Name	*checkCodeForStandardsCompliance*
Operation ID	CL-F-10.6
Description	Provided source code is checked for compliance with the selected safety and security-focused coding standards.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**), Selection of coding standards by user
Output	Compliance Report

Operation Name	*certifyCode*
Operation ID	CL-F-10.7
Description	Out-of-the-box certification test is applied to provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**), Certification selection by user
Output	Certification Report

4.3.3 Static Code Analysis

4.3.3.1 Specification Overview

Service Contract Name	**Static Code Analysis**
Specification ID	SA-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-SA1 (UR-CE-F-4), LZR-SA2 (UR-CE-F-5), LZR-SA3 (UR-C-F-24), LZR-SA4 (UR-C-F-25), LZR-SA5 (UR-C-F-26)
Priority	High
Description and Rationale	The LAZARUS system must be able to offer various effective static code analysis techniques.

Tool(s) / Solution(s)	Static Code Analysis Tools

4.3.3.2 Interface Specifications

Operation Name	*applyDataFlowAnalysis*
Operation ID	SA-F-10.1
Description	Apply data flow analysis to the provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Data Flow Analysis Report

Operation Name	*createCFG*
Operation ID	SA-F-10.2
Description	Create Control Flow Graphs (CFG) based on provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	CFG Report

Operation Name	*applyTaintAnalysis*
Operation ID	SA-F-10.3
Description	Apply taint analysis to provided source code.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Taint Analysis Report

Operation Name	*applyLexicalAnalysis*
Operation ID	SA-F-10.4
Description	Apply lexical analysis to provided source code.

Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Lexical Analysis Report

Operation Name	*checkForCryptographyRelatedIssues*
Operation ID	SA-F-10.5
Description	Check for misused cryptographic functions, encryption/decryption modes, and Initialization Vector selection/handling.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Scanning Report

4.3.4 SQL Injection Vulnerability Detection

4.3.4.1 Specification Overview

Service Contract Name	**SQL Injection Vulnerability Detection**
Specification ID	SI-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-SI1 (UR-CE-F-6), LZR-SI2 (UR-CE-F-7), LZR-SI3 (UR-CE-F-8)
Priority	High
Description and Rationale	The LAZARUS system must offer penetration testing services to detect possible injections at the API level.
Tool(s) / Solution(s)	Penetration Testing Tools

4.3.4.2 Interface Specifications

Operation Name	*checkForSQLInjections*
Operation ID	SI-F-10.1
Description	Apply penetration testing to detect possible SQL injections.
Depended / Related Services	None
Input	User provides target API(s) or form(s)
Output	Penetration Testing Report

Operation Name	*checkQueryInput*
Operation ID	SI-F-10.2
Description	Detect whether the provided source code whitelists query input validation and escapes all supplied query input.
Depended / Related Services	None
Input	Source Code (Decrypted from stored encrypted copy as described in Service **GENERAL-S-50**)
Output	Scanning Report

4.3.5 Fuzzing

4.3.5.1 Specification Overview

Service Contract Name	**Fuzzing**
Specification ID	FZ-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-FZ1 (UR-CE-F-9), LZR-FZ2 (UR-C-F-27), LZR-FZ3 (UR-C-F-28)

Priority	High
Description and Rationale	LAZARUS should provide reliable fuzzing services that are fully configurable, with the options to specify target(s), fuzzer(s), test cases, credentials, input types and combination logic.
Tool(s) / Solution(s)	Fuzzing Tools

4.3.5.2 Interface Specifications

Operation Name	*fuzzTarget*
Operation ID	FZ-F-10.1
Description	Apply fuzzing to selected target(s).
Depended / Related Services	None
Input	User specifies target(s)
Output	Fuzz Testing Report

Operation Name	*applyProtocolFuzzing*
Operation ID	FZ-F-10.2
Description	Apply protocol fuzzing to selected target(s).
Depended / Related Services	None
Input	User specifies target(s)
Output	Fuzz Testing Report

