UPPSALA
UNIVERSITET

UPTEC STS 26021
Examensarbete 30 hp
Juni 2026

Wiki for Knowledge Formation

A Systems Engineering Approach for Developing an Information Management System

Lisa Blidby och Emma Göransson

Civilingenjörsprogrammet i System i teknik och samhälle

UPPSALA
UNIVERSITET

Wiki for Knowledge Formation

Lisa Blidby och Emma Göransson

Abstract

This thesis explores how Systems Engineering methods can be used to develop a collaborative information management system, with the aim to facilitate knowledge formation within transdisciplinary teams in an organisational context. The international Systems Engineering standard ISO/IEC/IEE 12207 was selected as methodological framework and chosen technological processes of the system life cycle were applied throughout the project. Applied processes included definition of stakeholder needs and requirements, architecture and design definition for the system, realisation of the system, user testing and verification and validation of the system. The information management system was developed by configuring and adapting the open-source software MediaWiki. The resulting system included a module for hierarchically structuring stored information and the ability to navigate, search, categorise and create new pages. Findings indicated that the developed system could serve as an effective solution for information management. However, trade-offs needed to be made for the system to be both collaborative and secure.

Teknisk-naturvetenskapliga fakulteten

Uppsala universitet, Utgivningsort Uppsala

Handledare: Gabriel Nordin Ämnesgranskare: Ramiz Gindullin

Examinator: Elísabet Andrésdóttir

Populärvetenskaplig Sammanfattning

I många organisationer är kunskap en av de viktigaste tillgångarna. Det gäller särskilt i kunskapsintensiva organisationer, så som konsultföretag, där det huvudsakliga erbjudandet inte är en fysisk produkt utan istället medarbetarnas expertis och problemlösningsförmåga. En utmaning blir därför att ta vara på denna kunskap på ett sätt som är strukturerat och ökar tillgängligheten. Detta är speciellt viktigt i komplexa projekt där personer med olika kompetenser och erfarenheter samarbetar.

Detta examensarbete har undersökt hur ett digitalt verktyg kan utvecklas för att stötta informationshantering och kunskapsdelning i tvärvetenskapliga team. Verktyget skapades genom att använda och anpassa open-source mjukvaran MediaWiki, vilket är samma teknik som används av Wikipedia. Samarbetspartnern för examensarbetet var AFRY, vilket är ett stort konsultföretag med fokus på industri, infrastruktur och energi.

Arbetet genomfördes med hjälp av den tvärvetenskapliga ingenjörsmetodiken kallad Systems Engineering, där den internationella standarden ISO/IEC/IEEE 12207 användes som huvudsakligt ramverk. Det praktiska arbetet inkluderade processer från initiala faser i systemets livscykel, från definiering av behov och kravställning, till arkitektur- och designdefinition, implementering och konfigurering av mjukvaran, vidare till testning och utvärdering.

Resultatet av arbetet var en digital plattform där information kan organiseras på ett tydligt sätt, anpassat efter identifierade behov och krav. Användare kan bland annat navigera mellan sidor, söka efter information, strukturera innehåll i kategorier och själva bidra genom att skapa och redigera sidor. Examensarbetet visar att ett sådant Wiki-baserat system har potential att fungera som ett effektivt verktyg för informationshantering och kunskapsdelning, särskilt i tvärvetenskapliga projektgrupper där många kompetenser möts. Samtidigt visar resultatet att det finns viktiga avvägningar att göra i hänsyn till utformningen av systemet. För att systemet ska uppfylla sitt syfte av att underlätta samarbetet behöver systemet bygga på öppenhet, men samtidigt behöver regler och kontroller implementeras för att säkerställa säker hantering av viktig information. Balansen mellan tillgänglighet och säkerhet blir därför en central del av systemets utformning.

Förord

Detta arbete markerar slutet på våra studier på Civilingenjörsprogrammet System i Teknik och Samhälle med inriktning IT-system. Vi är otroligt tacksamma för allt vi lärt oss och alla härliga människor vi lärt känna under vår studietid. Det har varit en väldigt rolig tid i våra liv!

Tack till vår ämnesgranskare Ramiz Gindullin på Uppsala universitet och tack till Veronica Johansson på AFRY för ert engagemang. Tack till alla som ställt upp på intervjuer och användartester under projektets gång.

Slutligen vill vi rikta ett stort tack till vår fantastiska handledare Gabriel Nordin, som med stor entusiasm handlett och stöttat oss i detta arbete.

Nu ska vi ut i arbetslivet och fortsätta lära oss en massa spännande saker!

Lisa Blidby och Emma Göransson

Table of Contents

1. Introduction

Knowledge is widely recognised as a fundamental organisational asset. When internal knowledge, such as the expertise and experiences of the employees, is combined with an ability to efficiently use and build this knowledge base, a strategic advantage is created for the organisation (Atwood, 2009). Effectively managing information flows is therefore of great importance, especially in complex projects relying on specialist knowledge from experts in various disciplines (Li et al., 2022). This is particularly important for knowledge-intensive organisations, such as advisory firms, as their primary offering is not a tangible product but rather their expertise and problem-solving capabilities.

Research has shown that implementing a central platform for information sharing across trans-disciplinary teams, results in knowledge formation (Li et al., 2022). Knowledge formation presents as new ideas, innovative solutions, new processes, and new procedures, creating value and improving project performance (Li et al., 2022). Furthermore, by allowing employees to collaborate and contribute to the knowledge base, viewpoints and input from different sources can be collected. It has also been proven to improve the engagement amongst employees in regards of knowledge creation (Atwood, 2009). The way in which an information-sharing platform, or often referred to as an information management system, should be organised to fulfil its purpose is however highly dependent on the context and the organisation to which it belongs (Jiao et al., 2022). Furthermore, while the internal dissemination of information and knowledge is in the strategic interest of an organisation, it is also important to implement controls to protect these valuable information assets from unauthorised access (Nieles et al., 2017). In other words, organisations are confronted with the challenging task of achieving the optimal trade-off between openness, sharing information to generate new knowledge and control, to protect the valuable information assets they already possess.

This thesis project explores how the open-source software MediaWiki can be used as an information management system for transdisciplinary project teams at the advisory firm AFRY. To create a system that facilitates knowledge formation while balancing the need for controls and security, a Systems Engineering approach was used to structure the processes in thesis project. Systems Engineering was used since its purpose is to enable successful realisation of a system while optimising the design of the system, by balancing and making trade-offs between competing stakeholder objectives (International Council on Systems Engineering [INCOSE], 2023). To ensure implementation of best practices, international standards for Systems Engineering have been used as the guiding framework in this thesis project. In combination, theory and methods linked to computer security and information architecture has been used to guide the design of the system.

1.1 Partner Organisation and Case Context

The partner organisation for this thesis project is AFRY, an international engineering, design and advisory company, employing around 19.000 people. AFRY focuses on sustainability and digitalisation, with expertise within the fields of infrastructure, industry and energy (AFRY, n.d.b). AFRY was formed 2019 when the two companies ÅF of Sweden and Pöyry of Finland merged. The roots of AFRY does however go back to 1895 when Ångpanneföreningen (The Southern Swedish Steam Generator Association) was founded, which later changed name to ÅF (AFRY, n.d.b).

Prior to this thesis project, AFRY had identified an initial need for a collaborative information management tool, in the format of a web application. This need was identified within project teams using Systems Engineering, a method that includes several complex processes and places high demands on documentation. Furthermore, many of the project teams at AFRY consist of project members from different disciplines, so-called transdisciplinary teams. The idea was for the web application to serve as a common platform where project members could find relevant information, collaborate on building a shared knowledge base, and develop a common understanding of project-related information,

1.2 Aim and Objectives

The aim of this thesis project is to investigate how a collaborative information management system can be developed for transdisciplinary teams working in complex projects. This by using a System Engineering approach and configuring the open-source software MediaWiki. This aim is pursued through the following objectives:

1. Use System Engineering methods and international standards within the field as a framework, to define the needs and requirements for a collaborative information management system.

2. Use the MediaWiki open-source software to develop a suitable information management system based on the defined needs and requirements.

3. To apply the relevant processes in the chosen Systems Engineering framework and evaluate their suitability.

4. To evaluate the suitability of the open-source software MediaWiki for the information management system, with the guidance of established principles, such as information architecture and computer security.

1.3 Delimitations

Since the initial identified need from AFRY was a collaborative web-application, the solution space for the information management system was limited to solutions that meet this need. Furthermore, the development of the information management system has been limited to utilising and configuring the open-source software MediaWiki.

Some of the projects at AFRY, which have been used as cases in this thesis project, have certain security restrictions to follow which has limited the level of detail in the information we have been allowed to take part in. As a result, the focus has been on understanding the needs for the information management system developed in this thesis project and specific details regarding current systems used in the studied projects or at AFRY at large, have been left out.

2. Background

This chapter presents relevant background information beginning with a section addressing information management and Wikis, which is a type of information management system. This is followed by background about the practise called information architecture, which focuses on how information can be structured in systems to for instance increase findability. Next, the background on how to create secure computer systems is discussed. Lastly, this chapter concludes with background about the practise of Systems Engineering, which serves as the framework for this thesis project.

2.1 Information Management and Wiki

To understand the importance of information management, one needs to understand how data, information and knowledge are interconnected. Data consists of factual elements, such as measurements and statistics. When data is logically structured to support analysis, information is formed. Lastly, knowledge is created when information is interpreted and internalised by humans (Herald, Berkemeyer & Lawson, 2004).

Information management refers to the process of controlling how information is generated, maintained, secured, and made available within a system. Information exists in many different forms and are within organisations often called information assets. This can for instance include files, specifications and models of different types such as software models and system models. Effective information management provides authorised employees with accessible and well-structured information (INCOSE, 2023). Information which, according to Herald, Berkemeyer and Lawson (2004), can be used to form knowledge within the organisation.

As described in the case context for this thesis project, the initial need expressed from the partner organisation was for a collaborative information management system. Another word for this type of information management system with focus on collaboration is a Wiki. The definition of a Wiki is: “A Wiki is web-based software that allows all viewers of a page to change the content by editing the page online in a browser.” (Ebersbach, 2008, p.12). The name Wiki comes from the Hawaiian word wikiwiki which means hurry or quick and refers to the quick and uncomplicated manner a Wiki is supposed to operate in (Ebersbach, 2008). A Wiki allows users to add and edit information directly in the system, which enables the collection of viewpoints and input from different sources (Atwood, 2009). A well-known example of a Wiki is Wikipedia. The use of Wikis in organisations have, according to Atwood (2009), been proven to improve employee engagement in regards of the creation of knowledge and it has expanded the collaboration in the organisation. Organisations can host several different Wikis suited for different purposes, such as topic-based or for specific departments (Atwood, 2009).

2.2 Information Architecture

A practice, part of the broader practice of user experience (UX), is called information architecture. The term information architecture is according to Gabriel-Petit (2025) defined as "a design practice focusing on defining the structure of digital information spaces [...] by organising information and supporting findability and usability through well-designed labelling, navigation, and search systems" (Gabriel-Petit, 2025 Chapter 1, para.1).

The term information architecture does however go back to the 1970's, when it was coined by the architect and author Richard Saul Wurman in 1975 (Gabriel-Petit, 2025). Wurman described that the term 'information design', commonly used at this time, brought attention primarily to aesthetics. Instead, Wurman thought the attention should be focused on systemic design that would produce not primarily aesthetically pleasing systems, but systems that worked and performed. Wurman thought 'architecture' would be a clearer word that put systemic design in focus. Hence, he introduced the term 'information architecture' (Knemeyer, 2004). However, information architecture was first defined as a practice in 1998 by Lou Rosenfelt and Peter Morville in their seminal work called *Information Architecture for the World Wide Web* (Gabriel-Petit, 2025). Rosenfelt & Morville, coloured by their backgrounds in library and information science, focused on organisation, clear labelled, navigation and search of primarily digital information when describing information architecture. Their work laid the ground for information architecture as a practice and strongly influenced how information architecture is defined today (Gabriel-Petit, 2025).

Designed information architectures in systems come with several benefits and creates value, both for the users and the business (Gabriel-Petit, 2025). Some examples for the users are improvement of the user experience, frustration linked to not finding information is reduced, search and browsing are facilitated, and information is easier to manage, among others. From the business perspective it saves time, increases employee productivity and improves decision-making (Gabriel-Petit, 2025).

The practice of information architecture includes six key components, according to (Gabriel-Petit, 2025):

- **Taxonomies –** A taxonomy provide systematic classification of content and supports organisation into a hierarchy of pre-defined but evolving categories of information (Gabriel-Petit, 2025).

- **Metadata** – Metadata describes other data and aids in identifying and finding different instances of information (Gabriel-Petit, 2025).

- **Labelling systems** – An effective system for how to label data helps users search for and find the needed information. This can be achieved by implementing

domain-specific language that is used to label information consistently (Gabriel-Petit, 2025).

- **Organisational structure –** Structural patterns provide coherent and navigable structure, and familiar organisational schemes enable logical and comprehensible organisation of information (Gabriel-Petit, 2025).

- **Navigation systems –** Navigation systems build upon the foundational work done in creating the information architecture and makes information findable and usable to users (Gabriel-Petit, 2025).

- **Search systems –** Search systems as well as navigation systems utilise the effort put into other aspects of the information architecture to support findability and usability (Gabriel-Petit, 2025).

2.3 Secure Information Systems

The National Institute of Standards and Technology (NIST) highlights the importance of implementing controls to protect an organisation's information assets, as it does not only help keep the information itself secure but in extension helps protect the organisations profitability, reputation, and legal position (Nieles et al., 2017). Information security is defined as "The protection of information and information systems from unauthorised access, use, disclosure, disruption, modification, or destruction in order to ensure confidentiality, integrity, and availability" (Nieles et al., 2017, p.2). Confidentiality, integrity and availability constitute the three fundamental security properties, collectively referred to as the CIA-triad. Confidentiality is defined as the ability of a system to ensure that an asset is viewed only by authorised parties. Integrity is defined as the ability of a system to ensure that an asset is modified only by authorised parties, and availability the ability of a system to ensure that an asset can be used by any authorised parties (Pfleeger et al., 2024).

Applications like a Wiki can be considered an information system in its own right, but can also be considered part of a larger information system encompassing all other information systems or systems supporting information handling within the organisation. Security controls implemented in an organisation at large affect requirements on security controls of specific applications (Nieles et al., 2017). As it is policy at AFRY not to publicly share information regarding organisational security infrastructure, the focus of this thesis will not be implementing comprehensive security. Instead, the security portion of this thesis will investigate whether the application supports future implementation of fundamental security controls at the application level.

The Open Worldwide Application Project (OWASP) provides recommendations on implementation of security controls to protect against web application attacks through

their *Application Security Verification Standard* (ASVS). ASVS is widely used for assessing the security of web applications (Wen & Kat, 2023) and the baseline level one ASVS requirements will be used in this thesis, see Appendix A for a full list of the ASVS requirements. Common vulnerabilities susceptible to attacks include unauthorised access, insecure communication, exposure of sensitive information, and injection attacks. To control these vulnerabilities, baseline web application security such as authorisation, authentication, secure communication, and input validation is recommended (OWASP, 2025a).

2.3.1 Authorisation and Access Control

Authorisation decisions are governed by the system's security policy which through access control rules specifies which accesses are authorised, meaning which subjects are authorised to access what objects in what way. Controlling access is a useful tool for enforcing both integrity and confidentiality. Access control can be implemented with different levels of granularity. A common approach is Role-Based Access Control (RBAC), where permissions are assigned to roles associated with groups of subjects sharing similar needs and responsibilities, simplifying management of access control (Pfleeger et al., 2024). It should be documented which accesses should be authorised and checks should be enforced on functionality as well as data access (OWSAP, 2025). A common way to represent access control rules are access control matrixes where rows represent subjects, columns represent objects and each cell specifies the given subjects allowed access modes on the given object as visualised in Table 1 (Pfleeger et al., 2024).

Table 1: An example of an Access Control Matrix.

	Object X	**Object Y**	**Functionality Z**
Subject A	**Modify Delete Read Write**	**Modify Read Write**	**Read**
Subject B	**Write**	**Read**	**Read Write**
Subject C	**Read**	**Read Write**	

2.3.2 Authentication and Session Management

To accurately assess whether a subject should be authorised and granted access, a computer system first needs to determine if the subject requesting authorisation is who they claim to be. Thus, authentication, proving the identity of a subject, becomes necessary for access control to be effective (Pfleeger et al., 2024). Authentication mechanisms are based on something a subject knows, is or has. Knowledge-based

methods, such as passwords or passphrases are most widely used due to their low cost and simplicity but may be vulnerable to guessing and dictionary attacks if weak passwords are chosen. The length of passwords affects their strength (Pfleeger et al., 2024) therefore it is recommended that passwords contain at least 8 characters. To reduce the risk of unauthorised access, modern systems are recommended to implement multifactor authentication (MFA) combining two or more authentication factors. Other authentication methods include biometric authentication based on physical characteristics and token-based authentication, relying on physical or digital objects (Pfleeger et al., 2024).

Additionally, rate limiting and account lockout is recommended as these implementations limit allowed sign in attempts per minute and locks accounts if too many failed attempts have been recorded (OWASP, 2025a). Requiring users to authenticate themselves by signing in every time an action is performed in the web application is not feasible. Sessions are designed to solve this by either assigning each user a random unguessable number called a session identifier, using session cookies or web tokens, to identify users after they have signed in (McDonald, 2024). Session tokens need to be validated before used and securely managed to prevent theft, forgery and unauthorised use (OWASP, 2025a).

2.3.3 Cryptography and Secure Communication

Cryptography is used to protect the confidentiality and integrity of information or data during storage and transmission. Encryption encodes readable data (plaintext) into a form that is difficult to interpret (ciphertext) using cryptographic algorithms and keys, making it an effective tool to preserve confidentiality of data. Cryptography also uses hash functions to protect data. A hash function is applied to data to turn it into a hash which is a short fixed-size value where even a small change in the data results in a completely different hash. This makes hash functions useful to help preserve integrity of data by preventing attackers from modifying data without it being detected even when the hash function itself is publicly known (Pfleeger et al., 2024). Secure communication protocols such as TLS 1.2 and 1.3 should be implemented for communications between client browsers and server as they use cryptographic techniques to protect integrity and confidentiality of data during transmission (OWASP, 2025a).

2.3.4 Input Handling and Validation

Software systems are exposed to security threats whenever they process externally provided input. If input is not handled properly, the application becomes vulnerable to injection attacks, which means that user input may be treated as code and can lead to malicious code being executed on the server. Injection attacks can come in many forms and have a wide range of and serious consequences such as bypassed authentication, stolen data and disrupted system functionality (McDonald, 2024). Inputs should therefore be validated to be of expected format and ranges before they are processed to ensure they

are not interpreted as executable code or commands. Recommended controls include server-side validation, safe handling of database queries and secure handling of uploaded files (OWASP, 2025a).

2.4 Systems Engineering and the System Life Cycle

Systems Engineering is defined as a "transdisciplinary and integrative approach to enable successful realisation, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods" (INCOSE, 2019, p. 315) An *engineered system* is defined by INCOSE (2019, p. 312) "as a system designed or adapted to interact with an anticipated operational environment to achieve one or more intended purposes while complying with applicable constraints". For the purpose of this thesis, the term "system" will hereafter refer to an engineered system (INCOSE, 2023, pp. 1-3).

The purpose of Systems Engineering is to enable successful realisation of the system of interest (SoI), while optimising the design of the SoI by balancing and making trade-offs between competing stakeholder objectives (INCOSE, 2023). The definition of the system of interest (SoI) is "the system whose life cycle is under consideration" (INCOSE, 2023, p. 314). One way in which the realisation is managed is by breaking down the processes in stages and through so called *decision gates* that evaluate whether desirable characteristics of the SoI are fulfilled and if risks are acceptable before deciding if the SoI is ready to enter other stages (INCOSE, 2023, p. 25). The set of stages a system goes through from conception to retirement are termed a s*ystem life cycle* (INCOSE, 2023, p. 314), and the stages a system life cycle is broken down into are called *system life cycle stages.* See Figure 1. The stages of a system life cycle can be repeated as many times as needed. Stages are often not sequential and can occur concurrently with other stages (INCOSE, 2023, pp. 25-26).

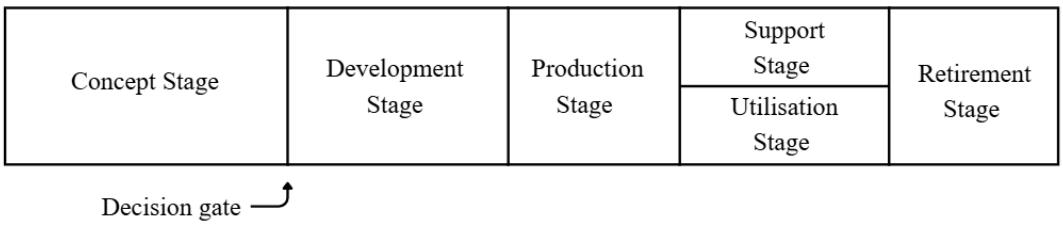

Figure 1: The generic life cycle stages of a system. The vertical lines separating all stages represent decision gates.

The typical life cycle stages of a SoI include concept, development, production, utilisation, support, and retirement. The Concept stage defines the problem at hand,

explores possible solutions, evaluates feasibility and risks, identifies stakeholder needs and requirements, as well as a preliminary architecture of the SoI. This stage is considered an especially critical part of the system life cycle. Decisions made in the Concept stage severely affect and limit the later stages of the life cycle and as the project progresses through the stages, it becomes increasingly difficult to change direction. The Development stage transforms stakeholder needs and requirements into an engineering baseline consisting of system requirements, architecture, design, documentation, and plans. The Production stage is initialised by an approval allowing the engineering baseline to be turned into an actual system with associated documentation that will be passed on for use in subsequent stages (INCOSE, 2023, pp. 26-29).

The Utilisation stage and the Support stage are implemented concurrently. The Utilisation stage is initialised with the transition of a system from production to use and ends as the system enters the Retirement stage. The duration of the Utilisation stage typically lasts longer than the other life cycle stages and its main tasks include maintenance of documentation in addition to product modifications to remedy deficiencies, enhance capabilities, or extend the life of the system. In the Support stage deficiencies and failures are recorded and used as basis for reparation or modification of the system. This stage ends when the Retirement stage is entered, or a decision is made that the system should no longer be supported. Finally, the system is taken out of operation in the Retirement stage. The focus in this stage is mainly to ensure that the disposal requirements set up during the concept and Development stages are satisfied before the Retirement stage is completed (INCOSE, 2023, pp. 26-29).

One tool used in Systems Engineering to help manage systems across their life cycle stages is *system life cycle processes.* System life cycle processes are a series of activities which in turn consist of a series of tasks. The activities and tasks of the respective processes are applied to a systems life cycle stages to achieve one or more outcomes for a stated purpose (INCOSE, 2023, pp. 39-40). Applying the system life cycle processes with concurrency, iteration, and recursion helps ensure communication between the processes which in turn helps ensure a good system definition that effectively and efficiently meets the stakeholder needs and requirements (INCOSE, 2023, pp.42-44).

3. Process framework: ISO/IEC/IEEE 15288 and 12207

The International Organisation for Standardization (ISO) and the International Electrotechnical Commission (IEC) form the specialised group for worldwide standardisation. The technical committees of ISO and IEC consist of national bodies of the organisations, and they collaborate in fields where there is a mutual interest. The Institute of Electrical and Electronics Engineers (IEEE) is another international organisation who forms international standards by a so-called consensus development process, to reach a final product based on expert opinions from different perspectives. The three organisations have collaborated to together form the standard ISO/IEC/IEE 15288, with the title *Systems and software engineering – System life cycle processes* (International Organisation for Standardization et al. [ISO et al.], 2023). This standard will be called ISO 15288 in this thesis.

The purpose of ISO 15288 is to provide a common framework for the processes in all parts of the life cycle of systems, using a Systems Engineering approach. The standard can be applied for systems which include system elements such as hardware, software, materials, humans, data, among others (ISO et al., 2023). To further tailor the standard specifically for software systems, the ISO/IEC/IEE 12207 standard with the title *Systems and software engineering - Software life cycle processes*, which will be called ISO 12207 in this thesis, was created. This standard includes the same life cycle processes as ISO 15288 (Fairley, 2019, pp. 99-101) but the activities are more adapted to software systems. The two are in other words closely related and can either be used alone or in combination with each other (International Organisation for Standardization et al. [ISO et al.], 2017). The choice of which standard to follow for the software life cycle processes should be based on the characteristics of the SoI (ISO et al., 2017). As stated in ISO 15288, when the predominant SoI is software, ISO 12207 should be followed (ISO et al., 2023). Therefore, ISO 12207 has been applied in this thesis project. The system life cycle processes supporting the system of interest through its life cycle are divided into the same four categories in both ISO 15288 and ISO 12207. The four categories are Agreement, Organisational Project-Enabling, Technical Management, and Technical processes, see Figure 2 (ISO et al., 2023) (ISO et al., 2017) (INCOSE; 2023).

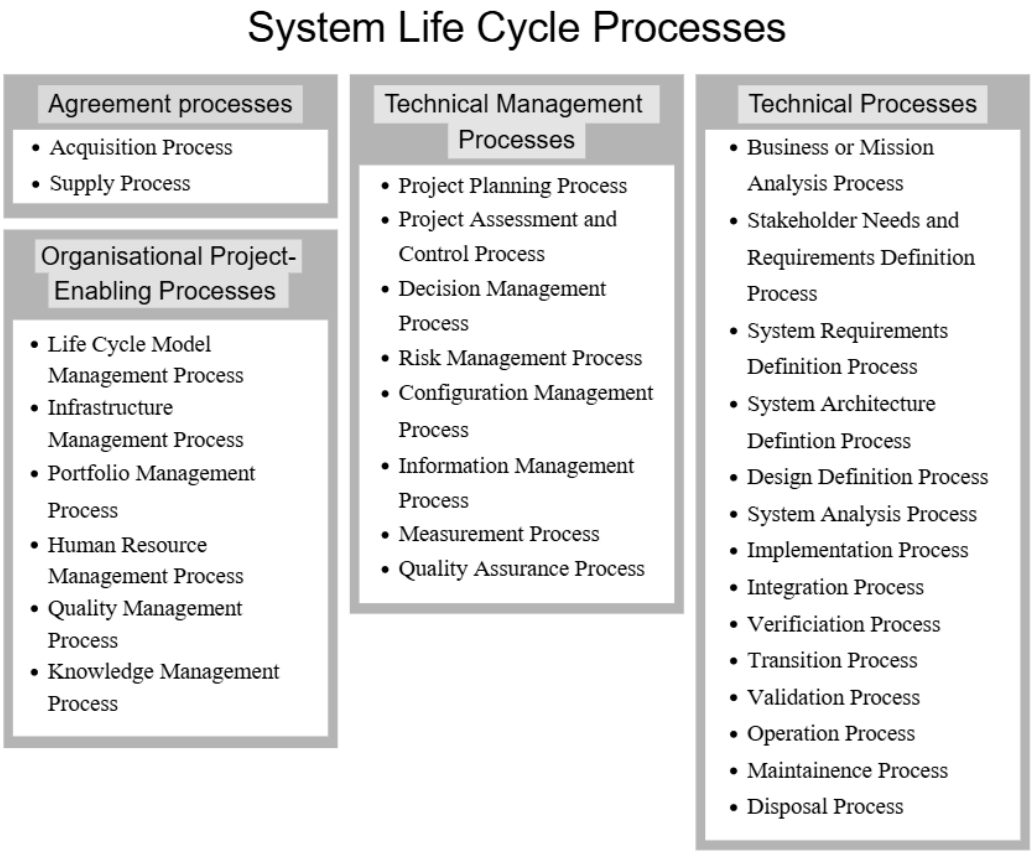

Figure 2: System life cycle processes included in ISO 15288 and ISO 12207.

3.1 Agreement Processes

The Agreement processes are initialised before a project starts or in the very beginning of a project, where one organisation is acquiring products or services from another organisation that supplies them (ISO et al., 2017). The Agreement processes include the Acquisition process and the Supply process. The objective for these two processes is to ensure the product or service meets the requirements and to define the conditions for this relationship. In cases where the acquiring and supplying parties are both part of the same organisation, the Agreement processes can be used with less formality (ISO et al., 2017).

3.2 Organisational Project-Enabling Processes

The Organisational Project-Enabling Processes are typically concerned with the management and improvement of the organisation's activities at a strategic level (ISO et al., 2017). However, ISO, IEC, and IEEE (2017) do not focus on capabilities intended to enable general business management objectives but instead focuses on the capabilities of an organisation relevant to supporting the system throughout its system life cycle. The

aim of the Organisational Project-Enabling Processes is to provide projects within the organisation with the resources the projects need to meet the needs and expectations of the organisation's stakeholders. The Organisational Project-Enabling Process establishes the environment in which projects are carried out (ISO et al., 2017).

3.3 Technical Management Processes

The Technical Management processes manage recourses and assets allocated by organisation management as well as managing application of these resources and assets to projects, aiming to fulfil the agreements the organisation has entered. Technical Management processes concern the responsibilities of the Systems Engineer, which in a high degree overlap with the responsibilities of a Project manager, but focuses in particular on the technical aspect of the project. Technical Management processes ensure the technical aspects are properly planned, monitored, controlled, and corrected if needed, as well as handles management of information throughout the project (ISO et al., 2017).

3.4 Technical Processes

The Technical processes are followed to create a product or service from the needs of stakeholders. The processes can be used throughout the entire life cycle of the technology, starting with concept definition, to gathering and establishing stakeholder requirements, to product development, the implementation and usage and finally the retirement of the technology (ISO et al., 2017). According to ISO (2017), the usage of the activities defined within the Technical processes reduce the risks from technical actions and decisions during the project. The activities are used to ensure the software system meets a wide range of qualities, such as functionality, maintainability, and usability (ISO et al., 2017).

While agreement, organisational, and Technical Management processes have important supporting roles in projects and throughout the life cycle of a system of interest, the Technical processes have a more practical role. Through a series of transformational actions across the system life cycle, the Technical processes convert inputs from one Technical process into outputs which are in turn used as input in the following Technical process. The application of these Technical processes eventually result in a system of interest that addresses stakeholder concerns (INCOSE, 2023, pp. 101-103). It is important to understand that systems do not necessarily move through their life cycles in a linear manner, life cycle stages can occur with concurrency and iteration, similarly life cycle processes can be applied concurrently, recursively and iteratively (INCOSE, 2023, pp. 33-35 & pp. 102-103). Here, the Technical processes will be described linearly and mapped to the generic life cycle stages. This is done to capture the main idea of how Technical processes can be applied to take the system of interest through its life cycle stages.

3.4.1 Concept Stage

During the Concept stage of the system life cycle mainly two Technical processes are applied, the Business or Mission Analysis process and the Stakeholder Needs and Requirements Definition process (INCOSE, 2023, pp. 26-29). The Business or Mission Analysis process aims to clearly define the business or mission problem or opportunity, outline the range of possible solutions, and identify suitable categories of solutions that potentially could address the problem or take advantage of the opportunity. The purpose of the Stakeholder Needs and Requirements Definition process is to establish what stakeholders require from a system to be able to deliver the capabilities needed in its intended environment. To fulfil this purpose, relevant stakeholders are identified, their needs are elicited and then analysed and translated into stakeholder requirements that expresses how the system is intended to interact with the environment it is intended to operate in. The stakeholder requirements will then serve as a basis for validating that the SoI delivers the expected capabilities (ISO et al., 2017).

3.4.2 Development Stage

Primarily, the System Requirements Definition, Architecture Definition, and Design Definition processes are implemented during the Development stage (INCOSE, 2023, pp. 26-29)**.** By performing the System Requirements Definition process, the intention is that the stakeholder needs and requirements are transformed into system requirements. The resulting system requirements should be measurable requirements that specify what characteristics, attributes and functionality the system should have and how it should perform to fulfil the stakeholder requirements. The aim of the Architecture Definition process is to develop alternative system architectures and select one or more options that address stakeholder concerns and satisfy system requirements and express the chosen architecture(s) through a consistent set of views. Notably, the Architecture Definition process can be conducted at different levels of detail depending on the degree of detail needed to support relevant decisions, preferably the architecture should be as design-agnostic as possible to allow more flexibility in design choices. While the design of a system will likely change over time, the architecture will stay constant and outputs resulting from the Architecture Definition process are used across the life cycle processes. In particular, the architectural entities defined through the Architecture Definition process will be considered in the Design Definition process. The purpose of the Design Definition process is to provide a more detailed description of the system and its elements to enable implementation of the system, while remaining consistent with the architectural entities (ISO et al., 2017).

The System Analysis process, like all life cycle processes, can be employed at any stage of a systems life cycle and is applied to develop inputs required for technical assessments and can help ensure confidence in system requirements, architecture and design. However, due to cost and schedule constraints, the System Analysis process is typically only performed for critical characteristics. It includes a wide range of analytical methods

such as mathematical analysis, modelling, simulation, experimentation amongst other techniques aiming to analyse Technical performance, system behaviour, feasibility and other factors relevant for the technical assessment at hand. The System Analysis process can be conducted with varying degree of formality and rigor depending on how much ingoing data or information there is to base the analysis on, how critical the information resulting from the analysis is to the assessment in question, the size of the project and the time constraints (ISO et al., 2017).

3.4.3 Production Stage

The Implementation and Integration processes are predominantly used during the Production stage (INCOSE, 2023, pp. 26-29). The purpose of the Implementation process is to transform requirements, architecture and design, including interfaces, to create a specified system element adhering to those by using the chosen implementation technology and relevant technical specialties or disciplines. The Implementation process is usually performed concurrently with the Integration process. While the Implementation process creates system elements, the Integration process assembles the implemented system elements as well as identifies interfaces to enabling systems and integrates these with the system of interest. To summarise, the Integration process strives to combine the system elements into a complete system that satisfies the system requirements, architecture, and design (ISO et al., 2017).

The purpose of the Verification process is to provide objective evidence that a system or system element meets its requirements and fulfils the specified characteristics, that "the product is built right" (ISO et al., 2017, p. 82). This process identifies anomalies and supplies the information needed to analyse and resolve the identified anomalies. Verification can be performed throughout all Technical processes. The Verification process is usually carried out at decision gates and other key points in a system's life cycle to confirm that the requirements have been met or that process activities have been performed or process outcomes have been achieved. For software systems, this process is typically initialised for three purposes. First, to verify that a software work product, such as a software entity, element or documentation, or the service that this work product supplies, correctly reflects its specified requirements. Second, to verify that the integrated software elements correctly meet their defined requirements. Third, to verify that each system requirement has been implemented and tested so it can be established that the software system is ready for delivery (ISO et al., 2017).

To support the system in progressing from the Production stage to the Utilisation stage, the Transition process is often applied. The Transition process intends to support this progression by preparing the system so that it can deliver the services required by stakeholders in the operational environment. The transition introduces a verified system into operation in a planned and controlled way while ensuring that the system is functional, stable, and compatible with other systems in the operational environment. Furthermore, the Transition process can be deployed at any stage to complete the criteria

required to exit that stage. When the SoI is primarily a software product the Transition process is often applied to support the deployment of the software system to different environments, for example through the transition from a development environment to a test or maintenance environment (ISO et al., 2017).

The purpose of the Validation process is to provide objective evidence that the system, when used in its intended operational environment, fulfils its business or mission objectives and stakeholder requirements (ISO et al., 2017). While the Verification process determines that "the product is built right" (ISO et al., 2017, p. 82), the Validation process determines that the "right product is built" (ISO et al., 2017, p. 89). The goal of the Validation process is to build confidence that a system or system element can be used as intended in its operational environment by having stakeholders confirm that it fulfils its purpose. Furthermore, the Validation process identifies any anomalies and provides the information needed to resolve these by reinitiating the Technical processes where the anomaly was created. The Validation process is typically applied at decision gates and other key points in a systems life cycle to ensure that requirements for stakeholder intended operational use have been fulfilled. For software systems, mainly two purposes of the Validation process are presented. First, validating that the requirements for a specific intended use of the software work product, such as system entities or elements, are met. Second, to gain the confidence of stakeholders, especially acquirers or customers, that the delivered software system meets stakeholder requirements and the system is fit for use (ISO et al., 2017).

3.4.4 Utilisation Stage

The most dominant process active during the Utilisation stage is the Operation process (INCOSE, 2023, pp. 26-29). The main goal of the Operation process is to utilise the system to deliver its services. This process defines the requirements for operating the system, assigns personnel to operate the system, and monitors the system and operator-system performance. In addition, the Operation process identifies and analyses operational issues in relation to agreements, stakeholder requirements and organisational constraints to ensure that the system supplies its services without unintended interruptions (ISO et al., 2017).

3.4.5 Support Stage

During the Support stage of the system life cycle, the primarily active process is the Maintenance process (INCOSE, 2023, pp. 26-29), which aims to enable the system to ensure that the system continues to provide its intended services. This is done by monitoring the system's ability to deliver these services in addition to recording and analysing incidents. Thereafter, this process implements appropriate corrective, adaptive, perfective, and preventative actions to restore the systems' capabilities and follows up with confirming that the actions have had the desired effect (ISO et al., 2017).

3.4.6 Retirement Stage

The Disposal process is active primarily during the Retirement stage of a systems life cycle (INCOSE, 2023, pp. 26-29). The purpose of the Disposal process is to end the existence of a system or system element for its intended purpose, ensure that replaced elements are handled properly, and address critical disposal requirements in line with agreements, Organisational policies, legal, safety, and security considerations. This process involves deactivating, disassembling, and removing the system or any of its elements from use as well as addressing any waste product arising in any of the life cycle stages. The waste is handled, stored, recycled, or disposed of in an environmentally responsible manner, in accordance with legislation, agreements, organisational constraints, and stakeholder requirements. In addition, the process ensures that unstable or obsolete elements do not re-enter the supply chain and, where required, maintains records to support monitoring of user health, operator safety, and environmental impact (ISO et al., 2017).

3.5 Tailoring

In most cases, all aspects of the Systems Engineering processes and life cycle models cannot be applied to a project without adaptation. The process of making these adaptions is called tailoring, and can include removing, modifying, or adding activities and tasks. The purpose is to tailorise in a way that scales the processes to a level of rigor that ensures that the needs of the project are met within an acceptable level of risk (INCOSE, 2023, pp. 215-218). Tailoring is in many aspects focus on balancing the degree of formality in the Systems Engineering processes and the risk of overrunning schedule and cost within the project. The goal is to conduct the optimal amount of tailoring that minimises both the risks and costs (INCOSE, 2023, pp. 215-218).

4. Method

This chapter outlines the methodological framework used in the thesis, based on a Systems Engineering approach guided by the ISO 12207 standard. The standard has been complemented with additional methodologies when needed. The chapter begins with presenting how the standard has been tailored for this project. This is followed by a presentation of the performed processes, including eliciting requirements, defining the architecture and design, realising the system, and performing user-testing. These parts of the chapter are divided into the performed life cycle stages in this thesis project: concept, development, and production.

4.1 Usage and Tailoring of ISO 12207

The methodology used to structure the Technical processes and related activities has been based on ISO 12207. As prescribed by ISO 12207, the standard has been tailored in accordance with the guidelines provided by ISO 12207 to better suit the project at hand (ISO et al., 2017). A general overview of the most significant changes made during tailoring and the motivations for these changes will be provided in this section.

The Organisational Project-Enabling processes aim to provide the projects within the organisations with recourses needed for the projects to be successful (ISO et al., 2017). As the Organisational Project-Enabling processes are conducted at an organisational level rather than a project level, they were excluded through the tailoring process. Likewise, the Agreement processes were eliminated through tailoring since ISO 12207 (2017) stipulates that the Agreement processes are taken care of by the organisation, and its main activities and tasks are applied outside of a project's lifetime. The Technical Management processes are on the other hand meant to be applied within a project's lifetime and Technical Management processes are often invoked through the application of Technical processes. These processes were conducted internally in this thesis project, but less formal documentation of the performed processes were conducted since this would significantly increase the risk of overrunning schedule.

The tailoring process proceeded by looking at each process within the Technical processes. The first major change in the Technical processes resulting from the tailoring process was the removal of the Business or Mission Analysis process. The purpose of the Business or Mission Analysis process is to define the business or mission problem or opportunity and characterise the solution space. Thereafter, the Business or Mission Analysis process aims to identify potential solution classes that could take advantage of a previously defined opportunity or solve a previously identified problem (ISO et al., 2017). The partner organisation AFRY had already identified the problem this thesis aims to solve and characterised the solution space in terms of open-source software, see section 1.1 Partner organisation and case context. Thus, the Business or Mission Analysis process

had already been conducted. Therefore, this process was eliminated when tailoring ISO 12207. Additional processes that were eliminated through tailoring included the Transition process, Maintenance process, and Disposal process. These processes depend heavily on information about the operating environment which the system of interest is to be placed in. Since detailed information about the operating environment was not available and could not be taken into consideration the Life Cycle processes part of the Transition, Maintenance and Disposal stages were removed through the tailoring process.

No additional processes were removed in their entirety. However, certain activities and tasks of the remaining processes were excluded through tailoring, and some were implemented with less rigor than others. The tailoring process eliminated most activities and tasks regarding planning and acquisition of enabling systems needed to conduct the later activities and tasks within that same process. This was done as the task of supplying projects with the needed resources is part of the Organisational Project-Enabling processes, and that responsibility lays on the organisation rather than the project (ISO et.al., 2017). In addition, activities and tasks concerning management were generally conducted in a less formal manner, as is recommended by the standard in projects with lower complexity (ISO et al., 2017). Instead of invoking the formal Technical Management processes of ISO 12207, management of artefacts and information was typically conducted informally and recorded through less formal documentation. In addition, relevant information and representations resulting from the Technical processes have been compiled in this thesis.

4.2 Concept Stage

This section presents the methods and processes used in the Concept stage for the system, which consisted of identifying stakeholders, conducting interviews with stakeholders to understand the needs for the system, defining these needs and finally transforming them into stakeholder requirements.

4.2.1 Stakeholder Needs and Requirement Definition Process

4.2.1.1 Identifying Stakeholders

A stakeholder is defined as "an individual or organisation having a right, share, claim or interest in a system or its possession of characteristics that meet their needs and expectations" (ISO et.al., 2017, p. 9). A method for outlining different classes of stakeholders is the power-interest matrix, see Figure 3. This grid has two axes: interest on the X-axis and power on the Y-axis, which forms four quadrants. In the first quadrant, the class with high interest and high power is placed, which are the key stakeholders that should be engaged closely throughout the creation of the system. In the second quadrant, the class with low interest and high power is placed. This class should be kept satisfied by meeting their needs during the project. In the third quadrant, the class with low interest and low power is placed, which is the least important group of stakeholders, and therefore minimal effort should be given to this class. Lastly, the fourth quadrant is the class with high interest, low power which is a class that should be shown consideration by being kept informed during the project (Maqbool et.al., 2022).

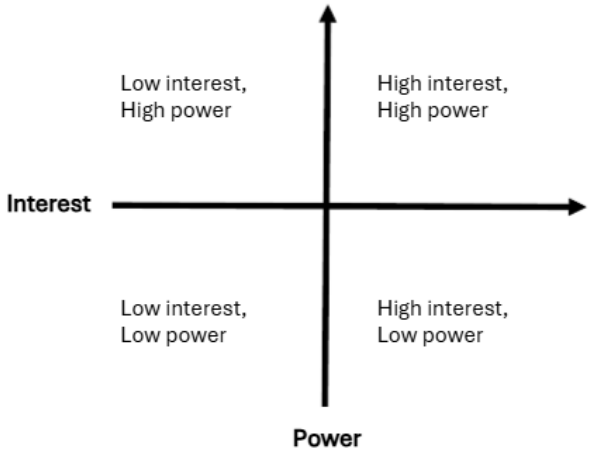

Figure 3: The Power-Interest Matrix.

Since this thesis project was small scaled, the process of identifying stakeholders was kept to a level of rigor suitable for the project. Due to this, a delimitation was to keep the process of identifying stakeholders internally within the partner organisation AFRY. The process of identifying stakeholders consisted of brainstorming sessions, with the guidance of the supervisor due to his insights of the organisation and colleagues. When stakeholders were identified, they were mapped in the power-interest matrix for guidance on how to work with different stakeholders.

4.2.1.2 Interviews

The chosen method to understand the stakeholder needs for the information management system was interviews. This since interviews are found to be a good practice to get an overall understanding of the work of the stakeholders (Sommerville, 2011, pp. 104-105) as well as being an easy-to-use technique to extract system-level requirements from the stakeholders (Laplante & Kassab, 2022). According to Wall and Stokes (2014), there are three different forms of interviews: structured, unstructured, and semi-structured. In the structured interview format, the questions are clearly written in preparation, and the interviewer strictly follows these questions. The unstructured format can, on the other hand, be seen as a conversation and is essentially the opposite of the structured format. The conversation is initialised by questions relating to the research theme and if needed, further questions can be used to steer the conversation in the anticipated direction (Wall & Stokes, 2014). In this project, the semi-structured format was chosen, which according to Wall and stokes (2014) is a combination of the structured and unstructured interview formats.

To understand the needs of identified stakeholders, five semi-structured interviews were conducted. Prior to the interviews with the identified stakeholders, an interview guide was constructed along with interview questions categorised into three topics: background about their role and current project, the current way of working with information and documents and the future information system. Several fixed questions were decided in preparation, however interesting findings during the interviews were followed up with spontaneous questions. Furthermore, the focus was open questions which are favourable when the objective with the interview is collecting qualitative opinions, since the interviewee is encouraged to elaborate on their responses in contrast to a closed question (Wall & Stokes, 2014). See Appendix B for the interview guide. The interview guide and the questions were constructed in alignment with the practical advice on semi-structed interviews from the methodology book *Interview Techniques for UX practitioners: a user-centered design method* (Wilson, 2014).

Semi-structured interviews typically benefit from being conducted in two-person teams, with one being responsible for notetaking and the recording equipment, and the other being responsible for asking questions (Wilson, 2014). Due to this, Lisa Blidby took notes and handled the recording equipment, and Emma Göransson asked the interview questions. Interview 1 was not recorded per request of the interviewee, instead extensive notes were taken during the interview. Interviews 2-5 were recorded and transcribed, in

addition to the detailed notes that were taken during the interviews. An overview of the interviews conducted can be found in Table 2.

Table 2: Conducted interviews with stakeholders.

Index	Role	Date	Duration
Interview 1	Project leader, project A	2026.02.02	65 min
Interview 2	Project member, project A	2026.02.03	30 min
Interview 3	Project member, project A	2026.02.06	60 min
Interview 4	Group leader, project C	2026.02.24	75 min
Interview 5	Project member, project B	2026.02.26	60 min

4.2.1.3 Requirements

According to Laplante and Kassab (2022), one of the challenges with requirement engineering is to understand what requirements really are. This since they can vary from high-level descriptions to detailed specifications, depending on the level of abstraction the stakeholder expresses their needs (Laplante & Kassab, 2022). The definition of a requirement is, however, according to ISO 12207: "Statement that translates or expresses a need and its associated constraints and conditions" (ISO et al., 2017, p.7). Constraints can, for instance, refer to pre-existing technologies or budget constraints. Conditions refer to measurable qualitative or quantitative attributes that are added to the requirement, to make it possible to validate and verify it. It is, however, important to not add bounds to the requirements if not strictly needed, since this will limit the possible solutions (ISO et al., 2017).

In ISO 12207, requirements are divided into two categories: stakeholder requirements and system requirements. Stakeholder requirements are user-oriented and specify what functionalities and capabilities the system should provide to fulfil their needs (ISO, IEC & IEEE, 2017; Laplante & Kassab, 2022). An example is: "Users shall be able to search in the system". The method for system requirements will be presented in the section 4.3.1. System Requirements Definition process.

Both categories of requirements can be divided into different types, where the most common are functional and non-functional. Functional requirements describe the functionality the system should have (Laplante & Kassab, 2022). The type of non-functional requirements is often used as an umbrella term for all other requirements that do not correspond to a specific functionality. These types of requirements can, for instance, define quality measures for the system, such as usability, security, and availability (Laplante & Kassab, 2022).

4.2.1.4 Defining the Stakeholder Needs and Requirements

The process of defining requirements begins with the Stakeholder Needs and Requirement Definition process. When conducting this process as described in ISO 12207, the starting point is to identify stakeholders and their needs related to the software (ISO et al., 2017). The method for collecting the stakeholder needs in this thesis project was, as previously described, through interviews. After the interviews were conducted, the stakeholder needs were defined by analysing the transcript and notes from the interviews. All identified needs from each stakeholder were listed and after this, analysed to resolve any duplication which occurred if several stakeholders expressed the same needs. Furthermore, the needs were analysed with a focus on feasibility and the constraints for the project, to down-select needs that were considered not feasible. All this following ISO 12207 (2017). As part of this process, additional attributes such as rationale, critical quality characteristics, and criticality should be identified and added for traceability and clarity (ISO et al., 2017). Critical quality characteristics refer to the characteristics of the system that users consider essential, and often relate to security, reliability, and availability.

When the needs were defined, they were transformed into stakeholder requirements. For additional guidance when working with processes related to requirements in ISO 12207, the additional standard ISO/IEC/IEEE 29148, *Systems and software engineering - Life cycle processes - Requirements engineering,* was used. This standard will be called ISO 29148 in this thesis. ISO 29148 provides a standardisation of how requirements should be written and what characteristics they should have. The requirements must, for example, be singular, meaning not composed of several requirements and written in a concise manner using phrases with unambiguous meaning. Furthermore, the requirements must be verifiable, which means that it must be possible to determine whether they are met or not. This can, for instance, be done by incorporating a measurable value and the unit in the requirement (International Organisation for Standardization et al. [ISO et al.], 2018).

It is, according to INCOSE, essential to store the needs and requirements in a database, for traceability and to be able to make refinements if needed. The database should include traces between the needs, stakeholder requirements, and system requirements (INCOSE, 2023, p. 111). The chosen database for this purpose was Excel. To ensure traceability, all needs and requirements were assigned a unique ID. In the excel sheet for the stakeholder requirements, a column with the title “Derived need” was added, where the ID for the related need was mapped. Furthermore, all stakeholder requirements were categorised as functional, or non-functional and rationale and verification methods were added for each requirement.

After the Stakeholder Needs and Requirement Definition process is conducted, it is followed by the System Requirements Definition process. In this process, the user-oriented stakeholder requirements are transformed into the technical system requirements (ISO et al., 2017). See section 4.3.1. System Requirements Definition process.

4.3 Development Stage

This section presents the methods and processes used in the Development stage for the system, which consisted of transforming stakeholder requirements into system requirements and defining the architecture and design for the system.

4.3.1 System Requirements Definition Process

For background regarding requirements as a concept, see previous section 4.2.1.3 Requirements. Where the stakeholder requirements are user-oriented, the system requirements are instead focused on the technical view of the system and how the needs of the users should be met from an operational point of view (ISO et al., 2017). An example is: “When a user is searching for a string, the system shall show information objects with this string”. When transforming the stakeholder requirements into system requirements, ISO 29148 (2018) was used for additional support, as previously done when defining the stakeholder requirements.

To ensure traceability, unique ID:s were assigned to all system requirements the same way as for the stakeholder needs and requirements was. In the excel sheet with the system requirements, a column with the title “Derived Need / Stakeholder Requirement” was added, where the ID for the derived need and/or stakeholder requirement was mapped. Furthermore, all system requirements were categorised as functional, or non-functional and rationale and verification methods were added for each requirement.

When the complete initial set of stakeholder and system requirements were compiled, they were analysed alongside the supervisor for this thesis project at the partner organisation AFRY, to validate that the requirements were feasible and written in a correct manner. When this was ensured, the project moved to the next processes in the Concept stage, namely defining architecture and design.

4.3.2 Architecture Definition Process

The Architecture Definition processes begin with preparing for the Architecture Definition. Included in this step is to review relevant information and identify key drivers of the architecture as well as stakeholder concerns. As prescribed by ISO 12007, key drivers of architecture were identified by analysing organisational constraints, the operational and system context, requirements, and other factors affecting the sustainability of the system through its life cycle. i can include feasibility of implementation and integration, availability of open-source software, and principles such as replaceable components, amongst others (ISO et al., 2017). Stakeholder concerns are things of interest or importance to stakeholders that influence a system in its environment. These concerns can include influences such as technological, operational, economic, legal, ethical, and social influences (ISO et al., 2023). In the Architecture Definition process, identifying stakeholder concerns refers to those concerns that have a strong

influence on the choice of architecture (ISO et al., 2011). Stakeholder concerns were derived from needs expressed by stakeholders during held interviews as well as from the requirements derived from the processes conducted during the Concept stage. Preparing for the architecture also includes defining the evaluation criteria for the architecture based on stakeholder concerns and key drivers (ISO et al., 2017). In this thesis project architectural solutions were evaluated on how effectively they addressed identified stakeholder concerns and critical system requirements while key architectural drivers were taken into consideration.

The Architecture Definition process specifies several work products to be created to support choosing an appropriate architecture for the system of interest. The description of a systems architecture is called an *architecture description* and consists of architecture views. An *architecture view* is a work product that expresses the architecture of a system from the perspective of one or more specific concerns articulated by the stakeholders of the system. Architecture views are governed by their architecture viewpoints. An *architecture viewpoint* is a work product that defines the conventions for creating, interpreting, and analysing architecture views to address specific system concerns held by stakeholders. Conventions of a viewpoint can include notations, design rules, model kinds and modelling methods as well as techniques for analysis and other operations that can be applied to views. (ISO et al. 2011). “A viewpoint is a way of looking at systems; a view is the result of applying a viewpoint to a particular system-of-interest" (ISO et al., 2011, Appendix A, Section A.4).

Which viewpoints are selected should depend on what concerns are expressed by relevant stakeholders (ISO et al. 2011). A concern commonly held by all the system’s stakeholders is whether the system will function as intended and fulfil its purpose. A viewpoint that addresses this concern is the functional viewpoint (Rozanski & Woods, 2005). A functional viewpoint focuses on the architectural elements that realise the system’s functionality, in other words the system’s logic (Eles & Cripps, 2010, Appendix B). To represent the system architecture from the functional viewpoint, a functional structure model is recommended. A functional structure model typically includes functional elements and external entities with assigned responsibilities necessary for the system to fulfil its purpose. In addition, connectors representing how elements are related as well as interfaces stating how other elements may access the functional element in question (Rozanski & Woods, 2005).

Another common concern, typically held by system administrators, developers, testers, and communicators, is what hardware, network and third-party software is required in the deployment environment to be able to support the systems' functionality (Rozanski & Woods, 2005). If this concern is identified for the system in question, a deployment viewpoint can be applied. The deployment viewpoint describes the physical environment in which the system is intended to be deployed, including the required hardware environment (Rozanski & Woods, 2005). A deployment viewpoint of the system is especially relevant in situations where the system may be deployed into several different

environments, in which case essential characteristics of the deployment environment need to be clearly illustrated. If requirements exist around using specific third-party software, network capacity, or physical constraints, those are also addressed by the deployment viewpoint. For the deployment viewpoint, a runtime platform model, network model, and technology dependency model is recommended to be used. A technology dependency model is meant to document what software and hardware the system depends on, so the system can be correctly installed and deployed. A runtime platform model defines the required hardware nodes, potential requirements on that hardware, which hardware nodes should be connected to each other via interfaces, and which hardware nodes host which functional elements. A network model should describe the network in mind during the development of the system, including aspects such as type of network service, to document what environments the system has been adapted for (Rozanski & Woods, 2005).

If there is a particular system quality that needs to be considered across several of the systems architectural views, a so called perspective or cross-cutting viewpoint can be applied to the rest of the views (Eles & Cripps, 2010). One such cross-cutting viewpoint that is commonly applied in software systems is the security viewpoint. There is often a concern regarding whether software systems are able to keep valued assets secure (Rozanski & Woods, 2005), stakeholder groups particularly likely to hold this concern are developers, maintainers, testers and system administrators. A security viewpoint addresses this concern by describing how the system is to make sure only authorised parties are allowed to view and modify the valued, and thus sensitive, assets of the system by addressing security concerns consistently across relevant architectural views. This should be done by identifying sensitive assets, defining a security policy, highlighting threats, and identifying implementations that control threats (Rozanski & Woods, 2005). These implementations to control threats are included in the architectural description by introducing new elements in existing views, for example adding a functional element responsible for access control in the functional structure model of the functional view (Eles & Cripps, 2010).