Operation Name	*applyFileFormatFuzzing*
Operation ID	FZ-F-10.3
Description	Apply file format fuzzing to selected target(s).
Depended / Related Services	None
Input	User specifies target(s)
Output	Fuzz Testing Report

Operation Name	*configureFuzzing*
Operation ID	FZ-F-10.4
Description	User configures fuzzing options by specifying target(s), fuzzer(s), test cases, credentials, input types and combination logic.
Depended / Related Services	None
Input	User selects fuzzing options
Output	Fuzzing options are applied

4.3.6 CVE Scan

4.3.6.1 Specification Overview

Service Contract Name	**CVE Scan**
Specification ID	CV-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-CV1 (UR-CE-F-10), LZR-CV2 (UR-CE-F-11), LZR-CV3 (UR-CE-F-12), LZR-CV4 (UR-C-F-29), LZR-CV5 (UR-C-F-30)
Priority	High
Description and Rationale	The LAZARUS system must be able to reliably detect outdated, end-of-life (EOL) and vulnerable components based on a provided Software Bill of Materials (SBOM).
Tool(s) / Solution(s)	CVE Scanning Tools

4.3.6.2 Interface Specifications

Operation Name	*ScanForCVEs*
Operation ID	CV-F-10.1
Description	Detect outdated, end-of-life (EOL) and vulnerable components based on a provided Software Bill of Materials (SBOM). Compatible with all widely used SBOM file formats (SPDX, CycloneDX, SWID, NPM package lock, Maven POM, etc.).
Depended / Related Services	None
Input	Software Bill of Materials (SBOM)
Output	Scanning Report

Operation Name	*ScanForDependencies*
Operation ID	CV-F-10.2
Description	Detect unnecessary direct and transitive dependencies based on a provided Software Bill of Materials (SBOM).
Depended / Related Services	None
Input	Software Bill of Materials (SBOM)
Output	Scanning Report

Operation Name	*checkProjectDependenciesHealthStatus*
Operation ID	CV-F-10.3
Description	Assess the health of project dependencies based on a provided Software Bill of Materials (SBOM)
Depended / Related Services	None
Input	Software Bill of Materials (SBOM)
Output	Scanning Report

Operation Name	*configureCVEScan*
Operation ID	CV-F-10.4
Description	User configures CVE scanning by providing a policy list, such as

	- Restrictions on component age - Restrictions on outdated and EOL/EOS components - Prohibition of components with known vulnerabilities - Restrictions on public repository usage - Restrictions on acceptable licenses - Component update requirements - Deny list of prohibited components and versions - Acceptable community contribution guidelines
Depended / Related Services	None
Input	User provides policy list
Output	Policies are applied to subsequent CVE scans

4.3.7 Container Vulnerability Scan

4.3.7.1 Specification Overview

Service Contract Name	**Container Vulnerability Scan**
Specification ID	CS-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-CS1 (UR-CE-F-13), LZR-CS2 (UR-C-F-31), LZR-CS3 (UR-C-F-32)
Priority	High
Description and Rationale	The LAZARUS system must be able to reliably detect insecure containers (outdated libraries, incorrectly configured containers, outdated operating system) based on a provided container image.
Tool(s) / Solution(s)	Container Vulnerability Scanning Tools

4.3.7.2 Interface Specifications

Operation Name	*ScanContainers*
Operation ID	CS-F-10.1
Description	Detect insecure containers (outdated libraries, incorrectly configured containers, outdated operating system) based on a provided container image or configuration YAML file.
Depended / Related Services	None
Input	Container Image(s) or configuration YAML file(s)
Output	Scanning Report

Operation Name	*ScanContainersForComplianceValidations*
Operation ID	CS-F-10.2
Description	Detect possible compliance validations based on a provided container image.
Depended / Related Services	None
Input	Container Image(s)
Output	Scanning Report