Following the guidelines of ISO 12207, viewpoints and model kinds were selected and adapted based on identified stakeholder concerns. Models and views of candidate architectures were then developed in accordance with the conventions specified by the viewpoints. In developing models and views, ISO 12207 prescribes that the system context should be expressed in terms of interfaces and interactions with entities external to the system of interest. Architectural entities of the SoI should be identified as well as relationships between the entities that address key stakeholder concerns and critical system requirements. Also, concepts, properties, characteristics, behaviours functions or constraints that are considered important to architecture decisions of the software should be assigned to the architectural entities of the SoI (ISO et al., 2017). These conventions regarding modelling technique recommended by ISO 12207 were included in all viewpoints and applied when creating the chosen models and views.

An additional convention applied across viewpoints was the choice of boxes-and-lines diagrams as design notation for creating models and Canva as the design tool of choice for visualising the chosen models. Boxes-and-lines diagrams as modelling method allows for development and application of a standard notation suitable for the stakeholders or audience it is meant to communicate the system architecture to. Although the choice of notation is more flexible in boxes-and-lines diagrams, compared to modelling techniques with predefined notation, the standard notations chosen should still be clearly defined and consistently applied (Rozanski & Woods, 2005). This was achieved by clearly stating what each visual element represented in the figures of the resulting models. In addition to the visual elements in the resulting figures, tables with further specification of requirements and other valuable information regarding each element were noted in tables alongside the respective figure.

In accordance with ISO 12207, the models and views were harmonised with each other. Whenever there are several models created of the same system, there is a chance these models are inconsistent, and using several views to describe an architecture result in a need to maintain consistency between views. One way to record consistencies and inconsistencies between views is to set up rules for how elements in different views should relate and define the relationships between these elements. For instance, every functional element in a functional view needs to be executed on one or more hardware components as defined by the deployment view. These rules are called correspondence rules and are included in each viewpoint (ISO et al. 2011).

Since ISO 12207 recommends the architecture to be as design-agnostic as possible, to allow for more flexibility in design choices, the views constructed focused on identifying high-level system elements addressing the stakeholder concerns and satisfying the critical system requirements (ISO et al., 2017). The purpose of applying the Architecture Definition process here is producing an architecture description representing the required characteristics of the system architecture that needs to be respected when specifying the design of the system as well as producing, deploying, utilising and maintaining the system. The description is, in other words, not a complete description of the architecture of the finished system, but rather a model of which requirements the system architecture must meet. As a result of applying the Design Definition process, see chapter 4.3.3. Design Definition process, a complete description of the systems architecture is produced.

Furthermore, rather than producing several architecture candidates and then assessing their suitability, architectural alternatives were, when relevant, continuously evaluated on their ability to address stakeholder concerns and system requirements through the Architectural Definition process.

4.3.3 Design Definition Process

The purpose of the Design Definition process is to describe the system and its elements in enough detail for developers to implement the system consistently with the intended

architecture (ISO et al. 2017). Since one objective of this thesis project was to assess the suitability of MediaWiki as an information management system, the Design Definition process will focus on determining if the default MediaWiki package has elements that fulfil the responsibilities of those defined in the established system architecture. Where required functionality is not provided by the default MediaWiki package, it will be examined whether the functionality can be achieved through installation of any available MediaWiki extensions or through supporting technologies compatible with MediaWiki. Where several viable alternatives exist, these will be compared, and motivation will be presented for the choice made. The design artifact resulting from this process will consist of a specification of which extensions and additional technologies should be implemented in addition to the default MediaWiki package. It will be evaluated if the design aligns with the architecture description by mapping responsibilities specified in the architecture description to elements in the design, and it will be made clear if any responsibilities were not fulfilled by the design.

4.4 Production Stage

This section presents the methods and processes used in the Production stage for the system, which consisted of the realisation of the system and conducting user testing to verify the developed system.

4.4.1 Realisation of the Information Management System

The integration and Implementation processes defined in ISO 12207, were applied to guide the configuration and realisation of the system building on the open-source software. The Implementation process in this project focused on installing and configuring the software to meet specified requirements, rather than developing new software components. The Integration process focused on the systematic combination of the components needed to have a functioning system, such as the web application, the database and the webserver.

As an initial activity in the Implementation process, the system environment was to set up, following the official documentation from MediaWiki. This to ensure that the set-up of the system meets the technical requirements for the software to function. For the hosting service for the system, the internal IT department at AFRY was contacted and assisted by providing a virtual machine on a local server for both the MediaWiki software and the database. See section 5.3.1., Technical specifications, for the technical details of the final hosting environment.

Following the set-up of the environment, the implementation activities continued with the configuration of the software. This consisted of adjustments to the accessibility of the system, selection and installation of relevant extensions, and personalising the look of the system. The layout and visual structure of default MediaWiki were complemented with

additional elements in the user interface, aiming to fulfill previously defined requirements. As part of preparing the integrated system for user-testing, content was added using internal AFRY guides on Systems Engineering. See section 5.3.2 The final information management system for details.

4.4.2 User Testing as Verification Method

The process of verification is, as previously described in section 3.4.4.3. Production stage, the process of checking "if the system is built right" (ISO et al., 2017, p. 82) and can be done through different methods. The chosen method to verify the usability aspects of the system was user testing and the book *Usability Testing Essentials* (Barnum, 2010) was used for guidance.

As part of the planning process prior to the conducted user tests, several task-based scenarios were constructed. Task-based scenarios refer to realistic scenarios given to the participants, which include completing certain tasks within the system linked to the scenario. This can, for instance, be locating a piece of information or performing a key functionality in the system (Barnum, 2010). An example of a task-based scenario is: "You are currently part of a project working with <topic> and have been given this system as an information base from your project leader. Find information linked to <topic>". By structuring the tests around these specified task-based scenarios, which were based on the identified needs for the system, the participants interaction with the system was goal-oriented which reduced the risk of the participants using the system without intention (Barnum, 2010). Furthermore, a test plan was constructed during the planning process and was used during the test sessions for guidance. See Appendix I for the test plan, including the task-based scenarios. During the test sessions, the think aloud method was used, a method in which the participants verbalise their thoughts, expectations and decision-making process while using the system (Barnum, 2010). The participants were instructed at the beginning of the test on how to use this method and were reminded during the test when needed. After each conducted task, an open-ended question about the experience doing this task was asked to receive immediate feedback.

Before conducting the user tests, it is beneficial to conduct a pilot test to see how the test work with a real user. After the pilot test, changes can be done to the test if necessary (Barnum, 2010). When conducting a smaller study, like in this thesis project, the participant in the pilot test could be chosen as an extra participant in case this session needs to be discarded. However, if the changes are minor, the findings from the pilot test can be included (Barnum, 2010). As per recommendation of Barnum (2010) the first test session was a pilot test with a colleague at the department which was not part of the interviews, in case of the need to discard this first test. After the pilot test was conducted, minor changes to the test were done such as changing some phrasing in the task-based scenarios to make the tasks clearer. Since there were not any significant changes, the findings from the pilot test were included in the test results.

Research has shown that the optimal number of participants in a user testing is between three and five participants. This since the trade-off between findings and number of participants reaches its maximum at this point (Barnum, 2010). Due to this, there were four participants in the conducted user testing, including the pilot test, see Table 3. All participants, except for the pilot user, were stakeholders who also participated in the interviews. This way, direct feedback could be collected from this group of stakeholders.

Table 3: Conducted user testing.

Index	Participant	Date	Duration
Pilot test	Colleague at department	2026.04.23	50 min
User test 1	Project member, project A	2026.04.27	45 min
User test 2	Project member, project B	2026.05.04	40 min
User test 3	Project member, project A	2026.05.11	40 min

During the tests the roles were divided between the researchers, where Lisa Blidby acted as the test moderator and Emma Göransson took responsibility for taking notes of the participants' think-aloud comments and observations. To ensure consistency across test sessions, all participants used the same laptop during the tests. The test sessions were conducted in a private, quiet meeting room. The laptop was connected via HDMI to a large television screen, allowing the researchers to observe the participants' navigation within the system without intruding on their interaction with the laptop. In addition, for all tests, except for the pilot test, screen recordings with audio were taken to enable further analysis after the conducted tests.

After the tests were conducted, the participants were asked to fill in a post-test questionnaire. The chosen questionnaire for this study was the so-called System Usability Scale (SUS). SUS is one of the most popular standard post-test questionnaires and was developed by John Brooke at Digital Equipment Corporation in 1986 (Barnum, 2010). It consists of ten statements linked to the usability of the system, that should be rated on a scale from one to five with one representing "Strongly disagree" and five representing "Strongly agree". The odd-numbered statements have a positive tone, and the even-numbered statements have a negative tone (Sauro & Lewis, 2016). The original SUS created by Brooke, was written in English. Since the participants in the test sessions were Swedish speaking and the tests were conducted in Swedish, the SUS was translated to Swedish. Two additional statements to assess non-functional requirements were also added at the end of the questionnaire. See Appendix J for the original English version of SUS and the translated version, which was used in the test sessions. Finally, after the SUS was filled in, some open-ended questions regarding the participants overall experience of the system were asked to collect feedback regarding this.

After the test sessions, the SUS score was calculated for each participant following the instructions by Sauro and Lewis (2016). For each statement in the SUS, a contribution from 0 to 4 was calculated. For positively worded statements, this was obtained by subtracting one from the participants' chosen answer ($xi - 1$). For negatively worded statements, it was calculated by subtracting the selected scale position from five ($5 - xi$). The SUS score was then calculated by summing all individual contributions and multiplying the total by 2.5. This results in a final SUS score ranging from 0 to 100, in increments of 2.5. The higher the score, the more usable the system is considered (Sauro & Lewis, 2016). According to Sauro and Lewis (2016), the overall mean score of the SUS is 68 with a standard deviation of 12.5. This was found by using data from 446 studies and over 5000 individual SUS responses (Sauro & Lewis, 2016). The additional statements included by the end of the questionnaire were not included in the SUS score, since they were not part of the original SUS questionnaire created by Brooke.

The participants' comments and observations were analysed to identify both potential usability problems and positive aspects of the system. The following aspects, as described by Herztum (2020), were focused on when conducting this part of the analysis: Verbal and nonverbal cues from users expressing uncertainty, frustration, or other negative emotions about the system or how it performs during the defined task-based scenarios indicate usability problems. Likewise, instances where users need assistance to complete a task indicate usability problems. According to Herztum (2020), the user should not be assisted before the cause of the usability problem is identified, which was followed during the user testing. Furthermore, expressions of surprise indicate that the system does not match the expectations of users. Where a negative surprise suggests a usability problem and a positive surprise indicates a positive user experience. Additional indicators of usability problems are situations where users need to try several actions before finding an action that brings them closer to completing the task or situations where users are led off track in the system, resulting in a need to go backward to solve the task. The action of going backward in the system is called a backtrack. This is opposed to situations where the actions taken lead the user closer to task completion, which indicates good usability. Finally, the collected data was analysed to determine if users missed any information or steps resulting in an incomplete task solution or if the information in the system was clear for users to complete the tasks (Hertzum, 2020).

To analyse the overall results from the user testing, a triangulation analysis was used, which means combining findings from different data sources to enrich the analysis and understanding (Barnum, 2010). The three data sources used in the analysis, all collected from the testing sessions, were the observations, participant comments, and responses from the SUS questionnaire.

4.5 Ethical Considerations

This thesis project has been conducted in accordance with the ethical principles outlined by the Swedish Research Council (Vetenskapsrådet, 2017). To begin with, participation in both the interviews and the user testing was voluntary and the purpose of the thesis project was clearly communicated to the participants. In the interviews, it was communicated that any question could be declined and the interviewees were encouraged to respond in a level of detail that felt comfortable to them, to avoid sharing confidential information about their current projects. This to protect the integrity of the participants during the data collection. All interviewees and participants in the user-testing were anonymised. In line with recommendations for responsible handling of sensitive research material, audio recordings were deleted after transcription, while the resulting transcripts were stored securely on computers given by the partner organisation with access limited to the two researchers in this thesis project. Furthermore, to ensure confidentiality, only information approved for publication has been included in this report. Any additional information shared with us during the thesis project have been treated as confidential. Before publication, the supervisor and the manager of the acquiring department reviewed and approved the content in the report to ensure that no confidential material was disclosed.

5. Results

This chapter presents the results from the life cycle stages of the information management system addressed in this thesis, which includes concept, development, and production.

5.1 Concept Stage

This section will present the results of the processes performed in the Concept stage of the information management system, which includes the results from the interviews and the stakeholder needs and requirements.

5.1.1 Identified Stakeholders

The results of the process of identifying stakeholders and analysing these stakeholders are presented in Figure 4 below.

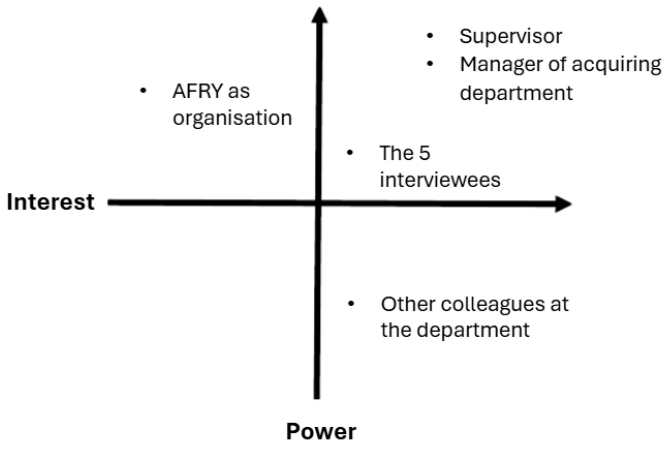

Figure 4: The identified stakeholders mapped in the Power-Interest Matrix.

Since AFRY prior to this thesis project identified a need for a collaborative information management system, some stakeholders were already identified. These were the stakeholders working in project A, which was the project where the initial need was found. To get a broader understanding of needs from different stakeholders, working in transdisciplinary projects using Systems Engineering methods, an assessment of the interest for the information management system was conducted internally with the help from the supervisor at AFRY. As a result, two more stakeholders showed interest in participating in interviews.

In the first quadrant in the matrix, the key stakeholders are placed. In the top part of the quadrant with most power and interest are the supervisor of the thesis project, and the manager of the department who are acquiring the system developed in this project. These stakeholders have been closely engaged in the progression of the project. In the same

quadrant but placed slightly lower in regards of power and interest are the five interviewees, since these are colleagues who expressed interest in participating in interviews and interest in the information management system. These stakeholders have also been engaged during the thesis project, but not to the same extent as the supervisor and the manager. In the second quadrant is AFRY as an organisation, which includes the rules and restrictions projects at AFRY need to follow. These rules exert power over the project, but the organisation itself does not have a high interest in the project. The interest in the project from the organisation has instead been assigned to the colleagues at the department where the thesis project is conducted, which is in the fourth quadrant. A final presentation was held to this group of stakeholders to inform them about the project. Since the process of identifying stakeholders was delimited to be kept internally at AFRY, there were no stakeholders identified in the third quadrant with low interest and low power.

5.1.2 Defined Stakeholder Needs and Requirements

As a result of the conducted interviews and the Stakeholder Needs and Requirements Definition process, a total of 45 explicit and implicit needs were identified. Explicit needs refer to needs the interviewee expressed directly, and the implicit needs are needs derived from the conversation in the interview as a whole. These needs were given unique ID:s from N:01 to N:45, to ensure traceability. When assigning each need a critical quality characteristic, the following 11 different characteristics were used: accuracy, availability, collaborative, correctness, customisability, efficiency, integrity, scalability, security, traceability, and usability. The identified stakeholder needs were divided into two main categories: information architecture and computer security. Needs within the information architecture category addressed aspects linked to findability, traceability, and efficiency in the system. Examples included the need to search for information, have an overview of the information structure in the system and receive updates when new information is added in the system. The computer security category addressed needs linked to access control, protecting the information in the system and meeting general security restrictions. Since the interviews were mainly focused on understanding the needs from a user's perspective, most of the needs were members of the information architecture category. This categorisation provided a structured understanding of the needs of the stakeholder for the system. Lastly, the criticality of the needs was decided, between high, medium, and low. These were assigned in relation to the purpose of the system namely being a functioning collaborative information management system. See Appendix C for the complete set of stakeholder needs.

As a result of the process of transforming the stakeholder needs into stakeholder requirements, following the methodology for writing concise requirements, 25 stakeholder requirements were defined. These requirements were assigned unique ID:s from ST:01 to ST:25. All stakeholder requirements were mapped to one or more needs which they were derived from. Two requirements were non-functional and related to the user's perception of the system's visual aspects and the perceived time required to find information. These two requirements were formulated in a way that allowed a

measurement using a Likert scale, which were part of the questionnaire given at the end of the user testing, see Appendix J. The other 23 requirements were functional and included requirements regarding what functionalities the system should have from the user's perspective. Three different verification methods were assigned to the stakeholder requirements. These were user test, functional test and inspection. The user tests focused on usability and were used to ensure that the core functionalities, such as searching, finding information, contributing by writing and uploading files, and editing existing information, worked well from a user's perspective. Functional testing was used to confirm that the system was built according to the specified requirements. Finally, inspection was used in a similar way to functional testing to verify requirements, but only in cases where execution was not required. For example, verifying that a search function exists requires inspection, but verifying that the correct string is generated as a search result requires functional testing. See Appendix D for the complete set of stakeholder requirements.

5.2 Development Stage

This section will present the results of the processes performed in the Development stage of the information management system, which includes the system requirements and the results from the architecture and Design Definition processes.

5.2.1 Defined System Requirements

As a result of the System requirement definition process, 30 system requirements were identified. They were given ID:s from SY:01 to SY:30. One requirement was non-functional, focusing on how quickly the search function should display search results. To make this measurable, a time baseline of 0.5 second was used since research shows response times of 0.5 seconds or below are not noticeable as a delay to users (Bai et al., 2018). The other system requirements were functional and formed the basis for the functionalities the system needed to fulfil its purpose and meet the needs of the stakeholder. The system requirements were mapped to one or more need and/or stakeholder requirement they were derived from. The exception from this, was the system requirement stating the system should be built on open-source software, which was a requirement derived from the delimitation from the case context. Three different verification methods were assigned to the system requirements: functional test and inspection. As discussed in 5.1.2. Defined stakeholder needs and requirements, functional testing was used to confirm that the system was built according to the specified requirements, and inspection was used in cases where execution was not required to test the requirement. See Appendix E for the complete set of system requirements.

5.2.2 Architecture Definition Process

As the first step in the Architecture Definition process key drivers were identified from both the material from the conducted interviews and a combination of the case context for this thesis project. The following were identified key drivers:

- Managing information in a way that ensures efficient navigation and a clear overview
- Collaboration amongst the users
- Separation of privileges
- Ability to adapt the system-code
- The time frame of the thesis project

Additionally, stakeholder concerns were identified as part of this process. The following were the identified stakeholder concerns:

- Security: How will the information in the system be kept securely?
- Efficiency: How will the system ensure more efficient work?
- Deployment: Will I be able to implement this system in my operational environment?

5.2.2.1 Viewpoints

Based on the identified stakeholder concerns, three separate viewpoints were selected and adapted to better align with stakeholder concerns, needs and requirements. First, a functional viewpoint was developed to handle concerns regarding system functionality and the ability of the system to fulfil its purpose. The preferred model for the functional viewpoint was chosen to be a functional structural model. Second, a deployment viewpoint was developed to address concerns pertaining to the possibility of future deployment of the system in a range of different environments. The model chosen for this viewpoint was a runtime platform model, where functional elements were mapped to hardware components to identify the essential technical characteristics the system require from its environment to uphold its intended functionality in an operational environment. Given the limited information on the future deployment environment and in the interest of keeping the Architecture Definition as design agnostic as possible, neither a network model nor a technology dependency model was included in the deployment viewpoint. More detailed descriptions of the functional and deployment viewpoints are presented in Appendices F and G, respectively.

Finally, stakeholders also expressed concern regarding whether information and other sensitive assets in the system would be kept secure. To meet this concern, a cross-cutting security viewpoint was chosen, and any identified requirements or concerns regarding security were to be addressed by this viewpoint. Due to the lack of contextual information threat modelling was deemed inappropriate. Therefore, instead of identifying sensitive assets and threats to implement suitable controls, the security viewpoint was adapted to

identify additional elements and responsibilities to append to the functional and deployment views primarily based on the security controls specified in the Application Security Verification Standard (ASVS) level 1 requirements.

5.2.2.2 Views

In accordance with the established viewpoints corresponding views were created by following the conventions specified in each viewpoint regarding what models to create and how, including how models in views should relate to models in other views, and additional information specified by viewpoints to create each view. The resulting views are represented by figures and tables in this section and make up the architectural description outlining the architectural requirements the system is expected to meet. Figure 5 below presents the functional elements, connectors, and interfaces of the functional structure model.

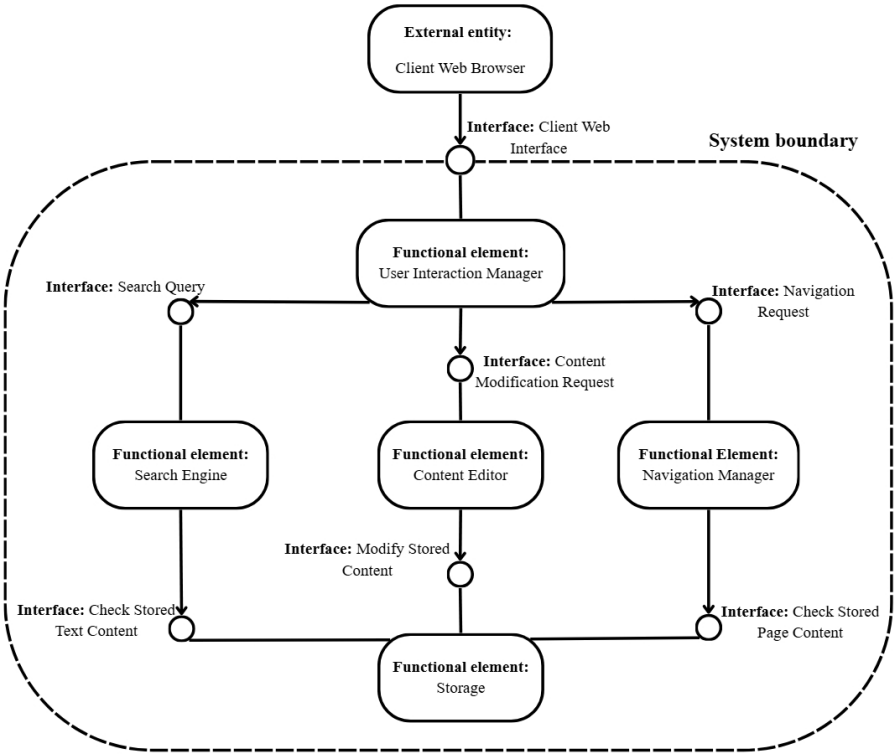

Figure 5: Functional structure model represented by boxes-and-lines diagram

Table 4 below includes information on the responsibilities of each functional element in the functional structure model as well as the system requirements that each functional element was derived from.

Table 4: Specifications of functional elements included in the functional structural model.

Name	Responsibilities	Derived from system requirements
User Interaction Manager	Interpret all user requests. Route requests to the right functional element. Return system-response to user. Provide structure for user interaction, including overview and navigation elements as well as structures content (UI-related functionality). Automatically update overview of pages when new pages are added.	SY:17, SY:23, SY:27
Search Engine	Search for and return content containing words from input string. Search for and return content containing words related to words from input string. Return content ranked after relevance. Return search results within ≤0.5s when result exists.	SY:01, SY:02, SY:03, SY:04, SY:06
Content Editor	Enable authorised users to create, edit, delete pages. Enable users to upload files. Enable authorised users to upload documents in PDF-format. Enable authorised users to upload documents in Word-format. Enable metadata to be added to pages with documents. Enable users to go back to earlier versions of pages. Enable authorised users to restore deleted pages. Enable authorised users to add pages to a category.	SY:07, SY:08, SY:09, SY:10, SY:11
Navigation Manager	Resolve navigation requests. Maintain navigation paths. Keep links to referenced pages up to date. Disable links if referenced page is deleted.	SY:16, SY:20, SY:21
Storage	Store content of current pages. Store content of deleted pages. Store content from old versions of edited pages. Store records of changes made to content.	SY:25, SY:26

Table 5 below includes information on the responsibilities of external elements in the functional structure model required for the system to function. Identified here was the need for the element User Web Browser, this element was derived from the need for users to be able to interact with the system for it to fulfil its intended purpose.

Table 5: Specifications of external entities included in the functional structural model.

Name	Responsibility
User Web Browser	Enable interaction between users and the system

Table 6 below includes information on the responsibilities of each interface in the functional structure model as well as the input triggering the communication between the elements it connects and the output it returns to the element initiating the communication.

Table 6: Specifications of interfaces and their associated connectors, included in the functional structural model.

Name	Associated Connector	Input	Output
User Web Interface	Connects User Web Browser and User Interaction Manager (interface between user and the SoI).	User-request	System-response
Search Query	Connects User Interaction Manager and Search engine.	Search query	Ranked search results
Content-edit request	Connects User Interaction Manager and Content Manager.	Content operation-query	Updated content
Navigation Request	Connects User Interaction Manager and Navigation manager	Navigation request	Page content
Check Stored Content	Connects Search Engine to Storage	Search query	Search results
Edit Storage	Connects Content editor to Storage	Content operation-query	Updated content
Check Page Content	Connects Navigation manager to Storage	Content request	Page content

Figure 6 below presents the hardware nodes and network links of the runtime platform model of the deployment view. This view makes explicit which hardware nodes must, at the least, be provided by an operational environment to enable the functional elements of the system to exercise their defined responsibilities. The hardware nodes and network links were identified by reviewing stakeholder concerns, key architectural drivers, critical system requirements, and the functional view. Identified elements included a server needed to host the functional elements of the system to enable the system to run, thus the hardware node Main Server was included. For the Client Browser to be able to operate a client node is essential. Through personal communication with AFRY the type of client node was identified to be restricted to computers, resulting in the node Client Computer. In addition, the Client Computer need to connect to the Main Server for the user to access the system, presenting a need for the network link Internet. And finally, the online storage hardware Min Storage was included, with the purpose of hosting the functional element Storage.

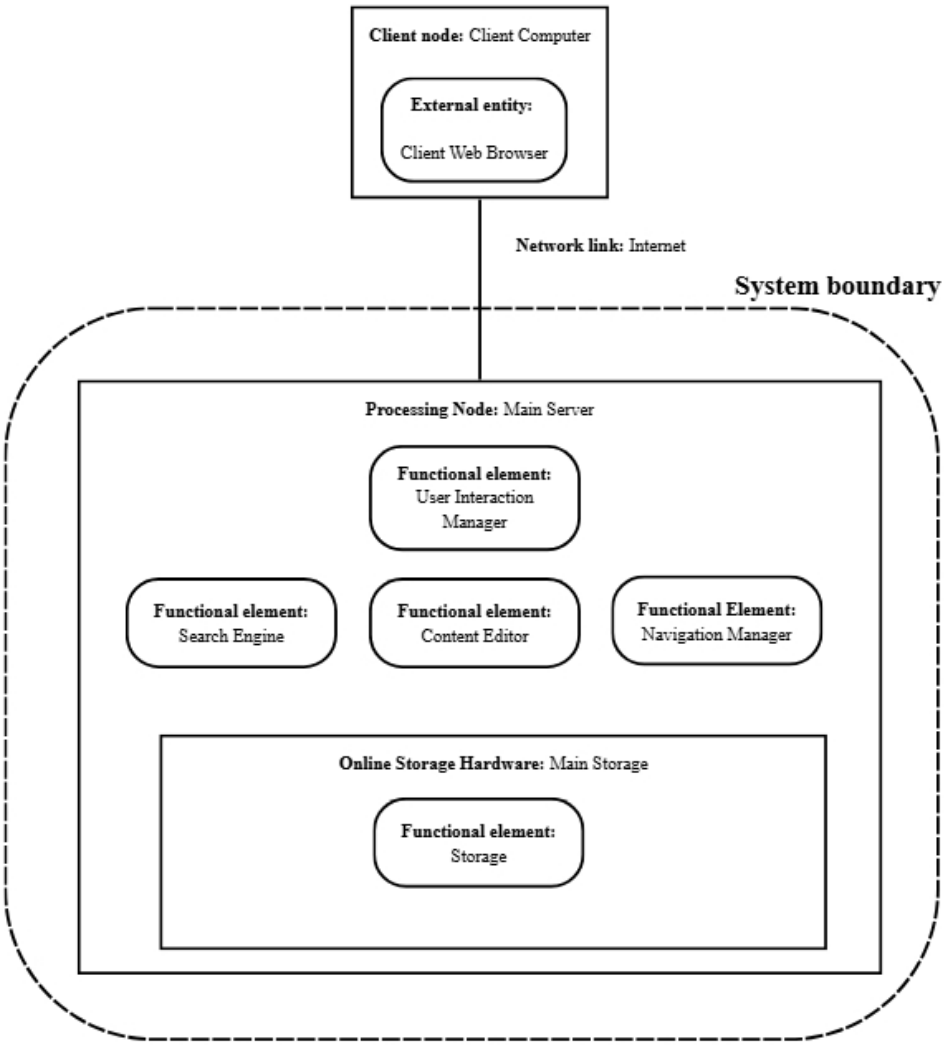

Figure 6: Runtime platform model represented by a boxes-and-lines diagram.

Table 7 below includes information on the hardware nodes of the runtime platform model, including which elements of the functional structural model it is to host and specifications of the hardware node if any were identified.

Table 7: Specifications of hardware nodes in the runtime platform model.

Name	Type	Runtime-element-to node mapping	Specifications
Client computer	Client node	Client Web Browser	Type of hardware: desktop/laptop
Main Server	Processing node	User Interaction Manager Search Engine Content Editor Navigation Manager	
Main storage	Online storage hardware	Storage	

The network links identified as part of the runtime platform model is presented in Table 8 and includes information on which hardware nodes the network link connect.

Table 8: Specifications of network connections in the runtime platform model.

Name	Type of Network Link	Connects
Internet	Public network	Main Server and Computer via Internet

Stakeholder concerns and critical system requirements as well as key architectural drivers were reviewed when applying the security viewpoint to the functional view. Included in the identified key architectural drivers was the need for separation of privileges. As the functional structure model of the functional view (see Figure 5) has no functional element responsible for access control, any anonymous user could access any assets or functionality, posing a considerable threat to the system. To counteract this, a functional element, Authorisation Manager, responsible for enforcing separation of privileges was added to the functional structure model. Reviewing the basic security requirements of the *Application Security Verification Standard* (OWASP, 2025a) introduced in the background of this thesis, two additional functional elements were identified and assigned security related responsibilities, see Figure 7.

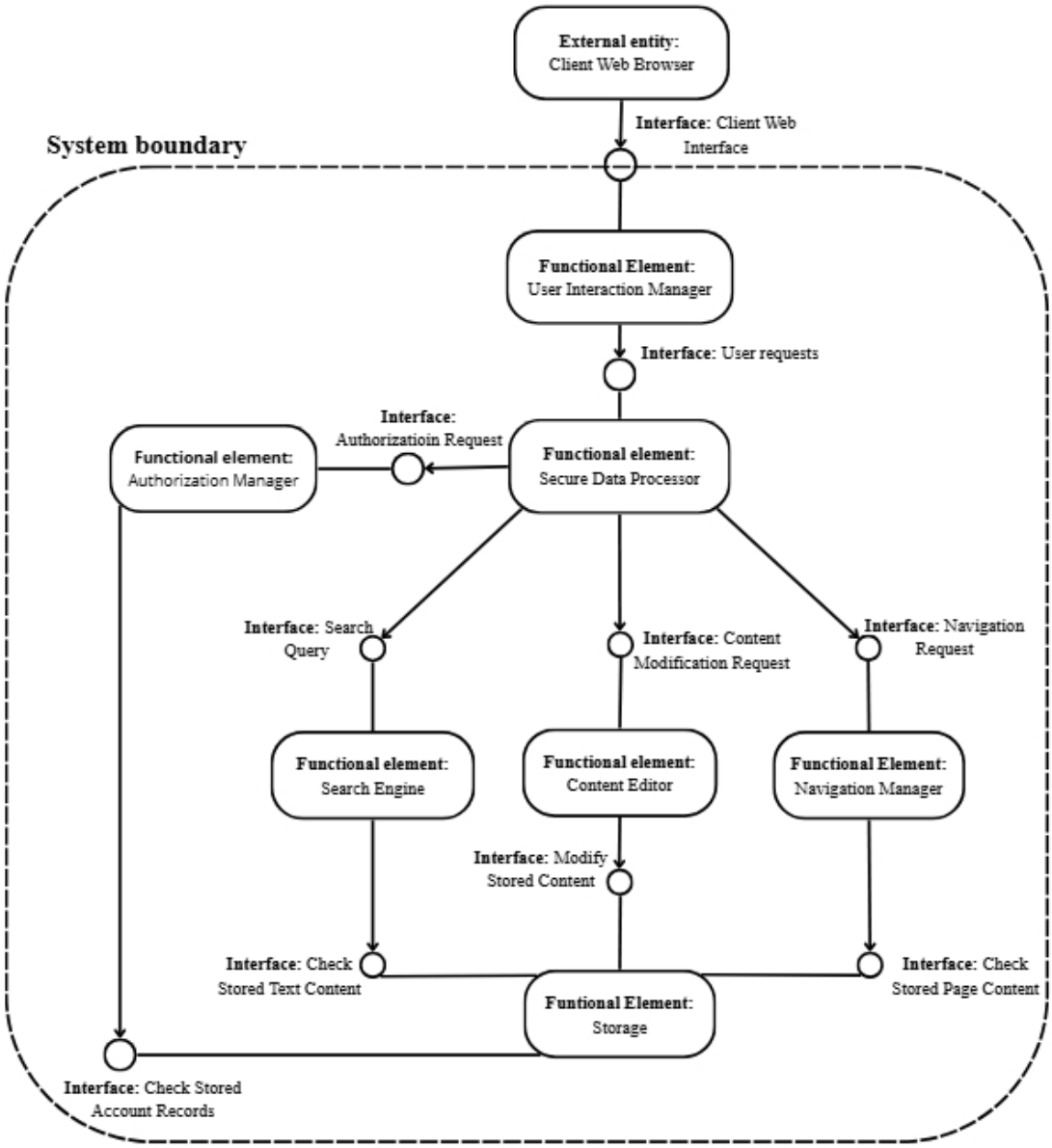

Figure 7: Functional structure model with additional functional elements identified by applying the security viewpoint to the functional view

The responsibilities of the resulting functional elements are presented in broad terms in Table 9 and referred to by their ASVS-identifier. The identifiers act as a reference to each requirements full description in Appendix A.

Table 9: Additional functional elements included in the functional structural model by applying the security viewpoint as well as and additional responsibilities of existing functional elements.

Name	Existing/New	Responsibilities	Derived from
User Interaction Manager	Existing	Responsible for maintaining secure communication, interpreting user actions and coordinating workflows.	ASVS: 2.3.1, 3.2.2, 3.3.1, 3.4.1, 3.4.2, 3.5.2, 3.5.3, 4.1.1, 4.4.1, 6.2.6, 6.2.7, 14.2.1,14.3.1
Storage	Existing	Store passwords as salted hashes	SY:29,
Authorisation Manager	New	Responsible for enforcing secure authentication by demanding certain features from passwords and other relevant parameters. Should implement multifactor authentication. Also, responsible for securely handling passwords, session tokens and other authentication factors as well as managing authorisation decision	ASVS: 6.1.1, 6.2.1-6.2.5, 6.2.8, 6.3.1, 6.3.2, 6.4.1, 6.4.2, 7.2.1, 7.2.2-7.2.4, 7.4.1, 7.4.2, 8.1.1, 8.2.1, 8.2.2, 8.3.1 9.1.1-9.1.3, 9.2.1, 10.4.1-10.4.5 SY:05, SY:12-15, SY:18 SY:19, SY:24, SY:30
Secure Data Processor	New	Responsible for validating untrusted data is transformed to a trusted form so it can be securely processed rendered and stored.	ASVS: 1.2.1-1.2.5, 1.3.1, 1.3.2, 1.5.1, 2.1.1, 2.2.1, 2.2.2, 3.2.1, 3.5.1, 5.2.1, 5.2.2, 5.3.1, 5.3.2, 11.3.1, 11.3.2, 11.4.1, 15.3.1

Note: System Requirement denoted by SY, Application Security Verification Standard denoted by ASVS.

Table 10 below contains identified interfaces and connectors related to the functional elements identified by applying the security viewpoint to the functional structural model.

Table 10: Specifications of additional interfaces and connectors included in the functional structural model by applying the security viewpoint.

Name	Associated Connector	Input	Output
User request	Connects User Interaction Manager with Secure Data Processer	User-request	System-response
Authorisation Request	Connects Secure Data Processer with Account Manager	Authorisation request	Authorisation decision
Check Stored Account Records	Connects Authentication Manager with Storage	Account information of party requesting access	Matching Account Records

The security viewpoint was also applied to the deployment view, where the analysis mainly focused on determining what would need to be in place for the application to be securely deployed. Looking at Figure 8, the runtime platform model from the deployment view does not appear to have changed very much. However, when reviewing the ASVS requirements, several requirements were dependent not only on the application code but also on additional deployment configurations.

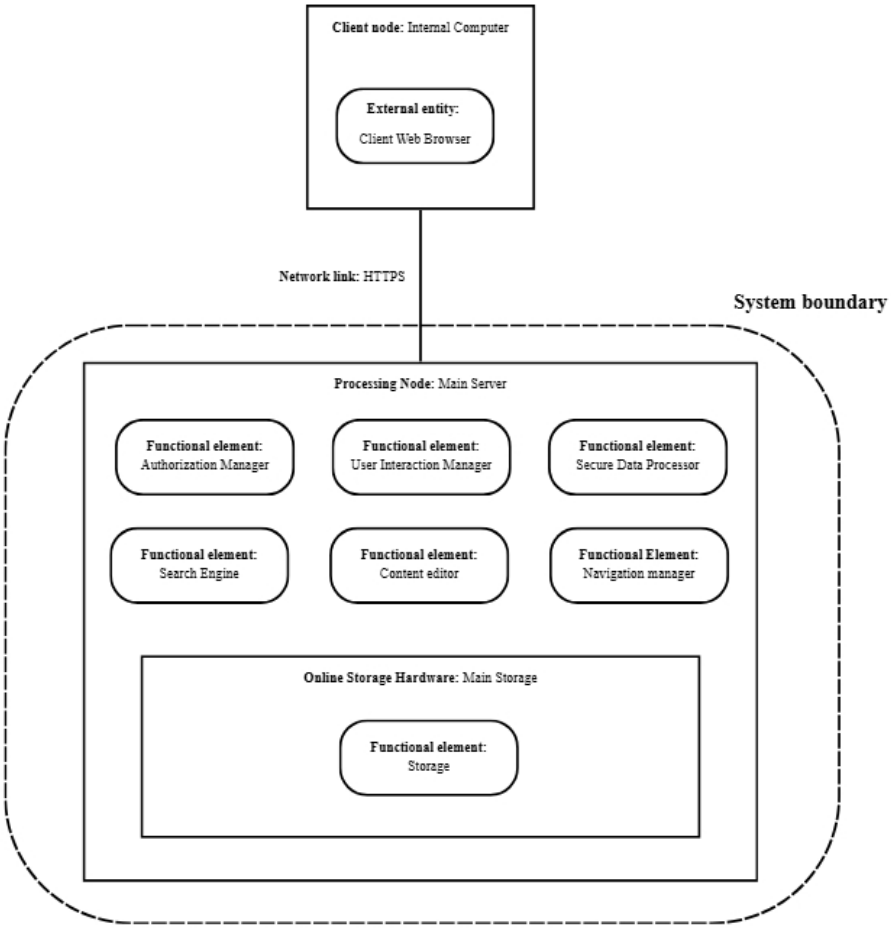

Figure 8: Runtime platform model with additional functional elements identified by applying the security viewpoint to the deployment view.

Particularly, requirements concerning secure communications using TLS certificates and protocols are dependent on server configuration before deployment. Thus, these are added as responsibilities of the Main server. The ASVS requirements also state it should be ensured through removal or access control that the application is not deployed while source control files are accessible, and default accounts created during application development should be deleted. Moreover, the file upload directory should be configured to prevent code from executing (OWASP, 2025a). All these mentioned responsibilities placed on the server support the functional elements in fulfilling their responsibilities and functionality. Thus, the responsibilities summarised in Table 11 are considered the minimum to require from an operational environment where the system is to be deployed.

For a detailed description of the requirements that the deployment environment is expected to fulfil, consult Appendix A.

Table 11: Specifications of network links in the runtime platform model.

Name	Existing/New	Responsibilities	Derived from
Main Server	Existing	TLS certificate and protocol, TLS 1.2 or 1.3. Source control folders should be removed or placed under appropriate access control. Configuration of upload directory to prevent scripts in files to run code on server. If WebSocket (WS) communication, is used force use of TLS so only WWS is accepted HSTS configuration should be used, forcing HTTPS, mTLS or DPoP if OAuth/OIDC tokens are used	**ASVS:** 3.3.1, 3.4.1, 4.4.1, 5.3.1, 6.1.1, 6.3.2, 10.4.5, 12.1.1, 12.2.1, 12.2.2, 13.4.1.

Note: Application Security Verification standard is denoted by ASVS

In addition to the Main Server, the network link of the runtime platform model, described in Table 12, has been updated to specify only HTTPS. However, this is an effect of the requirements imposed on the Main Server rather than the responsibility of the network link.

Table 12: Network links in the runtime platform model.

Name	Type of Network Connection	Connects	Specifications
HTTPS	HTTPS	Client Computer with Main Server	Only HTTPS

5.2.3 Design Definition Process

This section will present the results of the Design Definition process, which include a review of the prerequisites of MediaWiki and alignment of MediaWiki with the architecture description.

5.2.3.1 MediaWiki Prerequisites

In Table 13 below, the prerequisites of MediaWiki are presented.

Table 13: The Technical prerequisites of MediaWiki.

Type	Required technologies
Web server software	Either Apache or Nginx
PHP language	PHP ≥ 8.2
PHP extensions	Calendar, ctype, dom, fileinfo, iconv, Json, OpenSSL, xml, xmlreader, intl, mbstring, bcmath or gmp
Database server	Either MariaDB 10.3.0+, PostgreSQL 10.0+ or SQLite 3.24.0+
Command line access	Specific technologies not specified

Note: All information in this table is retrieved from (MediaWiki, 2026e).

5.2.3.2 Alignment of MediaWiki with Architecture Description

The default MediaWiki package was determined to align with the architecture description for the most part. Creation and editing of pages, as well as administrator-controlled pages (MediaWiki, 2026j) and restoration of earlier versions of pages is facilitated (MediaWiki, 2025h) as well as storing of passwords as salted hashes (MediaWiki, 2026d). However, several responsibilities are only partially fulfilled or require additional code configuration, extensions, or external technologies (MediaWiki, 2025s). Response times below 0.5 seconds for displaying search results cannot be guaranteed as search response time is affected by database performance, hardware resources, network capacity and other factors that may vary between deployment environments (Cambazoglu & Baeza-Yates, 2016). Functionality for overview of contents and content structure exists by default via special pages that list existing pages and categories of the Wiki (MediaWiki, 2025n; MediaWiki, 2025f), but more advanced overview structures can be implemented by installing the extension CategoryTree (MediaWiki, 2026a).

The default MediaWiki code does not require users to sign in to access the web application, but MediaWiki can be set to a so-called private Wiki where users are required to have an account and sign in to access the Wiki (MediaWiki, 2026g). A private Wiki

implements a type of Role Based Access Control (Wikimedia Foundation, 2025y). MediaWiki (2026g) lists several measures that should be taken to ensure access control is properly implemented but emphasises that these recommended measures might not block accesses in all ways, which could lead to private information being shown despite the Wiki being set to private. One such example is that files may remain publicly accessible by typing in the URL specific to that file, if not additional configuration of the server file directory is conducted. Therefore, it is also strongly recommended that thorough testing and customisation of the MediaWiki code is conducted to ensure that access control is properly implemented. In addition, multi-factor authorisation can be fulfilled with additional MediaWiki extension OATHAuth although this requires additional PHP extensions, configuration of the database and often an external user-side authentication app. (MediaWiki, 2026b; MediaWiki Foundation 2026). While administrators can create pages that are only modifiable by the administrator user group no user group can completely delete pages from MediaWiki. Even though the pages are removed from the Wiki and restorable only by the administrator user group they are newer fully deleted from the database. For content to be completely erased system administrators need to do this through command-line access. Further analysis is presented in Table H1 in Appendix H.

The following paragraphs give an overview of the results of analysing how well MediaWiki default open-source code aligned with the ASVS requirements specified as responsibilities in the architecture description. A more detailed description of the findings is provided in Table H2 in Appendix H. The analysis revealed several useful control mechanisms included in the default MediaWiki code package and if implemented consistently across the codebase, these controls could help ensure that basic application-level security is upheld. Examples of control mechanisms present in the default version of MediaWiki include mechanisms for encoding, sanitisation (Wikimedia Foundation, 2025v), secure database queries (Wikimedia Foundation, 2025q), session management, allowed password formats (Wikimedia Foundation, 2025x), and invalidation of session tokens on logout (Wikimedia Foundation, 2025at). However, many controls need further configuration of default MediaWiki. Allowed file types, file sizes, threshold for number of sign in attempts before lockout (Wikimedia Foundation, 2025x), permitted URL-formats (Wikimedia Foundation, 2025n), and other security relevant parameters should be set to values appropriate for the specific deployment environment in which the Wiki is intended to operate.

Some requirements also depend on additional MediaWiki extensions. The MediaWiki extension OAuth should be installed and to enable use of the required OAuth protocol (MediaWiki, 2025a). To enable blocking of accounts additional parameters need changing of values. For accounts to be removed from being visible in MediaWiki the UserMerge extension can be installed, although it requires many steps for removal of a user and does not seem to be easy to use. In addition, user account records are not fully removed even though the user is deleted, these records still exist in the database and need to be removed through command line access (MediaWiki, 2025d). Additionally,

MediaWiki includes parameters for secure communications such as secure cookies and HTTPS-behaviour, but these must be configured correctly and supported by server-side TLS settings (OWASP, 2025b). Finally, fulfilment of many ASVS requirements is dependent on preparatory analysis and configuration of both the MediaWiki code and additional technologies. When the architecture and design were defined, they were analysed alongside the supervisor at AFRY, to receive approval to proceed to the next stage, namely the Production stage. This analytical meeting served as a decision gate between the life cycle stages.

5.3 Production Stage

This section will present the results of the processes performed in the Production stage of the information management system. The section will begin by presenting the technical specifications of the developed system, the implemented extensions in the system and the implemented access control policy for the users. This is followed by a presentation of the finished developed system with images of it. Finally, the results from both the user testing and the functional testing will be presented.

5.3.1 Technical Specifications

The technical specifications of the information management system are stated below, in Table 14.

Table 14: The technical specifications of the system.

Component	Specification	Comment
Software	MediaWiki	Downloaded from MediaWiki.org Version: 1.45
Programming language	PHP	Installed on server Version: PHP ≥ 8.2
PHP extensions	Calendar, ctype, dom, fileinfo, iconv, Json, OpenSSL, xml, xmlreader, intl, mbstring, bcmath or gmp	All required PHP extensions were acquired
Operating system	Ubuntu TLS (Linux) server	Provided by internal IT-department at AFRY
Web server	Apache	Installed on the Ubuntu server via an SSH connection
Database	MariaDB	Database created during the setup process. Tables generated when configuring the MediaWiki software Version: MariaDB ≥ 10.3.0
File handling client	WinSCP	Used as a graphic tool for file handling between computer and the Ubuntu server
SSH (Secure Shell) client	PuTTY	Used to securely connect to the VM on the Ubuntu server and execute commands
Hosting type	Virtual machine (VM)	Provides an isolated server environment on the Ubuntu server

Note: Exact versions of required third-party software used was not specified beyond the minimum supported versions required by MediaWiki 1.45, as AFRY did not wish to disclose more detailed infrastructure information.

The implemented extensions in the system are presented in Table 15 below, as well as their purpose.

Table 15: The implemented extensions in the information management system.

Extension	Purpose
InputBox	Creates an InputBox for easier creation of new pages
VisualEditor	Aids the creation and editing of pages, by providing a view which replicates how the finished page will look.
CategoryTree	Visual tree structure of categories
PDFEmbed	Allows PDF files that have been uploaded to the Wiki to be embedded, via a scrollable window, into a Wiki page
Upload Wizard	Aims to make it easier to upload files by adding a step-by-step guide in the file uploading area

Lastly, an Access Control Policy was developed for the intended user groups, as defined in the requirements. The system has two intended user groups: project members and project manager. These groups were assigned different privileges in the system. For instance, the project manager can choose to protect certain content, such as governing project documents, which are project members are only granted read access to. See Table 16 below for the Access Control Matrix for the implemented policy.

Table 16: Access Control Matrix representing the implemented Access Control Policy in the system, for different user groups.

Object/Subject	Anonymous (not authenticated)	Authenticated User (Project Member)	Authenticated Administrator (Project Manager)	System Administrator
Content	None	Read Write Modify Archive Recreate from archive	Read Write Modify Archive Recreate from archive	Read Write Modify Delete
Protected Content Created by Admin	None	Read	Read Write Modify Archive Recreate from archive	Read Write Modify Delete

5.3.2 The Final Information Management System

This chapter will present the final information system created in this thesis project. This chapter will present images of selected parts of the system, along with guiding explanations to create a feel of the usage of the system.

The first page the user sees when accessing the system in their browser is a page informing them that they need to log in. When the user clicks on the hyperlinked text, they are taken to a page with input fields for their username and password. Here, the user can also get help with logging in or recovering their password if they have forgotten it. See Figure 9.

Figure 9: a) The page informing that the user needs to log-in, and b) the log-in page.

The next page the user is directed to after logging in is the home page. This page welcomes the user, provides an overview of the Wiki's content, includes an input box for easily creating new pages, displays the Wiki's language guidelines, and provides a hyperlink to the software user guide, which users can access as needed or if interested. The icon for the Wiki has been customised using the partner company AFRY's logo. In the sidebar, which is always present in the system, several useful links are placed. The top section of the sidebar, which was part of the default package, includes links to the home page and a link where the last changes in the system can be viewed. Furthermore, a link which navigates the user to a random page in the Wiki. This is a functionality which is intended to be a fun way for users to explore content. Lastly, a link to the documentation of the MediaWiki software and to special pages, which includes more advanced features such as statistics of the usage of the Wiki. The next section, which has been implemented as part of the customisation of the system, includes links that allow users to see an overview of the content and the categories in the system. The third and last section in the sidebar includes various tools. These include an overview of pages that links to the page the user is currently on, and a link where the user can see changes that have been made to pages which links to the current page. Furthermore, the tool for uploading files, which takes the user to a separate view made to facilitate the process. Lastly, the possibility to create a permanent link to the current version of the current page, detailed information about the current page (such as when the page was created and by whom and if there are any restrictions in privileges for different users linked to the page) and finally if the user

would like to use the current page as a reference, the last link provides references in different styles such as APA and Harvard. See Figure 10.

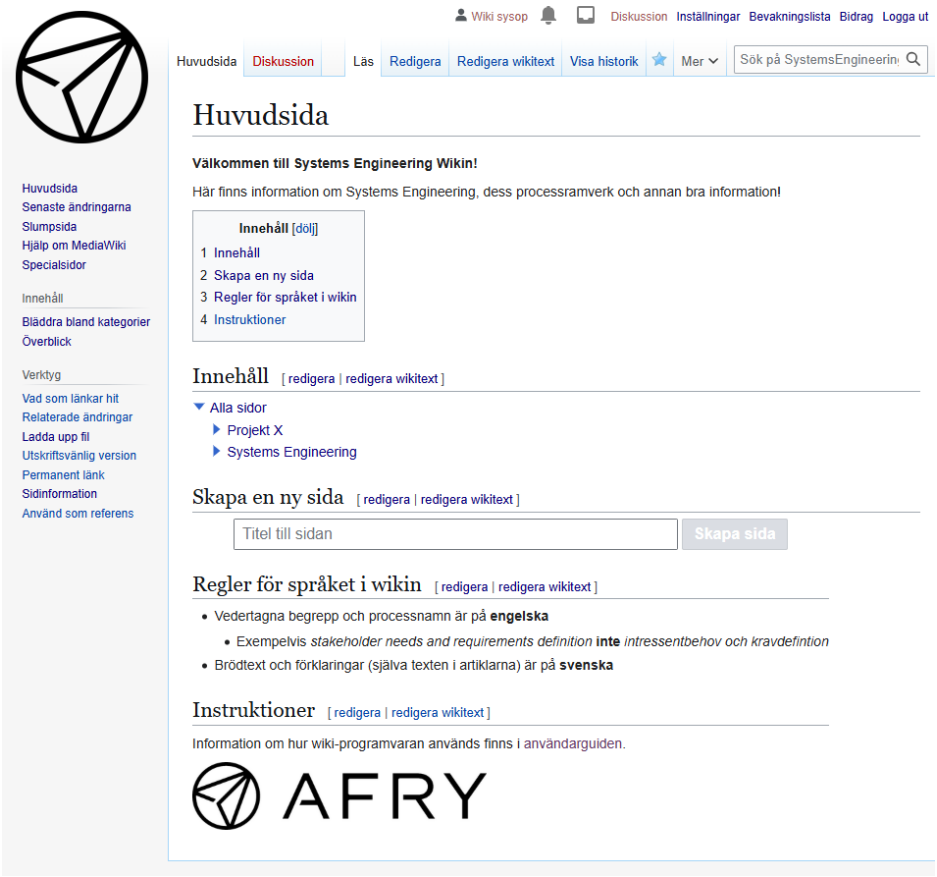

Figure 10: The main page of the system.

As previously described, the home page in the system includes an overview of the content. This overview was created using the CategoryTree extension. The established hierarchy consists of two main categories: Systems Engineering and Project X. The Systems Engineering category contains the theoretical framework of Systems Engineering itself, with a first level subcategory including definitions and process framework. The definitions category contains one page with terms and their corresponding explanations, similar to a glossary. Additionally, in this category there are pages with information on other definitions used in Systems Engineering. The processes framework category contains all lifecycle processes, with the existing subcategories within the framework and all processes as part of these subcategories. The category called Project X has been used to illustrate how project-specific information can be entered into the system. This category was also used for this purpose during the user testing. All categories and pages in this hierarchical overview are sorted in alphabetical order. To expand or collapse the content of this hierarchical overview, the arrows are clicked. See Figure 11.

Figure 11: The hierarchical structure of categories in the system and their associated pages.

The purpose of the system is to manage information, and this is achieved through information pages within the system. When a user navigates to an information page, they are presented with a standardised page that includes the title and a brief, clickable overview of the content. This is an example of an information page that explains the process of information management (a process part of the Technical Management processes in the Systems Engineering framework) with added explanatory images on the right-hand side. The images are clickable to see a larger version. See Figure 12.