Operation Name	*provideContainerRelatedSuggestions*
Operation ID	CS-F-10.3
Description	Suggest best practices based on a provided container image.
Depended / Related Services	None
Input	Container Image(s)
Output	Best Practices Report

4.3.8 Detection of Network Attacks

4.3.8.1 Specification Overview

Service Contract Name	**Detection of Network Attacks**
Specification ID	NA-F-10
Specification Type	Functional (F)
Category / Grouping	Use Case Services
Dependencies	None
Stakeholders	Software Engineers, Quality Assurance Professionals, Security Analysts, Security Experts and Managers
User Requirement(s)	LZR-NA1 (UR-CE-F-14), LZR-NA2 (UR-CE-F-15), LZR-NA3 (UR-C-F-33)
Priority	High
Description and Rationale	The LAZARUS system must be able to reliably detect vulnerabilities in tested IDS and/or networks.
Tool(s) / Solution(s)	Network Scanning Tools

4.3.8.2 Interface Specifications

Operation Name	*ScanNetwork*
Operation ID	NA-F-10.1
Description	Detect vulnerabilities in tested IDS and/or networks.
Depended / Related Services	None
Input	User specifies target(s)
Output	Scanning Report

Operation Name	*suggestBestPracticesBasedOnNetwork*

Operation ID	NA-F-10.2
Description	Suggest Incident Response and Recovery policies, improvements and best practices based on target IDS and/or networks.
Depended / Related Services	None
Input	User specifies target(s)
Output	Suggestions Report

5 Sub-system Integration & Operational Specifications

Even though the finalized reference architecture will be presented in D2.4, this section presents a high-level view of the sub-system integration approach that will be followed, along with the estimated operational specifications.

5.1 Integration

With interoperability and language-agnostic implementation in mind, REST APIs and a microservice architectural style will be used to integrate the various LAZARUS sub-systems into a unified platform. More specifically, it is envisioned that an Orchestrator component will take on the major task of forwarding requests from the platform to the service modules, while a separate Interoperability Module will resolve these requests efficiently, calling upon the appropriate tools to satisfy them and return the results to the service modules. In this way, the NxN relation among use cases and the underlying tools will be managed efficiently and seamlessly.

5.2 Operational Specifications

As LAZARUS aims to provide a fully integrated DevSecOps environment, it has been decided that the entire ecosystem will be hosted in a single virtual machine while all the underlying components and modules will be deployed inside separate containers. After multiple discussions and research, the estimated operational specifications for LAZARUS are the following:

	Estimated Resources
Hosting Environment	1 VM
Processing Power	8 vCPUs
GPU	NVIDIA GPU with 12GB VRAM
Memory	16 GB RAM
Storage	1TB SSD

6 Conclusions

This deliverable provides the System Requirements for the components to be developed in LAZARUS project. System requirements are meant to detail the software system's functions, services, and operational constraints. The resulting system requirements document D2.3 (also called a functional specification) defines exactly what is to be implemented. The readers of the system requirements need to know precisely what the system will do because they are concerned with how it will support the business processes or because they are involved in the system implementation.

All user requirements and project goals have been analysed in several iterations with users and partners to ensure that the system specifications accurately reflect the services that must be provided. In total, 28 requirements have been identified and split as follows:

- Per category
  - 20 general
  - 8 related to use cases

- Per priority
  - 24 High
  - 4 Medium

The report describes the methodology used for specifying the system requirements, their identification and prioritisation, as well as their connection to the upcoming deliverable D2.4.

The final set of LAZARUS system requirements presented in the report will be the basis for LAZARUS' architecture design (D2.4) and development phases (WP4), although it is an iterative process that will be reviewed during the development phase.

7 References

[1] *Robertson, J., & Robertson, S. (2012). Mastering the Requirements Process. 3rd Edition, Addison-Wesley Professional.*

[2] *Sommerville, I. (2016). Software Engineering. 10th Edition, Pearson Education Limited, Boston.*

[3] *Robertson, J., & Robertson, S. (2017). Volere Requirements Specification Template. 18th Edition.*