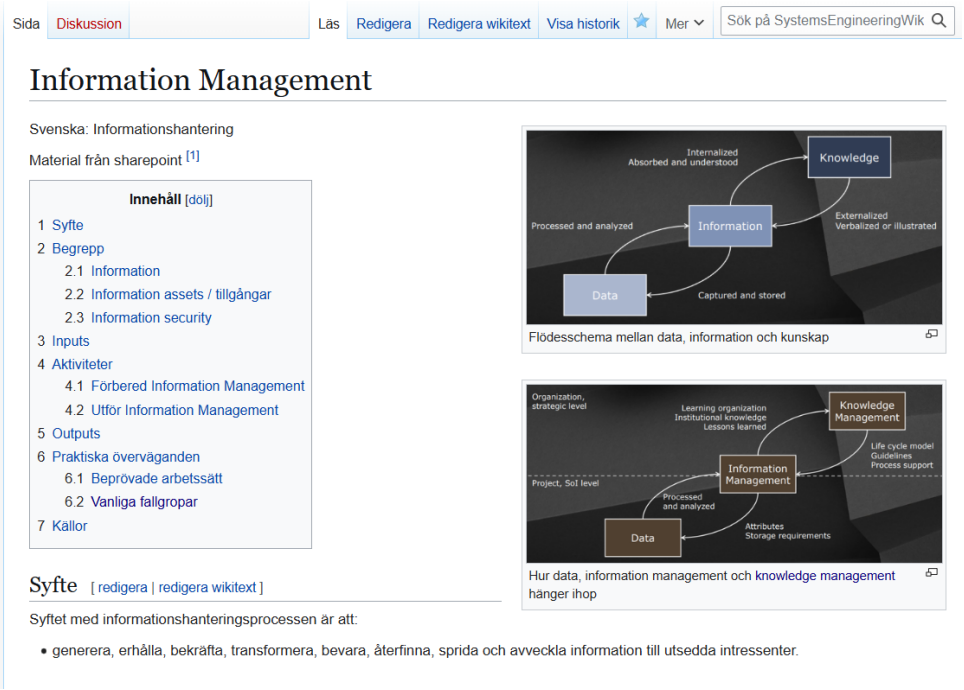

Figure 12: An example of an information page in the system.

At the bottom of each page, there is a section for source references. When sources are used while creating information pages, they are numbered and clickable links to the source are added to this section. At the very bottom, which category the page belongs to

is visible. This is also clickable, navigating the user to the corresponding “category page” where all members in that category are visible. See Figure 13.

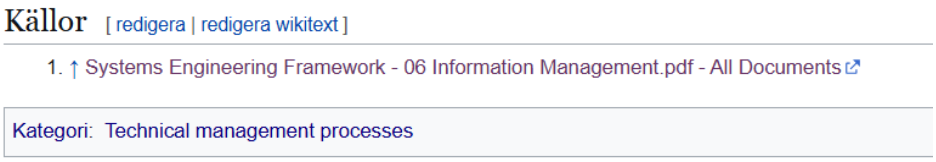

Figure 13: An example of the bottom of a page, showing the sources and which category the page is assigned to. This is from the page about the process called information management.

When a page should be edited, the user clicks “Edit” (“redigera” in Swedish) to open the visual editor, or “Edit Wikitext” (“redigera Wikitext” in Swedish) to edit in this format, which is like Markdown (a coding language used to format text). This can be done either at the top of each information page or next to each title within the information page, so the user can avoid scrolling all the way to the top to edit the page. In the visual editor users can, among other things, use the visual buttons to choose between different formatting options, add hyperlinks to other information pages, and cite sources. In the Wikitext editor, users need to know the rules for this style of writing where, for example, a heading formatting is created by adding “==” on either side of the heading. The user can switch between the different editing options by clicking on the pen in the right-hand corner of the top-menu. To help the user, a pop-up is shown when switching editing modes explaining they can switch back by using the pen. See Figure 14.

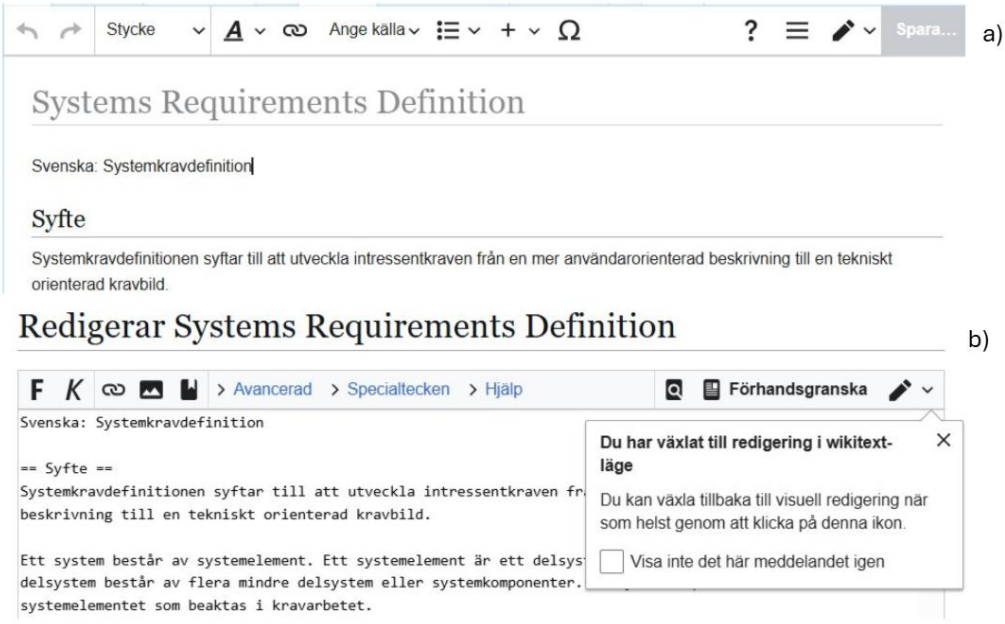

Figure 14: a) The visual editor and b) the Wikitext-editor.

5.3.3 Testing and Verification of Requirements

This chapter begins by presenting the results of the inspections and functional tests of the system conducted to assess whether stakeholder and system requirements are met. This is followed by the results of the user tests conducted to evaluate the system's usability.

5.3.3.1 Functional Testing and Inspection

The functional testing and inspection were conducted to verify the stakeholder and system requirements, that were not tested in the user testing which consisted of twelve out of the 25 stakeholder requirements and all 30 system requirements.

As for the stakeholder requirements, the results were that all were met since the specified functionalities can be found in the system. Three of these 25 requirements were however met indirectly. This since existing system functionality could be used to achieve the desired functionality described in the requirements, even though it was not explicitly implemented in the system. These were ST:21, ST:22 and ST:23, which state that project members should be able to suggest changes to pages they do not have permission to edit; submitting a suggestion should not alter the page, and the project manager should be notified when a suggestion is submitted. There is no "Suggest button" in the system, but instead, a user can start a discussion on a page they are not permitted to edit and post their thoughts and suggestions. If the project manager who created this page has it on their watchlist, also called favourite function in the requirements, they will receive a notification. Once a discussion is created, any interested users can subscribe to the discussion and thus receive a notification when the discussion thread is updated. While these features meet the requirements for making suggestions, they do not do so in a very intuitive manner.

As for the system requirements, the results were that 27 out of 30 system requirements were fully met. Requirement SY:30, regarding implementation of multi-factor authentication, was not met since this would require substantial configuration as well as several external technologies, which was not feasible to implement within the timeframe of this thesis project. System requirements SY:14 and SY:15 were partially met. Admins can remove all contents and users can delete only user contributed information. However, as explained in section 5.2.3.2, Alignment of MediaWiki with architecture description, content can never be fully deleted from MediaWiki unless they are deleted through command-line access. Additionally, Requirements SY:08 and SY:09, regarding uploading files in PDF and word format, were met but in order to upload, some extra steps are required in the system, and it is not possible to upload directly when editing a page. Since these requirements only specify that this functionality should exist in the system, and it does, they were met. Furthermore, SY:06 states search results should be returned within 0.5 second, although this requirement was found to be fulfilled during the functional testing it is important to point out this does not mean this requirement will always be fulfilled for the system, as described in section 5.2.3.2, Alignment of

MediaWiki with architecture description, search response time depend on factors that may vary between deployment environments.

5.3.3.2 User Testing

This chapter will begin by presenting the quantitative results of the user testing, including the observations and the score from the SUS questionnaire. These are followed by the qualitative results, from the observations and think-aloud comments.

The quantitative results from the observations are presented in Table 17. The data is presented in the following order for each task: user test 1, 2, and 3. The pilot test was not included in this table since this session was not screen recorded. A line represents that this task was not conducted during the user test, which was the case for task 1B and 1C, since these tasks depended on how the user solved task 1A. Depending on if the user navigated or used the search function to find the pages in task, they were given either task 1B or 1C to test the functionality they didn't use in 1A.

Table 17: Quantitative results from the observations.

Task ID	Task	Assistance needed	Actions not leading toward task completion	Backtracking
1.A	Find two pages	0, 0, 0	1, 0, 0	1, 0, 0
1.B	Navigate to page	-, -, 0	-, -, 2	-, -, 2
1.C	Search for page	0, 0, -	0, 0, -	0, 0, -
2.A	Create new page	0, 1, 0	0, 4, 2	0, 1, 1
2.B	Add picture to created page	0, 0, 0	1, 0, 0	0, 0, 0
2.C	Add category to created page	4, 0, 0	6, 1, 0	2, 1, 0
3	Edit existing page	0, 0, 0	0, 0, 0	0, 0, 0

Table 18 presents the results of the SUS (System Usability Scale) questionnaire, filled in by the participants at the end of the user test sessions.

Table 18: The SUS (System Usability Scale) scores for each user test and the mean score.

Index	SUS scores
Pilot	92.5
User test 1	82,5
User test 2	77,5
User test 3	90,0
Mean	85,6

As previously written, according to (Sauro & Lewis, 2016), the mean score of the SUS, calculated from 446 studies, is 68 with a standard deviation of 12.5. In this thesis project, the mean of the SUS scores was 85,6, placing the results more than one standard deviation

above the mean calculated by Sauro and Lewis (2016). This suggests a very good usability.

The two additional statements added to the questionnaire, regarding the visual appeal and the perceived time to solve tasks in the system, received a mean of 2 and 3.3 respectively on the five graded Likert scale (from Strongly disagree to Strongly agree). These statements were linked to the two non-functional requirements (ST:15 and ST:20) which was defined to receive three of less, and three or more respectively. Due to this, these requirements were met.

The following qualitative results will be presented divided into the main functionalities that were tested in the system followed by overall findings. In these results, the pilot test is included.

Navigation and overview of content

- Two of the participants did not realise that the arrows in the overview could be used to expand the tree to see more options of subcategories and pages, which could have been used for a faster path. Instead, the name of the category was clicked, and the right page was found by navigating through the category page.
- One of the participants who used the arrows expressed a positive comment, stating that this was a good way to get an overview of the information in the system and that it was very useful.
- When navigated to the first page in task 1A, two of the participants used the hyperlink between the two pages to navigate to the second page. Both commented that they enjoyed this functionality and that it helps to create an understanding of how different topics relate to each other.
- One user requested clearer indicators of where one is located in relation to the rest of the content in the system. One suggestion mentioned was placing the tree structure in the sidebar, in which the current location should be highlighted.

Search

- All participants located the search function without any trouble and could use it effectively. All commented that they enjoyed the search function.

Creating page

- All participants showed signs of some level of uncertainty regarding where to start in order to create a new page. It was not clear that this feature was located on the main page.
- When the input field to create a new page was found, all participants created a new page without trouble.

- Some uncertainty in regards of choosing the title for the page was however shown by all participants. One participant attempted to change the title of the page retroactively, which was not possible and led to some frustration.
- All participants located and used the tools in the menu in an efficient way.
- All participants used the visual editor to create the page. One user did however switch to the Wikitext editor and commented in a positive manner that they did so out of curiosity.

Uploading picture

- All participants quickly located the button at in the top menu top of the editor used to upload images.
- When uploading the image, all participants showed some level of negative surprise and/or frustration in regards of the following steps after choosing the image, which included adding a description of the image and optionally a caption and alternative text (part of the system designed to improve accessibility for users with visual impairments).
- All users voluntarily (not part of the task) went back and edited the layout of the image, which by default is placed on the right side of the page. One user began by trying to change the location of the image from the right side to the left side, by using "drag and drop". This was without success since drag and drop is only functioning when changing the location of the image vertically, not horizontally which instead is part of "advanced settings". All users did however manage to change the location of the image without trouble once the advanced settings were located.

Adding category

- Two participants clicked the "Move" button at the top of the page when tasked to add the category. One participant commented that they considered clicking the "Move" button but chose not to do so, which resulted in finding the correct button. These results show that there were some uncertainties in regards of finding the button to add categories.
- When the button to add categories was located, all users could add the correct category without any trouble.
- When tasked to verify that the category was added successfully, two participants commented that the category was visible at the bottom of the page. The other two participants instead navigated to the tree structure and then to the category page to verify.

Edit existing page

- All users edited the existing page quickly and without any hesitation.

Other findings

- All four participants commented that the interface of the system was familiar since they have previously used Wikipedia. They said that this feeling of familiarity made it easier to understand and get started with the system.
- All participants noted that prior knowledge of Systems Engineering makes it easier to use the system, as it provides a better understanding of which processes fall under which category.
- Two of the participants commented that they would have liked more graphic symbols included in the system, to clearer communicate where different functionalities are located. An example was to have a button with a plus-symbol where new pages could be created.

Overall, the user testing results indicates that the system demonstrates a high degree of usability. After identifying the relevant functionalities in the system, users were generally able to complete tasks without difficulty. However, certain functionalities, such as adding a category to a page, could be presented more clearly to improve discoverability and enable more efficient interaction, particularly for new users.

Lastly, an evaluation was conducted in which the stakeholder requirements, for which user testing was the chosen verification method, were compared against the test results. Based on this evaluation, it was determined that all of these stakeholder requirements had been met.

6. Discussion

This chapter begins with a discussion of the information management system, which is then followed by a discussion regarding the chosen methodology. Finally, the future system development and deployment is discussed.

6.1 The Information Management System

This subsection includes a discussion of the validation of the system in its entirety, goes on to discuss the security aspects of the system and ends with a discussion of the generalisability of the system.

6.1.1 Validation of the System

The process of validation is answering the question whether if the correct system has been built to meet the stakeholders' needs in a sufficient manner. The results from the user testing indicated that the system has generally good usability and supports users in completing the intended tasks. Most participants were able to accomplish the assigned tasks without major difficulties, suggesting that the overall design of the system is intuitive. However, some usability issues were identified from the user testing. For instance, some participants experienced difficulties in locating specific functionalities or understanding certain parts of the navigation. These issues highlight areas where the system could be further improved. Despite these challenges, the overall task success rate suggests that the system performs sufficiently well within the defined context of use. This is further supported by the high score from the System Usability Scale (SUS) questionnaire, indicating a high perceived usability among the participants. From the perspective of validation, these results provide evidence that the system is functional and usable, while also showing opportunities for improvement.

The validation results were also analysed in relation to both stakeholder and system requirements. All stakeholder requirements were met, indicating that the system successfully meets the needs identified from the conducted interviews with stakeholders. This is further supported by the user testing results, where participants were generally able to use the system effectively. Since stakeholder requirements are closely linked to the needs of the users, this demonstrates that the functionalities implemented in the system is fit for its intended use. While the majority of system requirements were met, one requirement related to authentication (SY:30) was not met since it was considered not feasible, and two additional requirements concerning content deletion (SY:14, SY:15) were not fully met since MediaWiki does not support full deletion as described in section 5.3.3.1. Functional testing and inspection. Despite not all system requirements being met, the core functionality of the system is considered to work to an extent that fulfils its purpose. However, this highlights another area for future development.

The validation results can also be interpreted from the perspective of information architecture, particularly in relation to how the content in the system is structured, organised, and presented to users. A well-designed information architecture facilitates navigation, improves findability and improve the overall user experience. The user testing results does on one hand suggest that the system's information architecture is overall effective. This since participants were generally able to locate relevant information and navigate between different sections in the system without significant confusion, indicating that the structure and categorisation of content are intuitive and aligned with established principles of information architecture. On the other hand, the identified usability issues could be linked to some limitations in the information architecture. Difficulties in locating specific features could suggest that some elements may not be optimally placed or labelled. Furthermore, all six key components information architecture (taxonomies, metadata, labelling systems, organisational structure, navigation systems, and search systems) can be identified within the system or are supported through the functionality of the system. The three components of labelling, search and navigation can be directly linked to functionalities in the system. While search and navigation are obvious functionalities, the component of labelling can be directly linked to the possibility to add categories and subcategories in the system. The other three components of taxonomies, metadata and organisational structure, can be implemented in the system by using existing functionalities, such as adding metadata when needed in information pages. However, the effectiveness of these components depends on how users utilise the functionalities of the system and on users defining clear rules and conventions for their use.

Lastly, it is important to note that the validation conducted in this thesis project has certain limitations. The user testing was conducted in a controlled environment and was limited to four selected participants. This can mean that the conducted testing may not fully reflect how the system could be used when deployed and may not reflect the experience of all different types of possible users. Furthermore, the prepared task-based scenarios could only partially cover the possible interactions with the system.

6.1.2 Security

When reviewing the results of the analysis regarding what controls MediaWiki has that could be implemented to fulfil the level one ASVS requirements, it was concluded that MediaWiki default open-source software implements several useful controls to ensure a secure system. These include controls for output encoding, restrictions on accepted URL protocols, encoding of JavaScript and JSON, mechanisms for secure handling of database queries, validation of file uploads, authentication, authorisation, and session handling. However, for many of the controls to be effective, some code configuration is needed primarily through changing allowed parameter values or the enabling of parameters. Additionally, some security-related MediaWiki extensions may be required, and a lot of the essential security controls rely on configuration of the deployment environment. Overall, the MediaWiki default open-source code provides many core controls necessary

to fulfil the ASVS requirements. Further code configurations, MediaWiki extensions, and deployment-dependent adaptations are however required to fulfil all the ASVS requirements addressed in this thesis.

An important point to note is that these results do not provide a basis for judging whether MediaWiki in fact fulfils the ASVS requirements in question, rather that they give an indication of whether controls exist that could be leveraged to fulfil the requirements if appropriately applied throughout the MediaWiki code. For an organisation whose main assets are the information and knowledge it produces, simply trusting that publicly developed open-source software will provide a reliable system might not be an acceptable risk. Performing a full review of the MediaWiki code to validate that the system is secure enough for organisational use could therefore be important. However, validating that these assets are handled securely would be expected to require validating not only the 70 level one ASVS requirements but all 345 ASVS requirements. Due to the size and complexity of the MediaWiki code base, the collaborative nature of its development, and the number of ASVS requirements to consider, validating the security of a MediaWiki application would likely be a substantial undertaking.

When viewing MediaWiki from a security perspective, a particularly interesting aspect is that the default open-source version of MediaWiki enforces complete openness and does not implement any security controls that restricts who may access the Wiki and its contents. For security controls such as authentication or access control to be implemented, the Wiki needs to be made private, additional configuration of both the MediaWiki code and deployment environment is required, and thorough testing is recommended to ensure defined authorisation rules are consistently followed. Moreover, implementing multi-factor authentication and enabling deletion of user accounts demands extensive configuration and there is an inability to fully delete content in the system using the web interface. Files and information can be archived through the web interface by selected user groups, but to fully delete information, access to the underlying server or database is required. This means that information cannot be swiftly deleted from the Wiki, which can become especially problematic in situations where sensitive data, that should not be viewed by all Wiki users, have been mistakenly or intentionally published in the system. All the above-mentioned aspects clearly demonstrate that MediaWiki's original purpose is to be an open and collaborative system, where anyone should be able to take part in and contribute to the content in the Wiki.

6.1.3 General Discussion of the System

The concept of a Wiki is fundamentally built on openness with the definition stating that all users are meant to be able to contribute and edit to the content (see 2.1 Information Management and Wiki). However, when managing information assets in a secure manner, particularly to ensure information integrity and confidentiality, some restrictions become necessary. In the system developed in this thesis project, measures such as implementing different privileges for the two intended user groups were introduced. Although these

restrictions partially contradict the definition of a Wiki, since it limits the possibility of collaboration, it was a necessary adaptation to meet defined requirements. This trade-off between openness and control has been interesting during the development and it highlights the challenge of balancing collaborative ideals with the need to protect and maintain reliable information in practise.

When it comes to the generalisability of the system it can on one hand, be assumed that a similar system could be applicable to organisations with comparable needs. The general design principles and functionalities implemented in the system may therefore be transferable to other settings with similar requirements. On the other hand, the system has been developed based on needs and requirements elicited specifically from stakeholders within the partner organisation. As a result, the system reflects the preferences of the stakeholders and the constraints shaped by how they intend to use the system. This may limit how widely the system can be used outside this context, and for organisations with different processes or needs the system would likely require adjustments.

6.2 The Chosen Methodology

The Systems Engineering methodology used in this project was based on ISO 12207, which provides a structured framework for software lifecycle processes. While this standard offers a comprehensive and well-defined approach to system development, its suitability for this thesis project can be discussed in relation to the project's scale and context.

ISO 12207 is primarily designed for large-scale projects with long timeframes, whereas this thesis project was relatively small in scope and was carried out by a team of two within a limited timeframe. Due to this, to make the method more suitable for this thesis project, a certain degree of tailoring of the framework was required. Although this tailoring process was at times challenging, particularly due to the limited prior experience, it also created flexibility and made it possible to focus on the most relevant processes and activities for the project. When tailored, the standard provided systematic and organised way of working throughout the project.

For several of the processes, it was also necessary to combine ISO 12207 with additional methods, since the standard focuses on defining what activities should be performed rather than providing guidance on how they should be done. The methodology was therefore supplemented with relevant literature on for instance interview methodology, techniques for developing architectural models, and methods for conducting user testing. While this combination of methods increased the complexity it also led to greater flexibility, such as the tailoring process did, where the methodology could be adapted to this specific thesis project.

Another challenge in this thesis project was the limited access to a complete system context. The development of the information management system was based on partial insight into the intended operational environment at the partner organisation, which made it difficult to fully apply certain Systems Engineering principles that rely on a comprehensive understanding of the system context. The methodology did however still provide helpful framework to the project, ensuring that key aspects of system development were considered.

6.3 Future System Development and Deployment

The information management system developed in this thesis project focused on implementing the core functionalities necessary to demonstrate the overall concept of the system to stakeholders. Due to the lack of contextual information about AFRYs security policies and technical infrastructure it was evaluated whether MediaWiki possessed a general set of security controls. This analysis showed promising results indicating many of the required security controls were present in the MediaWiki default open-source code. However, several controls require configuration of code and additional technologies in the deployment environment largely due to MediaWiki being built on principles of openness rather than information security. In addition, further analysis is recommended to ensure the level one ASVS requirements on security controls are consistently implemented. Importantly, information security is highly dependent on its context and if further security controls are required, and which additional security controls are required, should be reviewed before deployment into each specific operational environment.

In addition, the results from the user testing indicated that some functionalities could be presented in a clearer and more intuitive way. Improvements could due to this be made to the presentation of functionalities, for example through clearer visual elements. For instance, during the user testing, a suggestion was made to incorporate more symbols to improve the clarity and communication of system functionalities, with one example being a plus symbol to create a new page. However, as MediaWiki-based systems are mainly text-based, there are certain limitations regarding the implementation of visual elements. Nevertheless, this aspect represents a potential area for future development and should be further explored. Furthermore, providing clearer instructions directly in the relevant sections may be relevant to improve usability, especially for new users. This could for instance be done to the section where a new page is created.

During development and testing, the information management system has been hosted on a server environment created specifically for this thesis project, which was suitable for these purposes. However, since the complete context of the intended operational environment was not fully disclosed due to security restrictions, the system should be reviewed by qualified IT personnel at AFRY before deployment, to ensure that it meets organisational and security requirements.

7. Conclusions

The results of this thesis project suggest that a Wiki-based solution using MediaWiki can be a suitable approach for information management in transdisciplinary teams, particularly where collaboration is wanted to build a shared knowledge base. For the system to reach its potential, it depends on users consistently implementing and following guidelines set up utilising functionality to uphold the information architecture of the system. At the same time, certain restrictions should be implemented to ensure secure and reliable information. This demonstrates the need to balance openness with control in practical applications of a Wiki. For the system to be deployed, further development is required, especially in regards of the implementation of controls to ensure a secure system.

The Systems Engineering framework based on ISO 12207 provided a clear structure and supported a systematic way of working in this thesis project. Due to the project's small scale and limited resources, the standard needed to be tailored, which at times was challenging. Both in terms of deciding which processes were relevant and finding the right level of rigor in the applied processes. However, this also led to flexibility and allowing the methodology to be adapted to the specific context of this thesis project.

References

AFRY. (n.d.a). History, over 130 years of Making Future. Accessed 2026.02.10 from https://afry.com/en/about-us/history

AFRY. (n.d.b). Our brand portfolio. Accessed 2026.02.10 from

https://afry.com/en/about-us/our-brand-portfolio

Atwood, C. G. (2009). Knowledge management basics (1st edition). American Society for Training & Development.

Bai, X., Arapakis, I., Cambazogul, B. B., & Freire, A. (2018). Understanding and leveraging the impact of response latency on user behavior in web search. ACM Transactions on Information Systems, 36(2), Article 21. Accessed 29-05-2026 from https://dl-acm-org.ezproxy.its.uu.se/doi/abs/10.1145/3106372

Barnum, C. M. (2010). Usability Testing Essentials: Ready, Set... Test! (1st edition). Morgan Kaufmann Publishers.

Cambazoglu, B. B., & Baeza-Yates, R. (2016). Scalability challenges in web search engines. Morgan & Claypool. https://doi.org/10.2200/S00662ED1V01Y201508ICR045

Ebersbach, A. (2008). Wiki : Web collaboration (2nd ed.). Springer. https://doi.org/10.1007/978-3-540-68173-1

Fairley, R.E. (2019). Systems Engineering of Software-Enabled Systems. Wiley-IEEE Press. https://doi.org/10.1002/9781119535041

Gabriel-Petit, P. (2025). Designing Information Architecture : A practical guide to structuring digital content for findability and easy navigability (1st ed.). Packt Publishing Ltd.)

McDonald, M. (2024). Grokking Web Application Security. (1st ed.). Manning Publications Co. LLC.

Herald, T., Berkemeyer, W., & Lawson, H. W. (2004). 9.3.1 A Knowledge Management System Life Cycle Description Using the ISO/IEC 15288 Standard. INCOSE International Symposium, 14(1), 1898–1912. https://doi.org/10.1002/j.2334-5837.2004.tb00622.x

Hertzum, M. (2020). Usability Testing : A Practitioner's Guide to Evaluating the User Experience (1st ed. 2020.). Springer International Publishing. https://doi.org/10.1007/978-3-031-02227-2

International Council on Systems Engineering. (2023). Systems Engineering handbook : a guide for system life cycle processes and activities (5th ed.). Wiley.

International Organisation for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2017). Systems and software engineering — Software life cycle processes. (ISO/IEC/IEEE 12207:2017).

International Organisation for Standardization. https://www.iso.org/standard/63712.html

International Organisation for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2023). Systems and software engineering — System life cycle processes. (ISO/IEC/IEEE 15288:2023). International Organisation for Standardization. https://www.iso.org/standard/81702.html

International Organisation for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. (ISO/IEC/IEEE 29148:2018). International Organisation for Standardization. https://www.iso.org/standard/72089.html

International Organisation for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2011). Systems and software engineering — Architecture description (ISO/IEC/IEEE 42010:2011). International Organisation for Standardization. https://www.iso.org/standard/74393.html

Jiao, Z., Li, C., & Chen, J. (2022). Knowledge platform affordances and knowledge collaboration performance: The mediating effect of user engagement. Frontiers in Psychology, 13, 1041767. https://doi.org/10.3389/fpsyg.2022.1041767

Knemeyer, D. (2004, January). Richard Saul Wurman: The InfoDesign interview. InfoDesign. https://informationdesign.org/special_wurman/

Laplante, P. A., & Kassab, M. (2022). Requirements engineering for software and systems (Fourth edition.). CRC Press.

Li, Q., Lee, C.-Y., Jin, H., & Chong, H.-Y. (2022). Effects between Information Sharing and Knowledge Formation and Their Impact on Complex Infrastructure Projects' Performance. Buildings (Basel), 12(8), 1201. https://doi.org/10.3390/buildings12081201

Maqbool, R., Rashid, Y., & Ashfaq, S. (2022). Renewable energy project success: Internal versus external stakeholders' satisfaction and influences of power-interest matrix. Sustainable Development (Bradford, West Yorkshire, England), 30(6), 1542–1561. https://doi.org/10.1002/sd.2327

MediaWiki. (2025a). Download MediaWiki extension. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Special:ExtensionDistributor?extdistname=OAuth&extdistversion=REL1_45

MediaWiki. (2025b). Extension: JWTAuth. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Extension:JWTAuth

MediaWiki. (2025c). Extension: OAuth. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Extension:OAuth

MediaWiki. (2025d). Extension: UserMerge. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Extension:UserMerge

MediaWiki. (2025e). Help: Categories. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Help:Categories

MediaWiki. (2025f). Help: Links. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Help:Links

MediaWiki. (2025g). Help: Managing files. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Help:Managing_files

MediaWiki. (2025h). Help: Reverting. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Help:Reverting

MediaWiki. (2025i). Help: Special pages. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Help:Special_pages

MediaWiki. (2025j). Manual: Block and unblock. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Block_and_unblock

MediaWiki. (2025k). Manual: Configuring file uploads. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Configuring_file_uploads

MediaWiki. (2025l). Manual: MediaWiki architecture. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:MediaWiki_architecture

MediaWiki. (2025m). Manual: Security. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Security

MediaWiki. (2025n). Manual: SessionManager and AuthManager. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:SessionManager_and_AuthManager

MediaWiki. (2025o). Manual: $wgCookieSecure. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Manual:$wgCookieSecure

MediaWiki. (2025p). Manual: $wgFileExtensions. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:$wgFileExtensions

MediaWiki. (2025q). Manual: $wgForceHTTPS. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Manual:$wgForceHTTPS

MediaWiki. (2025r). Manual: $wgPasswordAttemptThrottle. Accessed 23.05.2026 from https://www.mediaWiki.org/Wiki/Manual:$wgPasswordAttemptThrottle

MediaWiki. (2025s). MediaWiki: Search. Accessed 25.05.2026 from https://www.mediaWiki.org/Wiki/Help:Searching

MediaWiki. (2026a). Extension: CategoryTree. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Extension:CategoryTree

MediaWiki. (2026b). Extension: OATHAuth. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Extension:OATHAuth

MediaWiki. (2026c). Extension: PDFEmbed. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Extension:PDFEmbed

MediaWiki. (2026d). Manual: Hashing Accessed 30.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Hashing

MediaWiki. (2026e). Manual: Installation requirements. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Installation_requirements

MediaWiki. (2026f). Manual: User rights. Accessed 13.05.2026 from https://www.mediaWiki.org/Wiki/Manual:User_rights

MediaWiki. (2026g). Manual: Preventing access. Accessed 1.05.2026 from https://www.mediaWiki.org/Wiki/Manual:Preventing_access

Nieles, M., Dempsey, K., & Pillitteri, V. Y. (2017). An introduction to information security (NIST Special Publication No. 800-12 Rev. 1). National Institute of Standards and Technology. Accessed 13.05.2026 from https://csrc.nist.gov/pubs/sp/800/12/r1/final

OWASP. (2025a). Application Security Verification Standard. OWASP. Accessed 15.05.2026 from https://owasp.org/www-project-application-security-verification-standard/

OWASP. (2025b). HTTP Strict Transport Security cheat sheet. OWASP Cheat Sheet Series. Accessed 23.05.2026 from https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html

OWASP. (2025c). Testing for cookies attributes. OWASP Web Security Testing Guide. Accessed 23.05.2026 from https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes

OWASP. (2025d). Transport Layer Security cheat sheet. OWASP Cheat Sheet Series. Accessed 23.05.2026 from https://cheatsheetseries.owasp.org/cheatsheets/Transport_Layer_Security_Cheat_Sheet.html

Pfleeger, C. P., Pfleeger, S. L., & Coles-Kemp, L. (2024). Security in computing (6th ed.). Addison-Wesley

Sauro, J., & Lewis, J. R. (2016). Quantifying the User Experience: Practical Statistics for User Research (2nd edition.). Elsevier Science & Technology.

Sommerville, I. (2011). Software engineering (9. ed., International ed.). Addison-Wesley.

Vetenskapsrådet. (2017). Good research practice (God forskningssed). Swedish Research Council. https://www.vr.se/english/analysis/reports/our-reports/2017-08-31-good-research-practice.html

Wall, T., & Stokes, P. (2015). Research Methods and Data Gathering. In Research Methods. Bloomsbury Publishing Plc.

Wen, S.-F., & Katt, B. (2023). *A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard.* Computers & Security, 135, Article 103532. Accessed 28.05.2026 from https://www.sciencedirect.com/science/article/pii/S016740482300442X

Wikimedia Foundation. (2025a). ApiAuthManagerHelper.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiAuthManagerHelper_8php_source.html

Wikimedia Foundation. (2025b). ApiBase class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Api_1_1ApiBase.html

Wikimedia Foundation. (2025c). ApiBase.php file reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiBase_8php.html

Wikimedia Foundation. (2025d). ApiChangeAuthenticationData.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiChangeAuthenticationData_8php_source.html

Wikimedia Foundation. (2025e). ApiLogout.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiLogout_8php_source.html

Wikimedia Foundation. (2025f). ApiMain class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Api_1_1ApiMain.html

Wikimedia Foundation. (2025g). ApiMain.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiMain_8php_source.html

Wikimedia Foundation. (2025h). ApiManageTags class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Api_1_1ApiManageTags.html

Wikimedia Foundation. (2025i). ApiQueryUserInfo.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/ApiQueryUserInfo_8php_source.html

Wikimedia Foundation. (2025j). AuthenticationRequest class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Auth_1_1AuthenticationRequest.html

Wikimedia Foundation. (2025k). AuthManager.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/AuthManager_8php_source.html

Wikimedia Foundation. (2025l). Authority interface reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/interfaceMediaWiki_1_1Permissions_1_1Authority.html

Wikimedia Foundation. (2025m). CommandLineInstaller class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classCommandLineInstaller.html

Wikimedia Foundation. (2025n). config-schema.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/config-schema_8php_source.html

Wikimedia Foundation. (2025o). ContentSecurityPolicy class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Request_1_1ContentSecurityPolicy.html

Wikimedia Foundation. (2025p). CookieSessionProvider.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.3/php/CookieSessionProvider_8php_source.html

Wikimedia Foundation. (2025q). Database access. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/database.html

Wikimedia Foundation. (2025r). DatabaseBlockStore.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/DatabaseBlockStore_8php_source.html

Wikimedia Foundation. (2025s). EditPage class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1EditPage_1_1EditPage.html

Wikimedia Foundation. (2025t). Html class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Html_1_1Html.html

Wikimedia Foundation. (2025u). HTMLFormField.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/HTMLFormField_8php_source.html

Wikimedia Foundation. (2025v). Html.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/Html_8php_source.html

Wikimedia Foundation. (2025w). LocalPasswordPrimaryAuthenticationProvider.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/LocalPasswordPrimaryAuthenticationProvider_8php_source.html

Wikimedia Foundation. (2025x). MainConfigSchema.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/MainConfigSchema_8php_source.html

Wikimedia Foundation. (2025y). MediaWiki\Permissions\PermissionStatus class reference. Wikimedia Code Documentation. Accessed 13.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/master/php/classMediaWiki_1_1Permissions_1_1PermissionStatus.html

Wikimedia Foundation. (2025z). MediaWikiServices.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/MediaWikiServices_8php_source.html

Wikimedia Foundation. (2025aa). OutputPage class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Output_1_1OutputPage.html

Wikimedia Foundation. (2025ab). OutputPage.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/OutputPage_8php_source.html

Wikimedia Foundation. (2025ac). PasswordFactory class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Password_1_1PasswordFactory.html

Wikimedia Foundation. (2025ad). PermissionManager class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Permissions_1_1PermissionManager.html

Wikimedia Foundation. (2025ae). PermissionManager.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/PermissionManager_8php_source.html

Wikimedia Foundation. (2025af). RsaJwtCodec.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/RsaJwtCodec_8php_source.html

Wikimedia Foundation. (2025ag). Sanitizer class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Parser_1_1Sanitizer.html

Wikimedia Foundation. (2025ah). Search results for eval. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/search.php?query=eval

Wikimedia Foundation. (2025ai). SessionBackend class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Session_1_1SessionBackend.html

Wikimedia Foundation. (2025aj). SessionBackend.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/SessionBackend_8php_source.html

Wikimedia Foundation. (2025ak). SessionManager.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/SessionManager_8php_source.html

Wikimedia Foundation. (2025al). Session.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/Session_8php_source.html

Wikimedia Foundation. (2025am). SessionProvider.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/SessionProvider_8php_source.html

Wikimedia Foundation. (2025an). Shell command class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Shell_1_1Command.html

Wikimedia Foundation. (2025ao). SiteImporter.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/SiteImporter_8php_source.html

Wikimedia Foundation. (2025ap). Token.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/Token_8php_source.html

Wikimedia Foundation. (2025aq). UploadBase class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classUploadBase.html

Wikimedia Foundation. (2025ar). UploadBase.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/UploadBase_8php_source.html

Wikimedia Foundation. (2025as). UploadVerification class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Upload_1_1UploadVerification.html

Wikimedia Foundation. (2025at). User.php source file. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/User_8php_source.html

Wikimedia Foundation. (2025au). WebRequest class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Request_1_1WebRequest.html

Wikimedia Foundation. (2025av). Xml class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classMediaWiki_1_1Xml_1_1Xml.html

Wikimedia Foundation. (2025aw). XmlTypeCheck class reference. Wikimedia Code Documentation. Accessed 23.05.2026 from https://doc.Wikimedia.org/mediaWiki-core/1.45.0-rc.0/php/classWikimedia_1_1Mime_1_1XmlTypeCheck.html

Wilson, C. (2014). Interview techniques for UX practitioners: a user-centered design method (1st edition). Morgan Kaufmann. https://doi.org/10.1016/C2012-0-06209-6

Wurman, R. S. (2001). Information anxiety 2. Que.

Appendix A: Application Security Verification Standard (ASVS)

Represented here are the level 1 requirements of the Application Security Verification Standard (ASVS) used as a guideline to evaluate whether MediaWiki provides controls necessary for the most basic application-level security expected of an application.

Encoding and Sanitization (OWASP, 2025a):

- **1.2.1;** Verify that output encoding for an HTTP response, HTML document, or XML document is relevant for the context required, such as encoding the relevant characters for HTML elements, HTML attributes, HTML comments, CSS, or HTTP header fields, to avoid changing the message or document structure.

- **1.2.2:** Verify that when dynamically building URLs, untrusted data is encoded according to its context (e.g., URL encoding or base64url encoding for query or path parameters). Ensure that only safe URL protocols are permitted (e.g., disallow javascript: or data:).

- **1.2.3:** Verify that output encoding or escaping is used when dynamically building JavaScript content (including JSON), to avoid changing the message or document structure (to avoid JavaScript and JSON injection).

- **1.2.4:** Verify that data selection or database queries (e.g., SQL, HQL, NoSQL, Cypher) use parameterised queries, ORMs, entity frameworks, or are otherwise protected from SQL Injection and other database injection attacks. This is also relevant when writing stored procedures.

- **1.2.5:** Verify that the application protects against OS command injection and that operating system calls use parameterised OS queries or use contextual command line output encoding.

- **1.3.1:** Verify that all untrusted HTML input from WYSIWYG editors or similar is sanitised using a well-known and secure HTML sanitisation library or framework feature.

- **1.3.2:** Verify that the application avoids the use of eval() or other dynamic code execution features such as Spring Expression Language (SpEL). Where there is no alternative, any user input being included must be sanitised before being executed.

- **1.5.1:** Verify XML parsers use restrictive configurations, disabling unsafe features like external entity resolution to prevent XXE attacks.

Validation and Business Logic (OWASP, 2025a):

- **2.1.1:** Verify that the application's documentation defines input validation rules for how to check the validity of data items against an expected structure. This could be common data formats such as credit card numbers, email addresses, telephone numbers, or it could be an internal data format.

- **2.2.1:** Verify that input is validated to enforce business or functional expectations for that input. This should either use positive validation against an allow list of values, patterns, and ranges, or be based on comparing the input to an expected structure and logical limits according to predefined rules. For L1, this can focus on input which is used to make specific business or security decisions. For L2 and up, this should apply to all input.

- **2.2.2:** Verify that the application is designed to enforce input validation at a trusted service layer. While client-side validation improves usability and should be encouraged, it must not be relied upon as a security control.

- **2.3.1:** Verify that the application will only process business logic flows for the same user in the expected sequential step order and without skipping steps.

Web Frontend Security (OWASP, 2025a):

- **3.2.1**. Verify that security controls are in place to prevent browsers from rendering content or functionality in HTTP responses in an incorrect context (e.g., when an API, a user-uploaded file or other resource is requested directly). Possible controls could include: not serving the content unless HTTP request header fields (such as Sec-Fetch-*) indicate it is the correct context, using the sandbox directive of the Content-Security-Policy header field or using the attachment disposition type in the Content-Disposition header field.

- **3.2.2:** Verify that content intended to be displayed as text, rather than rendered as HTML, is handled using safe rendering functions (such as createTextNode or textContent) to prevent unintended execution of content such as HTML or JavaScript.

- **3.3.1:** Verify that cookies have the 'Secure' attribute set, and if the '__Host-'prefix is not used for the cookie name, the '__Secure-'prefix must be used for the cookie name.

- **3.4.1**: Verify that a Strict-Transport-Security header field is included on all responses to enforce an HTTP Strict Transport Security (HSTS) policy. A maximum age of at least 1 year must be defined, and for L2 and up, the policy must apply to all subdomains as well.

- **3.4.2:** Verify that the Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header field is a fixed value by the application, or if the Origin HTTP request header field value is used, it is validated against an allowlist of trusted origins. When 'Access-Control-Allow-Origin: *'needs to be used, verify that the response does not include any sensitive information.

- **3.5.1:** Verify that, if the application does not rely on the CORS preflight mechanism to prevent disallowed cross-origin requests to use sensitive functionality, these requests are validated to ensure they originate from the application itself. This may be done by using and validating anti-forgery tokens or requiring extra HTTP header fields that are not CORS-safelisted request-header fields. This is to defend against browser-based request forgery attacks, commonly known as cross-site request forgery (CSRF).

- **3.5.2:** Verify that, if the application relies on the CORS preflight mechanism to prevent disallowed cross-origin use of sensitive functionality, it is not possible to call the functionality with a request which does not trigger a CORS-preflight request. This may require checking the values of the 'Origin' and 'Content-Type'request header fields or using an extra header field that is not a CORS-safelisted header-field.

- **3.5.3:** Verify that HTTP requests to sensitive functionality use appropriate HTTP methods such as POST, PUT, PATCH, or DELETE, and not methods defined by the HTTP specification as "safe"such as HEAD, OPTIONS, or GET. Alternatively, strict validation of the Sec-Fetch-* request header fields can be used to ensure that the request did not originate from an inappropriate

cross-origin call, a navigation request, or a resource load (such as an image source) where this is not expected.

Web Service Security (OWASP, 2025a):

- **4.1.1:** Verify that every HTTP response with a message body contains a Content-Type header field that matches the actual content of the response, including the charset parameter to specify safe character encoding (e.g., UTF-8, ISO-8859-1) according to IANA Media Types, such as "text/", "/+xml" and "/xml".

- **4.4.1:** Verify that WebSocket over TLS (WSS) is used for all WebSocket connections.

File Handing (OWASP, 2025a):

- **5.2.1:** Verify that the application will only accept files of a size which it can process without causing a loss of performance or a denial of service attack.

- **5.2.2:** Verify that when the application accepts a file, either on its own or within an archive such as a zip file, it checks if the file extension matches an expected file extension and validates that the contents correspond to the type represented by the extension. This includes, but is not limited to, checking the initial 'magic bytes', performing image re-writing, and using specialised libraries for file content validation. For L1, this can focus just on files which are used to make specific business or security decisions. For L2 and up, this must apply to all files being accepted.

- **5.3.1:** Verify that files uploaded or generated by untrusted input and stored in a public folder, are not executed as server-side program code when accessed directly with an HTTP request.

- **5.3.2:** Verify that when the application creates file paths for file operations, instead of user-submitted filenames, it uses internally generated or trusted data, or if user-submitted filenames or file metadata must be used, strict validation and sanitation must be applied. This is to protect against path traversal, local or remote file inclusion (LFI, RFI), and server-side request forgery (SSRF) attacks.

Authentication (OWASP, 2025a):

- **6.1.1:** Verify that application documentation defines how controls such as rate limiting, anti-automation, and adaptive response, are used to defend against attacks such as credential stuffing and password brute force. The documentation must make clear how these controls are configured and prevent malicious account lockout.

- **6.2.1:** Verify that user set passwords are at least 8 characters in length although a minimum of 15 characters is strongly recommended.

- **6.2.2:** Verify that users can change their password.

- **6.2.3:** Verify that password change functionality requires the user's current and new password.

- **6.2.4:** Verify that passwords submitted during account registration or password change are checked against an available set of, at least, the top 3000 passwords which match the application's password policy, e.g. minimum length.

- **6.2.5:** Verify that passwords of any composition can be used, without rules limiting the type of characters permitted. There must be no requirement for a minimum number of upper or lower case characters, numbers, or special characters.

- **6.2.6:** Verify that password input fields use type=password to mask the entry. Applications may allow the user to temporarily view the entire masked password, or the last typed character of the password.

- **6.2.7:** Verify that "paste"functionality, browser password helpers, and external password managers are permitted.

- **6.2.8:** Verify that the application verifies the user's password exactly as received from the user, without any modifications such as truncation or case transformation.

- **6.3.1:** Verify that controls to prevent attacks such as credential stuffing and password brute force are implemented according to the application's security documentation.

- **6.3.2:** Verify that default user accounts (e.g., "root", "admin", or "sa") are not present in the application or are disabled.

- **6.4.1:** Verify that system generated initial passwords or activation codes are securely randomly generated, follow the existing password policy, and expire after a short period of time or after they are initially used. These initial secrets must not be permitted to become the long term password.

- **6.4.2:** Verify that password hints or knowledge-based authentication (so-called "secret questions") are not present.

Session Management Documentation (OWASP, 2025a):

- **7.2.1:** Verify that the application performs all session token verification using a trusted, backend service.

- **7.2.2:** Verify that the application uses either self-contained or reference tokens that are dynamically generated for session management, i.e. not using static API secrets and keys.

- **7.2.3:** Verify that if reference tokens are used to represent user sessions, they are unique and generated using a cryptographically secure pseudo-random number generator (CSPRNG) and possess at least 128 bits of entropy.

- **7.2.4:** Verify that the application generates a new session token on user authentication, including re-authentication, and terminates the current session token.

- **7.4.1:** Verify that when session termination is triggered (such as logout or expiration), the application disallows any further use of the session. For reference tokens or stateful sessions, this means invalidating the session data at the application backend. Applications using self-contained tokens will need a solution such as maintaining a list of terminated tokens, disallowing tokens produced before a per-user date and time or rotating a per-user signing key.

- **7.4.2:** Verify that the application terminates all active sessions when a user account is disabled or deleted (such as an employee leaving the company).

Authorisation (OWASP, 2025a):

- **8.1.1:** Verify that authorisation documentation defines rules for restricting function-level and data-specific access based on consumer permissions and resource attributes.

- **8.2.1:** Verify that the application ensures that function-level access is restricted to consumers with explicit permissions.

- **8.2.2:** Verify that the application ensures that data-specific access is restricted to consumers with explicit permissions to specific data items to mitigate insecure direct object reference (IDOR) and broken object level authorisation (BOLA).

- **8.3.1:** Verify that the application enforces authorisation rules at a trusted service layer and doesn't rely on controls that an untrusted consumer could manipulate, such as client-side JavaScript.

Self-Contained Tokens (OWASP, 2025a):

- **9.1.1:** Verify that self-contained tokens are validated using their digital signature or MAC to protect against tampering before accepting the token's contents.

- **9.1.2:** Verify that only algorithms on an allowlist can be used to create and verify self-contained tokens, for a given context. The allowlist must include the permitted algorithms, ideally only either symmetric or asymmetric algorithms, and must not include the 'None'algorithm. If both symmetric and asymmetric must be supported, additional controls will be needed to prevent key confusion.

- **9.1.3:** Verify that key material that is used to validate self-contained tokens is from trusted pre-configured sources for the token issuer, preventing attackers from specifying untrusted sources and keys. For JWTs and other JWS structures, headers such as 'jku', 'x5u', and 'jwk'must be validated against an allowlist of trusted sources.

- **9.2.1:** Verify that, if a validity time span is present in the token data, the token and its content are accepted only if the verification time is within this validity time span. For example, for JWTs, the claims 'nbf'and 'exp'must be verified.

OAuth and OIDC Security (OWASP, 2025a):

- **10.4.1:** Verify that the authorisation server validates redirect URIs based on a client-specific allowlist of pre-registered URIs using exact string comparison.

- **10.4.2:** Verify that, if the authorisation server returns the authorisation code in the authorisation response, it can be used only once for a token request. For the second valid request with an authorisation code that has already been used to issue an access token, the authorisation server must reject a token request and revoke any issued tokens related to the authorisation code.

- **10.4.3:** Verify that the authorisation code is short-lived. The maximum lifetime can be up to 10 minutes for L1 and L2 applications and up to 1 minute for L3 applications.

- **10.4.4:** Verify that for a given client, the authorisation server only allows the usage of grants that this client needs to use. Note that the grants 'token'(Implicit flow) and 'password'(Resource Owner Password Credentials flow) must no longer be used.

- **10.4.5:** Verify that the authorisation server mitigates refresh token replay attacks for public clients, preferably using sender-constrained refresh tokens, i.e., Demonstrating Proof of Possession (DPoP) or Certificate-Bound Access Tokens using mutual TLS (mTLS). For L1 and L2 applications, refresh token rotation may be used. If refresh token rotation is used, the authorisation server must invalidate the refresh token after usage, and revoke all refresh tokens for that authorisation if an already used and invalidated refresh token is provided.

Cryptography (OWASP, 2025a):

- **11.3.1:** Verify that insecure block modes (e.g., ECB) and weak padding schemes (e.g., PKCS#1 v1.5) are not used.

- **11.3.2:** Verify that only approved ciphers and modes such as AES with GCM are used.

- **11.4.1:** Verify that only approved hash functions are used for general cryptographic use cases, including digital signatures, HMAC, KDF, and random bit generation. Disallowed hash functions, such as MD5, must not be used for any cryptographic purpose.

Secure Communication (OWASP, 2025a):

- **12.1.1:** Verify that only the latest recommended versions of the TLS protocol are enabled, such as TLS 1.2 and TLS 1.3. The latest version of the TLS protocol must be the preferred option.

- **12.2.1**: Verify that TLS is used for all connectivity between a client and external facing, HTTP-based services, and does not fall back to insecure or unencrypted communications.

- **12.2.2:** Verify that external facing services use publicly trusted TLS certificates.

Configuration (OWASP, 2025a):

- **13.4.1:** Verify that the application is deployed either without any source control metadata, including the. git or .svn folders, or in a way that these folders are inaccessible both externally and to the application itself.

Data protection (OWASP, 2025a):

- **14.2.1:** Verify that sensitive data is only sent to the server in the HTTP message body or header fields, and that the URL and query string do not contain sensitive information, such as an API key or session token.

- **14.3.1:** Verify that authenticated data is cleared from client storage, such as the browser DOM, after the client or session is terminated. The 'Clear-Site-Data' HTTP response header field may be able to help with this but the client-side should also be able to clear up if the server connection is not available when the session is terminated.

Secure Coding and Architecture (OWASP, 2025a):

- **15.1.1:** Verify that application documentation defines risk-based remediation time frames for 3rd party component versions with vulnerabilities and for updating libraries in general, to minimise the risk from these components.

- **15.2.1:** Verify that the application only contains components which have not breached the documented update and remediation time frames.

- **15.3.1:** Verify that the application only returns the required subset of fields from a data object. For example, it should not return an entire data object, as some individual fields should not be accessible to users.

Appendix B: Interview Guide

The following interview guide was used during the conducted interviews.

Activity	Comments	Approximate time
Intro	• Settle in • Present ourselves and the roles we will take in the interview • Explain the purpose of the interview and inform how long it's estimated to take • Explain: that they will be anonymised in the report, they should answer in the level of detail they're comfortable with and they do not need to answer questions if not comfortable. • Ask for consent to record and transcribe	~ 10 min
Structured topics	Topic 1: Background about their current role and project • Interviewee's role, experience, and responsibilities in projects • The influence of security requirements on daily work Topic 2: The current way of working with information and documents • Current practices for managing project information and documents • Use of different systems and solutions across projects • Strengths and weaknesses of existing solutions • Findability and accessibility of information • Consistency of information structure, format, and organisation • Handling of updates, missing information, and tacit knowledge Topic 3: The future information management system • Essential system functionalities • Ease of use and low adoption threshold • Support for structured, up-to-date information • Desired or ideal features beyond current solutions	~ 40 min
Closing dialog	• Inform what the next step for us in the project • Ask if we can email for potential follow-up questions • "Thank you for participating!"	~ 5 min

Appendix C: Stakeholder Needs

The following table includes the stakeholder needs, defined in the Stakeholder Needs and Requirements Definition process.

Acronyms used: CQC = Critical quality characteristic, IA = Information architecture

ID	Need	CQC	Criticality	Category
N:01	Information needs to be accessible to all users	Availability	High	IA
N:02	All users need to be able to upload information	Collaborative, Usability	High	IA
N:03	User-contributed information needs to be modifiable by all users	Collaborative	High	IA
N:04	Project specific governing documents and information needs to be modifiable only by project admins	Integrity	High	IA
N:05	Information needs to be correct and up to date	Accuracy	High	IA
N:06	Need to be able to search	Usability	High	IA
N:07	If an information object exists in more than one place in the system, there should not exist two versions, the information should be consistent	Correctness	High	IA
N:08	Users need to be able to access the information they are looking for quickly	Efficiency	High	IA
N:09	Need to meet the necessary constraints when it comes to information security specific to each project	Security	High	Security
N:10	Need to create a common understanding/knowledgebase specific to the project in question	Usability	Medium	IA
N:11	Need to help members new to a project to understand the regulatory framework of the project	Usability	Medium	IA
N:12	Need standard template for information organisation that can be modified by the project admin	Accuracy	Medium	IA
N:13	Need to be able to organise information in a specific way depending on the project	Customisability	High	IA
N:14	Need overview of how the information-pages are organised in the system	Usability, Efficiency	High	IA
N:15	Need overview of the content inside information pages	Usability, Efficiency	High	IA

N:16	The overview of the content inside information-pages and actual content inside information-pages should be consistent	Correctness	High	IA
N:17	The overview of how the information-pages are organised in the system and should be consistent with how the information-pages actually are organised	Correctness	High	IA
N:18	Information needs to be traceable	Traceability	High	IA
N:19	Needs to be able to handle large amounts of information	Scalability	High	IA
N:20	Needs the information management system to be able to handle daily use	Availability	High	IA
N:21	Ability to reference from one information page to another	Traceability	High	IA
N:22	Enable project admin to carry out information management activities	Usability	Medium	IA
N:23	Project admins should have admin privileges	Usability	High	IA
N:24	Users need to be able to search for one word and find content with that word as well as content with associated words	Usability, Efficiency	High	IA
N:25	Overviews should be zoomable	Usability	Low	IA
N:26	Needs to be visually appealing	Usability	Medium	IA
N:27	Needs to be functional	Usability	High	IA
N:28	Needs to be collaborative, users should be able to influence user-contributed information	Collaborative	High	IA
N:29	Need ability to give feedback on information (and point out mistakes)	Collaborative	Medium	IA
N:30	Information needs to be consistent	Correctness	High	IA
N:31	Need ability to archive information	Correctness	Medium	IA
N:32	Users need ability to delete user-contributed information	Correctness, Collaborative	High	IA
N:33	Project admins need ability to delete governing documents and other information	Correctness	High	IA
N:34	Need to be able to display information to all users	Usability	High	IA
N:35	Tagging information pages to make it easier to find them	Efficiency	Medium	IA
N:36	Need a way to make version management easy	Efficiency	Medium	IA

N:37	To be updated/notified when new information is available	Traceability	Medium	IA
N:38	Facilitate effective training regarding systems which will be used in a project	Usability	Medium	IA
N:39	Project managers need ability to edit how the information-pages are organised during a project	Customisability	High	IA
N:40	Need version control of the organisation of the information in the system	Traceability	High	IA
N:41	Information needs to be unambiguous	Correctness	High	IA
N:42	Information needs to be complete	Correctness	High	IA
N:43	Information needs to be maintained in accordance with integrity requirements	Security	High	Security
N:44	Information needs to be maintained in accordance with privacy requirements	Security	High	Security
N:45	The system needs to implement information security measures general to system-environments where it can be hosted	Security	High	Security

Appendix D: Stakeholder Requirements

The following table includes the complete set of stakeholder requirements, defined in the Stakeholder Needs and Requirements Definition process.

Definitions used in the requirements:

- *Project admin = a class of users, with more privileges in the system.*
- *Project members = a class of users with slightly less privileges in the system.*
- *Users = all authenticated users, in other words project admin and project members*

(See Table 16 in section 5.3.1 Technical Specifications for the full Access Control Policy.)

ID	Requirement	Type	Derived needs	Verification method
ST:01	Users shall be able to search in the system	Functional	N:01, N:06, N:08	User test
ST:02	Users shall be able to search for a word and find content with that word	Functional	N:01, N:06, N:08, N:24	User test
ST:03	Users shall be able to search for one word and find content with associated words	Functional	N:01, N:06, N:08, N:24	User test
ST:04	Users shall be able to upload files to the system.	Functional	N:02, N:22, N:28, N:41, N:42	User test
ST:05	Users shall be able to write information to the system.	Functional	N:02, N:21, N:22, N:28	User test
ST:06	Users shall be able to view all information pages in the system	Functional	N:01, N:08, N:10, N:11, N:34	Functional test
ST:07	Project members shall be able to edit only user contributed information within the system	Functional	N:03, N:05, N:22, N:28, N:30	User test
ST:08	Project members shall be able to delete only user contributed information within the system	Functional	N:05	Functional test
ST:09	Project admin shall be able to edit all types of information within the system	Functional	N:03, N:04, N:05, N:22	Functional test
ST:10	Project admin shall be able to delete all types of information	Functional	N:05, N:34	Functional test
ST:11	Project admin shall be able to decide the editing-access for project members	Functional	N:09, N:45	Functional test
ST:12	The project admin shall be able to manage user access to the system	Functional	N:09, N:45	Functional test
ST:13	Users shall have a clear overview of the information pages in the system	Functional	N:08, N:14, N:18	User test
ST:14	Users shall be able to navigate between information pages	Functional	N:08, N:14, N:19	User test

ST:15	Users should rate the time required to find specific information at 3 or lower on a Likert scale (Agree/Disagree format).	Non-functional	N:08	User test
ST:16	Users shall be able to influence the information content in the system	Functional	N:28	User test
ST:17	The overview of how the information pages are organised should be up to date	Functional	N:08, N:17	Inspection
ST:18	Project admins may be able to access a standard template for information organisation in the system	Functional	N:12	Functional test
ST:19	Users shall have an overview of information content	Functional	N:15	User test
ST:20	User satisfaction with the system's visual appeal should score at least 3 on a Likert scale (Agree/Disagree format).	Non-functional	N:26	User test
ST:21	When project members disagree with accuracy of governing documentation project members shall be able to suggest edits of that governing documentation	Functional	N:29, N:43	Functional test
ST:22	When project members suggest edits of governing documentation it shall not be edited	Functional	N:43, N:04	Functional test
ST:23	When project members have suggested edits of governing information project admins shall be notified	Functional	N:43, N:28	Functional test
ST:24	Users shall be able to archive information pages	Functional	N:05, N:31	Functional test
ST:25	Users shall have an overview of information organisation	Functional	N:13, N:07, N:36	User test

Appendix E: System Requirements

The following table includes the complete set of system requirements, defined in the System Requirements Definition process.

ID	Requirement	Type	Derived from	Verification method
SY:01	The system shall have a search-function	Functional	ST:01, ST:15	Inspection
SY:02	When the search-function in the system is used, results should appear	Functional	ST:01, ST:15	Functional test
SY:03	When a user is searching for a string, the system shall show information objects with this string	Functional	ST:02, ST:15	Functional test
SY:04	When a user is writing in the input field of the search component the search function shall dynamically display proposed searches beginning with the string the user is writing in the input field.	Functional	ST:03, ST:15	Functional test
SY:05	The system shall implement access control allowing project admins and project members different capabilities	Functional	N:45	Functional test
SY:06	The system shall display search results within 0.5 seconds after a user submits a search query if there are any results for that search	Non-functional	ST:15	Functional test
SY:07	The system shall allow users to upload files	Functional	ST:04, ST:16	Functional test
SY:08	The system shall support uploading of documents in PDF-format	Functional	ST:04	Functional test
SY:09	The system shall support uploading of documents in word-format	Functional	ST:04	Functional test
SY:10	The system shall allow users to create new information pages into the system	Functional	ST:05, ST:16	Functional test
SY:11	The system shall allow users to add meta information to pages with uploaded documents	Functional	ST:07, ST:16	Functional test
SY:12	The system shall implement access control that allow only project admins to edit all information within the system.	Functional	ST:09, N:43, N:45	Functional test
SY:13	The system shall implement access control allowing the project members to edit only the user contributed information within the system.	Functional	ST:07, ST:16	Functional test
SY:14	The system shall implement access control that allow only project admins to delete all types of information	Functional	ST:10	Functional test

SY:15	The system shall implement access control that allow project members to delete only user contributed information	Functional	ST:08, ST:16, N:43, N:45	Functional test
SY:16	The system shall enable navigation between information pages	Functional	ST:14	Functional test
SY:17	The system shall have a visual overview of the organisation of the information pages	Functional	ST:14	Functional test
SY:18	The system shall require users to sign in	Functional	N:45	Functional test
SY:19	The system shall require authenticated user identification before granting users access to the system	Functional	N:45	Functional test
SY:20	Information pages shall be linked to each other if the information content in the pages reference to each other	Functional	ST:13	Inspection
SY:21	Reference-links between information pages should be updated automatically by the system	Functional	ST:13, N:5	Functional test
SY:22	The system shall have a favourite-function	Functional	ST:14	Inspection
SY:23	When new information pages are added, the overview of how the information pages are organised should update automatically	Functional	ST:17	Functional test
SY:24	The system shall have an archiving-function	Functional	ST:24	Inspection
SY:25	Information shall be able to be restored from the archive, by the project admin	Functional	ST:24	Functional test
SY:26	The system shall store records of changes made to information	Functional	ST:25	Functional test
SY:27	The system shall contain an overview of information content	Functional	ST:25	Inspection
SY:28	The system should be built with open-source software	Functional	Delimitation from case context	Inspection
SY:29	The system should store passwords as salted hashes	Functional	Identified through personal communication	Inspection
SY:30	The system should implement multifactor authentication	Functional	Identified through personal communication	Inspection

Appendix F: Functional Viewpoint

The following viewpoint defines the conventions for how to create and interpret the functional view.

Viewpoint overview:

A functional viewpoint focuses on the architectural elements that realize the system's functionality, in other words the system's logic (Eles & Cripps, 2010, Appendix B). The functional elements are described in terms of their responsibilities, interfaces, and interactions between the functional elements. Through these descriptions, the functional view demonstrates how the system will manage to perform its intended functionality (Rozanski & Woods, 2005).

Concerns and "anti-concerns":

This viewpoint addresses functionalities the system needs to provide to address the concern regarding its functionality. Since the required system is an information system, the main stakeholder concerns revolve around functionality relating to information findability and effectiveness of finding information.

Concerns regarding computer and information security will not be addressed by this viewpoint but by a separate security viewpoint.

Typical stakeholders:

Users of the system as well as all other stakeholders as the functionalities included here aim to fulfil the purpose of the system (Rozanski & Woods, 2005).

Model kinds:

Functional Structure Model:

A functional structure model typically includes functional elements, interfaces, connectors, and external entities. The main functional elements of a functional structural model are as follows (Rozanski & Woods, 2005):

- *Functional elements:* Functional elements are well-defined parts of a system that are responsible for certain functionalities of the system. Furthermore, a functional element exposes clearly defined interfaces allowing it to interact with other elements.

  Underlying infrastructure supporting functional elements should not itself be modelled as a functional element or be included in the functional view. However, if that infrastructure, independently of other functional elements, performs a task important to the functionality of the system, it could be included as a functional element.

- *Interfaces:* An interface defines how other elements may access the functionality of a functional element by specifying inputs, outputs, and meaning of each operation as well as how the interaction between elements is carried out. Not to be confused with the user interface (UI).

- *Connectors:* A connector describes how elements are linked and how they communicate with each other. Depending on the level of detail needed, connectors can be modelled by simply noting that elements are connected without further specification, or they can be more explicitly modelled interaction mechanisms.

- *External entities*: An external entity is a system or component outside of the system boundary that interacts with the system. The system connects to external entities through interfaces, and external entities appear at the far end of an interface in the functional model. External entities are typically identified through the system context.

Modelling technique:

In line with ISO et al. (2017), ISO 12207, architectural entities of the SoI were identified as well as relationships between the entities that address key stakeholder concerns and critical system requirements. Since this was done from the functional viewpoint, the architectural entities consisted of functional elements of the SoI. The established requirements for the SoI as well as key drivers and stakeholder concerns were reviewed to identify high-level functional elements with functions needed for the system to fulfil its intended purpose. With the same method, interfaces and interactions with external entities contributing to the functionality of the system were identified.

Notation:

The functional structure was modelled and visualized through a boxes-and-lines diagram. A boxes-and-lines diagram allows for development and application of a standard notation suitable for the stakeholders or audience it is meant to communicate the system architecture to. Although the choice of notation is flexible in boxes-and-lines diagrams, compared to modelling techniques with predefined notation, the standard notations chosen should still be clearly defined and consistently applied (Rozanski & Woods, 2005).

Standard notation for functional structure boxes-and-lines diagram:

- *Functional elements:* represented by squares with rounded corners, and the name of the functional element written inside it. The name chosen should be representative of the function(s) that the element is responsible for. Further specifications of the responsibilities of the functional element and what requirements, concerns or key drivers were used to of the functional element should be included in text or table below the visual representation.

- *Interfaces:* represented by an empty circle with a suitable name giving insight into the meaning of interactions between elements, written next to the empty circle. Further specifications of inputs, outputs, and how the interaction is carried out should be included in text or table below the visual representation.

- *Connectors:* represented by arrow from the functional element/external entity invoking communication through the interface by sending an output, to the functional element/entity taking it as input, while passing through the empty circle signifying the interface.

- *System boundary:* represented by a dashed square separating the internal, functional elements from the external entities by placing the functional elements inside the square and the external entities outside the square.

- *External entities:* represented by squares with rounded corners, and the name of the external entity written inside it, as well as the text “External entity”. The name chosen should be representative of the function(s) that the external entity is responsible for. Further specifications of the functions meant to be provided by the external entity should be included in text or tables below the visual representation. External entities should be connected to functional elements through interfaces and

should appear at the far end of an interface in the diagram. To clearly signify that an entity is an external entity, it should be placed outside the system boundary.

Modelling tools:

The online design tool Canva.

Correspondence rules:

Functional model "cross view" correspondence rules:

- Each functional element in the functional view should be mapped to one or more physical elements in the physical view

Operations on views:

The view is created by using the models of this viewpoint and further explaining how relevant stakeholder concerns are addressed and which constraints apply to elements.

In the Design Definition process, this view will act as a reference when further specifying how architectural elements will be designed.

Appendix G: Deployment Viewpoint

The following viewpoint defines the conventions for how to create and interpret the deployment view.

Viewpoint overview:

The deployment viewpoint describes the environment in which the system is deployed. Including what hardware environment the system needs, each elements requirements regarding technical environment, and mapping the software elements to hardware components (Rozanski & Woods, 2005). A deployment viewpoint on the system relevant in situations where the system may be deployed into several different environments, in which case essential characteristics of the deployment environment need to be clearly illustrated (Rozanski & Woods, 2005).

Concerns and "anti-concerns":

This viewpoint will address what types of hardware are required to support the system. In addition, required quantity and specifications of hardware are provided when appropriate. If requirements exist around using specific third-party software or physical constraints those are also addressed by the deployment viewpoint (Rozanski & Woods, 2005). The main stakeholder concerns for this system revolved around information findability and effectiveness of finding information. Thus, this viewpoint focused primarily on identifying what is required of a deployment environment, for the system to support the functionalities specified by the functional viewpoint.

Concerns regarding computer security will not be addressed by this viewpoint but by a separate security viewpoint.

Typical stakeholders:

System administrators, developers, testers, communicators, and assessors (Rozanski & Woods, 2005).

Model kinds:

For the deployment viewpoint, a runtime platform model, network model, and technology dependency model is recommended to be used. However, the network can be adequately described by including it in the runtime platform model instead of creating a separate network model in cases such as this where network requirements are less complex. The technology dependency model is meant to state requirements on hardware and software needed for the system to run as intended (Rozanski & Woods, 2005). The choice of open-source software, made in the Design Definition process, and choices made in the Development stage of the systems life cycle process, are expected to largely determine the dependencies. Thus, only a runtime platform model was constructed at this stage.

<u>Runtime Platform Model</u>

A runtime platform model defines the required hardware nodes, which ones should be connected to each other via interfaces, and which hardware nodes hosts which software elements. The main elements of the model include (Rozanski & Woods, 2005):

- *Processing nodes:* Are used to represent the computers, such as servers, in the system to show what computing resources are needed by the system.

- *Client nodes:* A client node defines client hardware. If special needs for user interaction or presentation exist, this should be specified. However, there is often less control over client hardware in which case only types and quantities are required.

- *Online storage hardware:* Describes the storage that the system uses while running. This includes how much storage is required, potentially speed of accessing data, parts it is divided into, what it is used for, and whether data is processed close to its storage. Whether storage is part of a processing node or separate, dedicated storage node(s) should be specified as it can impact the physical setup required for the system to run as expected.

- *Offline storage hardware:* Information intensive systems often require offline storage, archives, in addition to online storage to allow backup of information located online. Type, capacity, speed, and location of offline storage hardware need to be sufficient for archive retrieve online data within an acceptable timeframe and should be included in the model.

- *Network links:* Represents essential connections between defined hardware nodes.
- *Other hardware components:* If there is a need to consider special hardware for network security, user authentication, special interfaces to other systems or special processing, these should be defined as well.

- *Runtime element-to-node mapping:* Maps the functional elements of the system to the processing nodes where they execute.

Modelling technique:

In line with ISO et al. (2017), ISO 12207, architectural entities of the SoI were identified as well as relationships between the entities that address key stakeholder concerns and critical system requirements. Since this was done from the deployment viewpoint, the architectural entities consisted of hardware nodes as well as network links The established requirements for the SoI as well as key drivers and stakeholder concerns were reviewed to identify high-level hardware nodes needed for the systems functional elements, and consequently the system, to be able to fulfil its intended purpose. With the same method, interfaces and interactions with external entities contributing to the functionality of the system were identified.

Notation:

As in the functional structural model of the functional view, the runtime platform model was modelled and visualized through a boxes-and-lines diagram. For the runtime platform boxes-and.-lines diagram the following standard notation was developed and applied:

- *Processing nodes:* Square with the text “Processing node:” followed by a representative name of the processing node

- *Client nodes:* Square with text “Client node:” followed by a representative name of the client node.

- *Online storage* hardware: Square with text “Online storage hardware:” followed by a representative name of the online storage hardware.

- *Offline storage hardware:* Square with text “Offline storage hardware:” followed by a representative name of the Offline storage hardware.

- *Other hardware components:* Square with text “Other hardware component:” followed by a representative name of the hardware component.

- *Runtime element-to-node mapping:* Demonstrated by placing the functional components from the functional structural model within the boarder of the relevant component in the runtime platform model.

- *Network links:* Lines connecting nodes
- *System boundary:* represented by a dashed square separating the internal, functional elements from the external entities by placing the functional elements inside the square and the external entities outside the square.

Modelling tools:

The online design tool Canva.

Correspondence rules:

Runtime Platform model "cross view" correspondence rules:

- Each element in the runtime platform model should map to at least one element in the functional structural model of the functional view

Operations on views:

The view is created by using the models of this viewpoint and further explaining how relevant stakeholder concerns are addressed and which constraints apply to elements.

In the Design Definition process, this view will act as a reference when further specifying how architectural elements will be designed.

Appendix H: Alignment of MediaWiki with Architecture Description

The following tables specifies the details of how well the default MediaWiki open-source code aligns with the responsibilities in the architecture description derived from needs and requirements. Furthermore, the tables specifies if any additional configuration, extension or technology can be added on top of MediaWiki to fulfil responsibilities specified in the architecture description. Prerequisites for MediaWiki installation are assumed to be fulfilled and elements that are prerequisites such as database are only mentioned if they carry much or all responsibility for a functionality. Table H1 addresses all requirements but the chosen ASVS requirements which are addressed in table H2.

Table H1: Alignment of responsibilities of the architecture description with the default MediaWiki open-source code. Where MediaWiki default does not fulfil responsibilities which changes need to be made are stated.

User Interaction Manager	
Responsibility	**Requires**
Interpret all user requests	**Default MediaWiki:** Yes. [a]
Route requests to the right functional element.	**Default MediaWiki:** Yes. [a]
Return system-response to user.	**Default MediaWiki:** Yes. [a]
Provide structure for user interaction, including overview and navigation elements as well as structures content (UI-related functionality)	**Default MediaWiki:** Partially [a] Overview partially exists. Through the page "Special:All Pages" which lists all pages on the Wiki and "Special:Categories" that provides overview of existing categories [a,b] **Extension:** CategoryTree Gives overview of pages that are placed into categories. Can be placed in created pages [c]
Automatically update overview of pages when new pages are added.	**Default MediaWiki:** Yes. [a] Overview in "Special: All Pages" and "Special: Categories" automatically updates. [b] Overview in CategoryTree automatically updates too. [c]
Search Engine	
Responsibility	**Requires**
When the search-function in the system is used, results should appear	**Default MediaWiki:** Yes. [d]
When a user is searching for a string, the system shall show information objects with this string	**Default MediaWiki:** Yes. [d]
When a user is writing in the input field of the search component the search function shall dynamically display proposed searches beginning with the string the user is writing in the input field.	**Default MediaWiki:** Yes. [d]
The system shall display search results within 0.5 seconds after a user submits a search query if there are any results for that search.	**Default MediaWiki:** No such guarantees can be made. [d] **Additional technologies:** Search response time is affected by on database performance, hardware resources, network capacity and other factors that may vary between deployment environments. [r]

Content Editor	
Responsibility	**Requires**
Enable users to create pages.	**Default MediaWiki:** Yes. [e]
Enable users to edit pages.	**Default MediaWiki:** Yes. [e]
Enable users to delete pages.	**Default MediaWiki:** No. [e] Only the user group "sysop" can remove pages from being accessible in the Wiki, but no pages can be completely deleted. [e] **Code configuration:** Only MediaWiki can be reconfigured to allow additional user groups to remove pages from being accessible in the Wiki. [e]
The system shall allow users to upload files. Including files in PDF- and Word-format.	**Default MediaWiki:** Conditionally. [f] Files can be uploaded but PDF- and Word-format are not in the allowed-list by default. [f] **Code configuration:** MediaWiki can be reconfigured to allow PDF- and Word-format [f]
The system shall allow users to add meta information to pages with uploaded documents	**Default MediaWiki:** Partially. [g] When files are uploaded, they get their own description page metadata can be added to this page [g] Embedding Images into pages is possible but no other filetypes are embedded. [g] **Extension:** PDFEmbed Allows PDF-files to be embedded into pages using Wikitext. [h]
Enable users to go back to earlier versions of pages.	**Default MediaWiki:** Yes. [i]
Enable users to add pages to a category.	**Default MediaWiki:** Yes. [j]
Navigation Manager	
Responsibility	**Requires**
Resolve navigation requests	**Default MediaWiki:** Yes. [a]
Maintain navigation paths	**Default MediaWiki:** Yes. [k]
Keep links to referenced pages up to date	**Default MediaWiki:** Yes. [a]
Disable links if referenced page is deleted	**Default MediaWiki:** Partially [k] Links are still clickable but marked red to signify they do not work. When clicked users will reach a page explaining the previously linked page does not exist [k]
Storage	
Responsibility	**Requires**
Store content of current pages	**Default MediaWiki:** Conditionally. **Code configuration:** Configuration of settings and directories. [l] **Additional technology:** Server filesystem MediaWiki requires a database to store content. However, files and images are stored directly in server filesystem a which requires additional configuration to be secure. [l]
Store content of deleted pages	**Default MediaWiki:** Yes. [a]
Store content from old versions of edited pages	**Default MediaWiki:** Yes. [a]
Store passwords as salted hashes	**Default MediaWiki:** Yes. [m]
Authorisation Manager	
Responsibility	**Requires**
Requires users to login to access Wiki content	**Default MediaWiki:** Conditionally

	If MediaWiki is set to be a so-called private Wiki. [n] **Code configuration:** Requires Wiki to be set to private. A private Wiki will require users to login or to create an account. If $wgGroupPermissions['*']['createaccount'] = false; only selected user groups can create an account otherwise anyone could access the private Wiki by simply creating an account. Additional permissions related to reading and editing should also be set to false to disable users that are not authorised to accessing the Wiki. Important to note is that MediaWiki warns the above-mentioned changes in code might not be enough to block access in all ways and might still show private information. For example, files may remain publicly accessible through URL this can partially be solved by activating existing mediaWiki code but might also need server-side configuration. Thus, MediaWiki recommends thorough testing and customization. [n] **Additional technology:** Configure server so uploaded files are not directly accessible from the web. [n]
Handles multi-factor authentication.	**Default MediaWiki:** Conditionally SessionManager check if logged in session. If not redirects to AuthManager that handles login (no multifactor authentication available). If authenticated through login the SessionManager maintains authenticated session. [o] **Extension:** OATHAuth Provides support for two-factor authentication on login [p] **Additional technology:** Either GMP Either GMP or BCMath is required for OATHAuth to work. In addition, configuration of the database ad often an external user-side authentication method such as an authentication app is required. [p]
Makes authorisation decisions for actions in the system based on access control rules	**Default MediaWiki:** Yes Implements RBAC-type access control and checks permissions through authentication and then sessions are used. [q]
Enable the user group admin to create pages that cannot be edited or deleted by users in the user group “user”	**Default MediaWiki:** Partially. [e] Users in the group admin can create pages that other users cannot edit but only the user group “sysop” can remove pages from being accessible in the Wiki, but no pages can be completely and permanently deleted through the web interface, this can only be done through command-line deletions. [e] **Code configuration:** MediaWiki can be reconfigured to allow additional user groups, such as admin, to remove pages from being accessible in the Wiki. [e]

Notes: a: (MediaWiki, 2025l) b: (MediaWiki, 2025i) c: (MediaWiki, 2026a) d: (MediaWiki, 2025s) e: (MediaWiki, 2026j) f: (MediaWiki, 2025p) g: (MediaWiki, 2025g) h: (MediaWiki, 2026c) i: (MediaWiki, 2025h) j: (MediaWiki, 2025e) k: (MediaWiki, 2025f) l: (MediaWiki, 2025k) m: (MediaWiki, 2026d) n: (MediaWiki, 2026g) o: (MediaWiki, 2025n) p: MediaWiki, 2026b) q: (Wikimedia Foundation, 2025y) r: (Cambazoglu & Baeza-Yates, 2016)

The following table addresses if there are any controls in the default MediaWiki open-source code that could, if implemented correctly across the application, fulfill the ASVS requirements. A full code review would be needed to assert if the default configuration of MediaWiki fulfills these requirements. The focus is instead to understand if there is functionality in the code that could be implemented to fulfill the ASVS requirements. Functionality that suggests appropriate control is lifted but for all requirements a full code review would be needed to confirm. In cases where certain functionality of default MediaWiki does not fulfill the ASVS requirements, it is stated what additional configuration, stable official MediaWiki extensions, or additional technology is needed.

Table H2: Controls present in MediaWiki that could enable fulfilment of ASVS requirements of the architecture description with the default MediaWiki open-source code. Where MediaWiki default does not fulfill responsibilities which additional changes need to be made are stated.

ASVS	Requirement
1.2.1	**Default MediaWiki:** Yes, if implemented correctly and consistently. In the Html.php file the Html class is defined to include methods such as Html::element() which replace special characters with encoded representations. Methods such as Html::expandAttributes() escape attribute values using htmlspecialchars(), helping to prevent browsers form interpreting user-controlled input as HTML or executable code. [a] Similar helper functions such as Xml::element() and Xml::expandAttributes() for XML escaping also exists through the XML class. [b] However, output functions like Html::rawElement() [c] and OutputPage::addHTML() [d] also exist, and can be considered safe only if content first is confirmed to be trusted. [a] All instances where Html::rawElement() are used should be controlled to first verify only trusted content is used as input to these functions
1.2.2	**Default MediaWiki:** Yes, if implemented correctly and configured suitably. The MediaWiki parser calls the function cleanUrl() in Sanitizer.php to make sure only safe URLs are permitted checks against predefined permitted protocols. [e] MediaWiki implements URL protocol restrictions through the $wgUrlProtocols whitelist in the core configuration schema (config-schema.php). Only predefined protocols are permitted, and these do not include potentially dangerous protocols such as javascript: or data:. This help prevent user-controlled URLs from being interpreted as executable code in browser contexts. [f] **Code Configuration:** Set $wgUrlProtocols to contain appropriate protocols.
1.2.3	**Default MediaWiki:** Yes, if implemented correctly and consistently. MediaWiki provides centralised JavaScript and JSON encoding through Html::encodeJsVar(), which in turn calls FormatJson::encode() to encode user-controlled input before inserting it in JavaScript contexts to help prevent input from being read as JavaScript. [a] OutputPage::addInlineScript calls Html:inlineScript. [d] if Html::inlineScript() detects script tags were present a warning is logged and input content is replaced with error message. [a]
1.2.4	**Default MediaWiki:** Yes, if implemented consistently and new code follows conventions. By default, MediaWiki implements centralised database functions and query builder APIs for database operations which automatically handle tasks like escaping user-controlled input counteracting changing structure of SQL queried and reducing risk of SQL injection. MediaWiki also provide a function for developers to write their own SQL-queries. [g] Since MediaWiki at this stage has not been edited in any way this should not be a concern but should always be confirmed only predefined queries are used.
1.2.5	**Default MediaWiki:** Yes, if implemented consistently and added extensions follow conventions. MediaWiki uses centralised shell command handling trough shell.php that build operating system commands through controlled API, Shell::command() and Shell::escape() help reduce risk of user input being interpreted as commands. [h]

1.3.1	**Default MediaWiki:** Needs further investigation, may need additional sanitisation library $allow in Sanitizer.php only predefined HTML allowed ex. p, div, b, img allowed but only with restricted set of permitted attributes [f] these are used as reference when checking validateAtributes(). [e]. Core clearly sanitises HTML-like input. However, MediaWiki uses own code instead of well-known and secure HTMLsanitisation library framework feature. **Additional technology:** Well-known and secure HTML sanitisation library.
1.3.2	**Default MediaWiki:** Needs further investigation Source code was searched for "eval", it appeared 197 times from search and all instances were not checked due to time constraints but for example eval() is used in eval.php, benchmarkEval.php these are restricted to command-line use and certain directories where mainly used by maintenance scripts indicating user input is not handled by eval(). [i]
1.5.1	**Default MediaWiki:** Yes, if controls exist where appropriate Includes reusable XML security functionality in XmlTypeCheck.php. [j] Howerver, ex. SiteImporter.php does not use the reusable functionality from XmlTypCheck.php but instead libxml_disable_entity_loader(true). Indicating no centralised functionality is used consistently, meaning no guarantees can be made about XML parsing being safe. [k]
2.1.1	**Default MediaWiki:** Yes, if checks are consistently done against defined parameters. MediaWiki generates MainConfigSchema.php through CommandLineInstaller a maintenance script run on installation. [l].MainConfigSchema.php contains default settings for MediaWiki including valid forms of different paramaters for example required length of passwords. [m]
2.2.1	**Default MediaWiki:** Yes, if implemented consistently In code reviewed parameters from MainConfigSchema.php are often used as references when checking received input
2.2.2	**Default MediaWiki:** Yes, if consistently implemented. Reviewed parts of the code consistently apply server-side validation before processing user-controlled input which strongly points to this being applied application-wide.
2.3.1	**Default MediaWiki:** Yes, if implemented consistently The EditPage class in MediWiki uses edit() - " The edit form is self-submitting, so that when things like preview and edit conflicts occur, we get the same form back with the extra stuff added. Only when the final submission is made and all is well do we actually save and redirect to the newly-edited page. Also contains editing states such as $save and $isConflict indicating controlled workflow and state management. [p]
3.2.1	**Default MediaWiki:** Yes, if implemented consistently. MediaWikis requests have a children class ContentSecurityPolicy Class which handles sending Content-Security-Policy headers, SendRestrictiveHeader() output a very restrictive CSP header to disallow all active content, and GetMediaHeader() gets the CSP header for a specific file. [q]
3.2.2	**Default MediaWiki:** Yes, if implemented consistently. Html.php has function Html::element() that escapes content before putting it in an element so it becomes text rather than executable code. But also has Html::rawElement() which does not escape content. [c] However, $wgRawHtml is by default set to false. [m]
3.3.1	**Default MediaWiki:** No, need reconfiguration and additional controls. ManualConfigSchema.php sets $wgCookieSecure = 'detect'; as default value in MediaWiki 1.45 if HTTPS-only it should be set to $wgCookieSecure = 'true'; to avoid cookie theft. Sites using reverse proxies, load balancing or other methods converting HTTPS requests into HTTP need to set header for detection to work correctly ($wgVarONXPF). [r] __Secure tells browser to only send cookie if HTTPS communication to avoid leaking sensitive info. __Host demands __Secure, __Domain that excludes domain attributes to protect against attacker harvesting cookies, and __Path to specify valid path for cookies in order to protect server. [s] Not specific __Secure, own definition, No specific __Host-cookie **Code configuration:** Set $wgCookieSecure = true; and **i**mplement required __Secure and __Host-cookie

	Additional technology: TLS configured server.
3.4.1	**Default MediaWiki:** Yes, if implemented consistently and configured correctly. MediaWiki has a parameter intended to help ensure secure HTTPS communication, $wgForceHTPS which is set to false by default, this should be changed to true. [m] $wgForceHTTPS supports HTTPS but is no guarantee for being implemented properly, server configuration is the more effective tool for ensuring this. **Code configuration:** Set $wgForceHTTPS = true; **Additional technology:** Server configuration to enforce HTTPS trough TLS
3.4.2	**Default MediaWiki:** Yes, if implemented consistently and configured for deployment environment in question. ApiMain::handleCORS() does origin validation to see if external websites are trusted domains checking against predefined $wgCrossSiteAJAXdomains and $wgCrossSiteAJAXdomainExceptions. And also checks if the request header is trusted against predefined $AllowedCorsHeader. [v] Needs to be checked if this is implemented everywhere in MediaWiki and if allowed domains and request headers are appropriate for the deployment context. **Code configuration: Set** $wgCrossSiteAJAXdomains, $wgCrossSiteAJAXdomainExceptions and $AllowedCorsHeader to appropriate values
3.5.1	**Default MediaWiki:** Yes, if implemented consistently. ApiBase::needsToken() supports core CSRF tokens, [w] ApiLogout::execute() requires POST as well as CSRF token, [x] also ApiChangeAuthenticationData::execute() demands CSRF token. [y]
3.5.2	**Default MediaWiki:** Yes, if implemented consistently. As seen in 3.4.2 and 3.5.1 MediaWiki mainly uses CSRF tokens + POST + and CORS origin validation instead of CORS preflight as main control. So the control suggested by 3.5.2. becomes less relevant.
3.5.3	**Default MediaWiki:** Yes, if implemented consistently. ApiProtect.php APi documentation states modules must override isWriteMode() to return true if they use “non-safe” operations for HTTP and must require POST over unsafe, also API actions demand CSRF token. [z]
4.1.1	**Default MediaWiki:** Yes, if implemented consistently. MainConfigSchema sets $wgMimeType to text/html by default. [m] Then OutputPage.php appends so Content-type: text/html; charset=UTF-8 also for HTML. [aa]
4.4.1	**Default MediaWiki:** Yes, if no WebSocket used otherwise additional configuration is needed. No evidence of MediaWiki using WebSocket was found but further code review is needed to confirm. **Additional Technology:** Server configuration to enforce TLS.
5.2.1	**Default MediaWiki:** Yes, if implemented consistently and with configuration appropriate for deployment environment. $wgMaxUploadSize is by default set to 100MiB. [m] This is then used in UploadBase.php which returns an error message saying the file is to large if it is larger than 100MiB. [ab] **Code Configuration:** Set $wgMaxUploadSize to appropriate value for deployment environment
5.2.2	**Default MediaWiki:** Yes, if implemented consistently and with configuration appropriate for deployment environment. MainConfigSchema.php gives predefined default allowed file extensions in parameter $wgFileExtensions and predefined prohibited in $wgProhibitedFileExtensions. [m] UploadVerification::verifyFile() rejects extension mismatches check for script content and does media-handler validation. [ac] **Code Configuration:** Set $wgFileExtensions and $wgProhibitedFileExtensions to values appropriate for the deployment environment.
5.3.1 server	**Default MediaWiki:** Yes, with additional configuration of server and allowed file extensions configured to be appropriate for deployment environment. MainConfihSchema.php sets $wgProhibitedFileExtensions = 'html', 'htm', 'js', 'jsb', 'mhtml', 'mht', 'xhtml', 'xht', 'php', 'phtml', 'php3', 'php4', 'php5', 'phps', 'phar' 'shtml', 'jhtml', 'pl', 'py', 'cgi', 'exe', 'scr', 'dll', 'msi', 'vbs', 'bat', 'com', 'pif', 'cmd', 'vxd', 'cpl', 'xml'] . And excludes dangerous MIME types. [m]

	Code Configuration: Set $wgProhibitedFileExtensions to values appropriate for the deployment environment. **Additional technology:** Appropriate configuration of file directory on server before deployment, access control for file directory.
5.3.2	**Default MediaWiki:** Yes, if implemented consistently. UploadBase::getTitle() rejects invalid names of files and strips prohibited filename characters as well as normalises result using Title::makeTitleSafe(). UploadBase::validateName() has a function that sets accepted name after validation. [ae] MainConfigSchema.php disables upload using URL through $wgAllowCopyUploads=false, $wgCopyUploadsFromSpecialUplod=false, and $wgCopyUploadDomains=[] as defaults. [m]
6.1.1 Deployment	**Default MediaWiki:** Yes $wgPasswordAttemptThrottle facilitates rate limiting, limits password attempts per IP and username to 5 attempts/300seconds and short-term and 150 attempts/ 48 hours. $wgAccountCreationThrottle can set limits account creation per IP. Regular account creation limits are set to 0 by default. This should be changed to fulfil requirement. $wgRateLimits set rate limits for other actions like editing and uploading per user or IP. [m] **Code configuration:** $wgMainCacheType must be set to something other than the default CACHE_NONE for all above parameters to be implemented. [af]
6.2.1	**Default MediaWiki:** Yes MainConfigSchema.php sets default values for parameters on setup and installation. Default MinimalPasswoedLenght is 8 for normal users and 10 for privileged groups. Should be changed to fulfil the stronger recommendation of 15 chars. [m]
6.2.2	**Default MediaWiki:** Yes MediaWiki facilitates password changes. [y]
6.2.3	**Default MediaWiki:** No, needs reconfiguration. ApiChangeAuthenticationData.php treats password change as securitySensitiveOperation() sensitive. y Which in turn checks authentication has happened within limit of $wgReauthenticateTime. a[g] $wgReauthenticateTime is set to 1 hour. [m] Meaning recent authentication, within one hour, is demanded to be able to change password. Meaning users are not always required to enter old password when setting new. **Code Configuration:** Set $wgReauthenticateTime = 0 or implement additional control.
6.2.4	**Default MediaWiki:** Yes. $wgPasswordInCommonList is by default set to 'PasswordNotInCommonList' => ['value' => true, 'suggestChangeOnLogin' => true]. [m]
6.2.5	**Default MediaWiki:** Needs further investigation No default requirement was found concerning uppercase, lowercase, number or special characters. [m]
6.2.6	**Default MediaWiki:** Yes AuthenticationRequest::getFieldInfo() defines input fields for passwords as type = ”password” and marks them as sensitive. This is used by AuthManager APIs and UIs. [ah] **Code configuration:** Set wgMainCacheType to something other than CACHE_NONE. [af]
6.2.7	**Default MediaWiki:** Needs further investigation No evidence found either for or against in MediaWiki\Auth\AuthenticationRequest Class Reference. a[h] **Code configuration:** Set wgMainCacheType to something other than CACHE_NONE. [af]
6.2.8	**Default MediaWiki:** Yes $wgLegacyEncoding is set to false by default meaning exact handling is expected . [m] Furter evidence in LocalPasswordPrimaryAuthenticationProvider.php suggests this requirement is upheld. [ai] **Code configuration:** Set wgMainCacheType to something other than CACHE_NONE. [af]

6.3.1	**Default MediaWiki:** Yes, with configuration $wgPasswordAttemptThrottle facilitates rate limiting, limits password attempts per IP and username to 5 attempts/300seconds and short-term and 150 attempts/ 48 hours. $wgAccountCreationThrottle can set limits account creation per IP. Regular account creation limits are set to 0 by default. This should be changed to fulfil requirement.$wgRateLimits set rate limits for other actions like editing and uploading per user or IP. [m] But $wgMainCacheType must be set to something other than the default CACHE_NONE. [af] **Code configuration:** Set wgMainCacheType to something other than CACHE_NONE and wgPasswordAttemptThrottle as well as $wgAccountCreationThrottle to values suitable for the deployment environment.
6.3.2	**Default MediaWiki:** Yes, if configured appropriately before deployment. **Configuration:** Make sure default user accounts are not present in the application or disabled before deployment.
6.4.1	**Default MediaWiki:** Yes, if implemented consistently. $wgNewPasswordExpiry is by default set to 7 days. [m] PasswordFactory.php generates random passwords of length 10 or more. [aj]
6.4.2	**Default MediaWiki:** Yes. MainConfigSchema.php does not show any default parameters created for this. [m]
7.2.1	**Default MediaWiki:** Yes, if implemented consistently. CookieSessionProvider.php is backend [ak] as well as notes for sessions stating they are for backend use only, suggests this is done backend. However, needs to be confirmed by checking all frontend code.
7.2.2	**Default MediaWiki:** Yes, if implemented consistently. getid() used in sessionBackend.php to verify session. [al] (see 7.2.2 below how it is generated)

7.2.3	**Default MediaWiki:** Yes, if implemented consistently.
	SessionManager.php has function public function generateSessionId() {
	$id = \Wikimedia\base_convert(\MWCryptRand::generateHex(40), 16, 32, 32);
	// Cache non-existence to avoid a later fetch
	$info = new SessionInfo(SessionInfo::MIN_PRIORITY, ['id' => $id, 'idIsSafe' => true]);
	$this->sessionStore->set($info, false, 0, BagOStuff::WRITE_CACHE_ONLY);
	return $id;
	}. [am]
	Which generates new random id 160 bits of entropy, uniqueness presumed by randomness

7.2.4	**Default MediaWiki:** Yes. When logging in whether active session or not new session is created. [ao]
7.4.1	**Default MediaWiki:** Yes, if implemented consistently. DoLogout in user.php triggers $session->unpersit in Session.php. [ap] unpersit in Session.php which deletes session. And then marks session as not persistent and saves it and triggers delete on browser-side session. Alsio has timer on session, so it is deleted after some time. [am]
7.4.2	**Default MediaWiki:** Yes, with additional configuration or extension. MediaWiki 1.45 default does not have functionality allowing deletion of users invalidateSessionsForUSer($user) in SessionProvider.php invalidates existing sessions for a user [ar] invalidateSessionsForUSer($user) is called from DataBlockStore.php when a user is blocked. a[q] If $wgBlockDisablesLogin is set to true users, when blocked, are also logged out and cannot login again. [as] User groups with the $wgGroupPermissions['usergroup']['block']=true; can block other users.[at] **Extension:** Accounts can be removed from MediaWiki but requires extension UserMerge and a longer sequence of actions from the system user. This extension allows users with usermerge permission to merge one Wiki user's account with another Wiki user's account. Following the merge the one user's account merged to the other user's account can be deleted. However, this does not delete user records from the database, for this command-line access is needed [bc]

8.1.1	**Default MediaWiki:** Yes, if documentation is created and implemented for specific deployment. PermissionManager.php has permission checks ex. UserCan($action, User $user, LinkTarget $page, $rigor = self::RIGOR_SECURE) as well as a getPermissionStatus() function. Indicating use of RBAC-type access control.[au] However, documentation for authorisation rules should be created.
8.2.1	**Default MediaWiki:** Yes, if implemented consistently Functions like checkActionPermissions (), checkPageRestrictions(), suggests permissions are checked before allowing actions. [av]
8.2.2	**Default MediaWiki:** Yes, if implemented consistently Functions authorisedRead(), isAllowed(), and isRegistered(), in authority.php are referenced by usercan() from permissionsmanager.php. [aw] Should control this is done as expected by checking especially sensitive functions, and resources
8.3.1	**Default MediaWiki:** Yes, if implemented consistently. In mediaWikiServices.php function getPermissionManager. [ax] Also "One of PermissionManager::RIGOR_ constants: RIGOR_QUICK : does cheap permission checks from replica DBs (usable for GUI creation) RIGOR_FULL : does cheap and expensive checks possibly from a replica DB RIGOR_SECURE : does cheap and expensive checks, using the primary DB as needed" Suggests at least RIGOR_SECURE checks server-side. [av]
9.1.1	**Default MediaWiki:** Yes, if implemented consistently. $wgUseSessionCookieJwt enabled provideSessionInfo() calls verifyJwtCookie() and invalid JWT returns null. verifyJwtCookie parses the JWT before claiming this. [ak] Depends on extension JWAuth 2.1.0. [ay]
9.1.2	**Default MediaWiki:** Yes, if implemented consistently. $signingAlgorithm = new Rsa\Sha256 and validates using SignedWith ($signingAlgorithm, $privateKey), indication of 'None' algorithm not found. [az] Depends on extension JWAuth 2.1.0. [ay]
9.1.3	**Default MediaWiki:** Yes, if implemented consistently. No evidence of headers like jku, x5u, jwk but further investigation on key generation is needed to establish they are Depends on extension JWAuth 2.1.0. [ay]
9.2.1	**Default MediaWiki:** Yes, if implemented consistently. RsaJwtCodec.php sets LooseValidA.[az] and time for expiry seems to depend on provider. [an] This should be investigated further to determine
10.4.1	**Default MediaWiki:** Yes, with added extension. **Extension:** Needs OAuth extension and some parameter configuration.[ba] Client-specific URI validation. Callback URL stored callbackUrl/oarc_callback_url. This is then compared to redirect_uri with function ClientEntity::getRedirectUri().[bb]
10.4.2	**Default MediaWiki:** Yes, with added extension. **Extension:** OAuth. auth_code_id, revokeAuthCode(), isAuthCodeRevoked() shows one time use but could not find evidence each code used only once. [bb]
10.4.3	**Default MediaWiki:** Yes, with added extension. **Extension:** OAuth. PT10M and expire_time is checked when tokens exchanged, fulfilled requirements for L1. However 0Auth2GrantExpirationInterval controls access token lifetime but not evidence for authorisation code lifetime. [bb]
10.4.4	**Default MediaWiki:** Yes with added extension. **Extension:** OAuth.oauth2GrantTypes/oarc_oauth2_allowed_grants and ClientEntity::isGrantAllowed(). No exposure of implicit or password grant types. However, need to analyse further to deduce if 0Auth2EnabledGrantTypes alone is complete backend for all client-registration paths. [bb]
10.4.5	**Default MediaWiki:** Yes, with configuration of server **Extension:** OAuth. A valid request revokes old refresh token and issues new ones.. [Bb] **Additional technology:** Server with DpoP or mTLS certificate required
11.3.1	**Default MediaWiki:** Yes, if implemented consistently

	getEncryptionAlgorithm() in session.php prefers using aes-256-ctr but if that is not available it will fallback on aes-256-cbc as encryption algorithm. And to encrypt session secrets HMAC-SHA256 is used also uses hashing for integrity. [bd] Needs to check all code to ensure
11.3.2	**Default MediaWiki:** Needs further investigation. No evidence was found that AES-CGM is used in MediaWiki. Those mentioned in 11.3.1 are used but need to be reviewed properly to see whether their use in context they are used in is approved according to current official guidelines.
11.4.1	**Default MediaWiki:** No Token.php uses HMAC-MD5 for token generation, even though this is stronger tain just MD5 it is still relying on MD5 for cryptographic use. [be]
12.1.1	**Default MediaWiki:** Yes, with server configuration. This cannot be configured via MediaWiki source code since TLS protocols are negotiated by the deployed service such as the web server**.** [bf] **Additional technology:** Thus, it is the responsibility of system administrator to before deployment verify server hosting the application also enforces TLS 1.2 or TLS 1.3 protocol.
12.2.1	**Default MediaWiki:** Yes, if configured and implemented consistently and with additional technology. MediaWiki generates MainConfigSchema.php through CommandLineInstaller a maintenance script run on installation.MainConfigSchema.php contains default settings for MediaWiki. [l] Included here is the parameter $wgForceHTTPS which is by default set to false. By setting it to true HTTP requests should be redirected to HTTPS requests and set the secure-flag on cookies to help facilitate secure communication. But this gives no guarantee and is not an implementation considered secure standalone. [t] **Code configuration**: Set $wgForceHTTPS = true; and $wgCookieSecure. **Additional technology:** To ensure all communications use TLS should be implemented and this needs to be configured on server before deployment. [u]
12.2.2	**Default MediaWiki:** Yes, with additional technology. **Additional technology:** TLS certificates are stored and held by servers and is not part of application code. Thus, this requirement needs to be verified before deployment of the application. [bf]
13.4.1	**Default MediaWiki:** No, needs additional actions before deployment MediaWiki does not have automated feature for this. It becomes the responsibility of system administrator to guarantee this before deployment. Recommendations on how to handle source control metadata provided by MediaWikis manual on security should be used as a guideline. [bg]
14.2.1	**Default MediaWiki:** Yes, if implemented consistently Main access point in MedeiaWiki for API queries is api.php API queries are then handled by apiBase.php which is parent of all other API files meaning they inherit the rules set by apiBase.php. Whenever needsToken() is not false mustBePosted() returns true meaning tokens are put in the body of a request through POST. The token parameter is also marked as sensitive. [bh] ApiMain.php then enforces these rules, if a token-using module does not use POST it will produce error and if token appears in the URL the request is rejected. [bi]
14.3.1	**Default MediaWiki:** Yes, if implemented consistently User::doLogout clears cookies and server-side session state but no cleanup of non-cookie browser storage was found. [ap]
15.1.1	**Default MediaWiki:** No, this requirement is part of maintenance life cycle stage and will not be considered in this thesis Documentation on how quickly vulnerable third-party libraries must be fixed or updated do not exist in MediaWiki code or documentation. This is highly dependent on the context in which the Wiki will be deployed and should be provided by system administrators before deployment.
15.2.1	**Default MediaWiki:** No, this requirement is part of maintenance life cycle stage and will not be considered in this thesis

	This is not a functionality provided by the application. This requirement is something that the maintenance team need to continuously need to work with during the utilisation stage.
15.3.1	**Default MediaWiki:** Yes, if implemented consistently. ApiBase.php is a common template for MediaWiki API files and lets each API modules define what parameters are allowed and validate those parameters. [bh] This is implemented in ex. ApiQueryUserInfo.php but a full code review of all files of the same type would be needed to conclude [bj] this requirement is fulfilled.

Note: a:(Wikimedia Foundation, 2025v), b:(Wikimedia Foundation, 2025av), c:(Wikimedia Foundation, 2025t), d:(Wikimedia Foundation, 2025aa), e:(Wikimedia Foundation, 2025ag), f:(Wikimedia Foundation, 2025n), g:(Wikimedia Foundation, 2025q), h:(Wikimedia Foundation, 2025an), i:(Wikimedia Foundation, 2025ah), j:(Wikimedia Foundation, 2025aw), k:(Wikimedia Foundation, 2025ao), l:(Wikimedia Foundation, 2025m), m:(Wikimedia Foundation, 2025x), n:(Wikimedia Foundation, 2025au), o:(Wikimedia Foundation, 2025u), p:(Wikimedia Foundation, 2025s), q:(Wikimedia Foundation, 2025o), r:(MediaWiki, 2025o), s:(OWASP, 2025c), t:(MediaWiki, 2025q), u:(OWASP, 2025b), v:(Wikimedia Foundation, 2025f), w:(Wikimedia Foundation, 2025b), x:(Wikimedia Foundation, 2025e), y:(Wikimedia Foundation, 2025d), z:(Wikimedia Foundation, 2025h), aa:(Wikimedia Foundation, 2025ab), ab:(Wikimedia Foundation, 2025ar), ac:(Wikimedia Foundation, 2025as), ad:(MediaWiki, 2025k), ae:(Wikimedia Foundation, 2025aq), af:(MediaWiki, 2025r), ag:(Wikimedia Foundation, 2025a), ah:(Wikimedia Foundation, 2025j), ai:(Wikimedia Foundation, 2025w), aj:(Wikimedia Foundation, 2025ac), ak:(Wikimedia Foundation, 2025p), al:(Wikimedia Foundation, 2025ai), am:(Wikimedia Foundation, 2025aj), an:(Wikimedia Foundation, 2025ak), ao:(Wikimedia Foundation, 2025k), ap:(Wikimedia Foundation, 2025at), aq:(Wikimedia Foundation, 2025am), ar:(Wikimedia Foundation, 2025r), as:(MediaWiki, 2025j), at:(MediaWiki, 2026f), au:(Wikimedia Foundation, 2025ae), av:(Wikimedia Foundation, 2025ad), aw:(Wikimedia Foundation, 2025l), ax:(Wikimedia Foundation, 2025z), ay:(MediaWiki, 2025b), az:(Wikimedia Foundation, 2025af), ba:(MediaWiki, 2025c), bb:(MediaWiki, 2025a), bc: (MediaWiki, 2025d)
, bd:(Wikimedia Foundation, 2025al), be:(Wikimedia Foundation, 2025ap), bf:(OWASP, 2025d), bg:(MediaWiki, 2025m), bh:(Wikimedia Foundation, 2025c), bi:(Wikimedia Foundation, 2025g), bj:(Wikimedia Foundation, 2025i)

Appendix I: Test Plan and Task-Based Scenarios

The following test plan was used during the conducted user testing sessions. The task-based scenarios were the foundation of the testing.

Test plan

Objectives

- Collect data to verify requirements linked to usability
- Collect feedback from users regarding the information management system
- Identifying any problems users might encounter while using the system
- Receive feedback regarding what improvements could be made to the system.

Method

- Task-based scenarios
- Think-aloud
- Post-test questionnaire, SUS

Questions

After completed task:

- "What was your experience solving this task in the system?"

After completed user test:

- "What was your overall experience using this system?"
- "Do you have any other comments or suggestions?"

Task-based scenarios

Note: The user should start from the home page for all scenarios.

1. Finding Information

Scenario 1A

"You've just joined a project team who is working with Systems Engineering, and they have started the process of verification and validation. You are familiar with these things but would need to read more about the two processes. Use the system to find information about verification and validation."

Task 1A completed when: The user has located both articles about verification and validation.

Scenario 1B *(If the user navigated to validation and verification through the search function)*

"The project also includes work on quality assurance. Use the system to find information about quality assurance without using the search function. "

Task 1B completed when: The user has found the article on quality assurance without using the system's search function.

Scenario 1C *(If the user navigated to validation and verification without using the search function)*

"The project also includes work on quality assurance. Use the system's search function to find information about quality assurance."

Task 1C completed when: The user has found the article on quality assurance by using the system's search function.

2. Creating New Content

Scenario 2A

The project team you have joined has requested you to add information regarding an internal process, so other team members can read it later. Create a new page in the system with information about the process. Here is the text you should add to the system, please try to match the formatting. *(Provide a printed copy of the process below)*

> [Onboarding new team members
> **Purpose**
> This page describes the basic steps for onboarding new team members to the project.
>
> **Steps**
>
> 1. Make sure the new member has access to all relevant tools.
> 2. Share links to project documentation.
> 3. Assign a point of contact for questions.
>
> **Notes**
>
> *The onboarding process may vary depending on the role.*]

Task 2A completed when: The user has created the page with the given information and used at least one of the built-in functions in the system to match the style, such as title, numbered list, bold.

Scenario 2B

The process you just described also has a nice picture visualising it. Add this picture to the page. The picture is called *NewMember.jpg* and can be found in the Desktop folder in the computer.

Task 2B completed when:

The user has added the picture to the page.

Scenario 2C

Since this is an important process, you want the other project members to be able to find it easily. The standard way your team does this is by using categories. Assign the page to the category called *Projekt X* and verify that it's been done successfully.

Task 2C completed when:

The user has assigned the category to the page, and the user is confident that it was done successfully.

3. Editing Existing Content

Scenario 3

"Since you have joined the project, called Project X, the information about the number of project members on the page titled *Project X Information* is out of date (there are now 8 members in the project instead of 7). Please find this page and update the text on it to reflect the correct number.

Task 3 completed when:

The user has updated the information about project members on the page called *Project X Information*. This includes navigating to the *Project X Information* page and locating where the outdated information is on the page.

Appendix J: System Usability Scale (SUS)

The System Usability Scale questionnaire used after the conducted user testing. The first one is the original one, developed by John Brooke at Digital Equipment Corporation. The second one is the questionnaire translated to Swedish with two additional statements, which is the version used in the user testing in this thesis project.

SUS - Standard version

		Strongly disagree				Strongly agree
No	Statement	1	2	3	4	5
1	I think that I would like to use this system frequently.	☐	☐	☐	☐	☐
2	I found the system unnecessarily complex.	☐	☐	☐	☐	☐
3	I thought the system was easy to use.	☐	☐	☐	☐	☐
4	I think that I would need the support of a Technical person to be able to use this system.	☐	☐	☐	☐	☐
5	I found the various functions in the system were well integrated.	☐	☐	☐	☐	☐
6	I thought there was too much inconsistency in this system.	☐	☐	☐	☐	☐
7	I would imagine that most people would learn to use this system very quickly.	☐	☐	☐	☐	☐
8	I found the system very awkward to use.	☐	☐	☐	☐	☐
9	I felt very confident using the system.	☐	☐	☐	☐	☐
10	I needed to learn a lot of things before I could get going with this system.	☐	☐	☐	☐	☐

SUS - Translated version

		Instämmer inte alls				Instämmer helt
Nr	Påstående	1	2	3	4	5
1	Jag tror att jag skulle vilja använda detta system ofta.	☐	☐	☐	☐	☐
2	Jag uppfattade systemet som onödigt komplext.	☐	☐	☐	☐	☐
3	Jag upplevde att systemet var lätt att använda.	☐	☐	☐	☐	☐
4	Jag tror att jag skullebehöva hjälp från någon med teknisk kunskap för att kunna använda systemet.	☐	☐	☐	☐	☐
5	Jag upplevde att de olika funktionerna i systemet var väl integrerade.	☐	☐	☐	☐	☐
6	Jag tyckte att systemet innehöll för många inkonsekvenser.	☐	☐	☐	☐	☐
7	Jag föreställer mig att de flesta personer skulle lära sig att använda detta system väldigt snabbt.	☐	☐	☐	☐	☐
8	Jag upplevde systemet som krångligt att använda.	☐	☐	☐	☐	☐
9	Jag kände mig väldigt säker när jag använde systemet.	☐	☐	☐	☐	☐
10	Jag behövde lära mig många saker innan jag kunde komma igång med systemet.	☐	☐	☐	☐	☐
11	Jag tycker systemet var visuellt tilltalande	☐	☐	☐	☐	☐
12	Jag upplevde att det tog lång tid innan jag hittade de sidorna jag sökte efter i systemet	☐	☐	☐	☐	☐