ÇANKAYA UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

CENG 407

PROJECT REPORT

CSEK Requirement Management System

Members

KIVILCIM IŞIK 202011006

SARPER ERBAR 202011001

ERAY EMİR 202011016

CAN METE BOZAR 202011052

INTRODUCTION

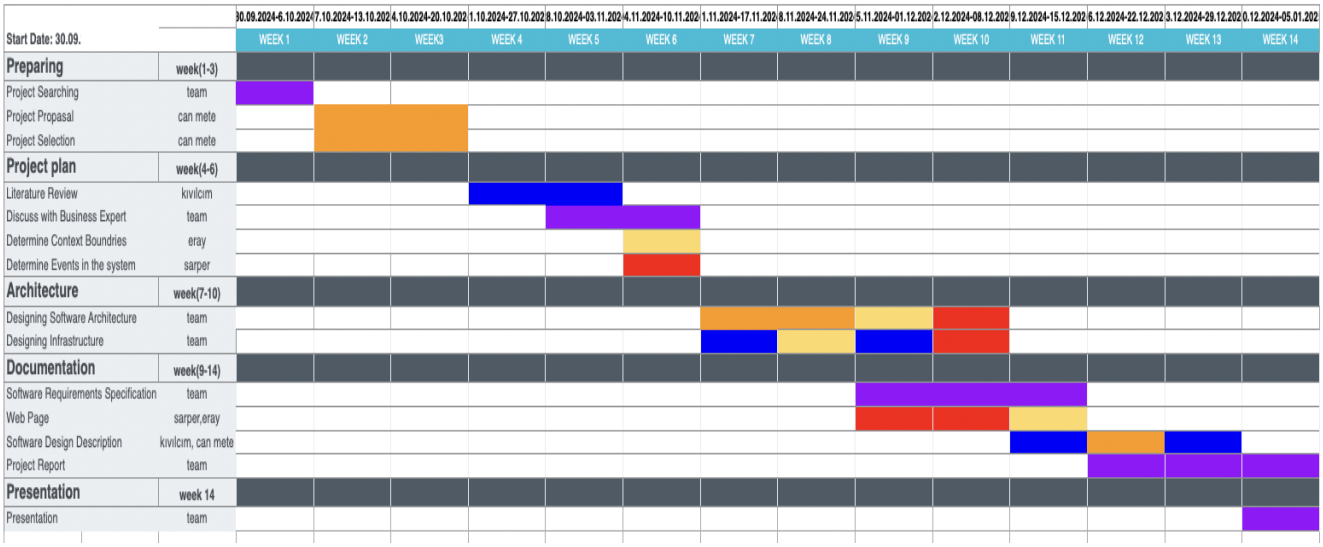
This report aims to examine in detail the development process of the CSEK Requirements Management System and the technologies used. Requirements management is essential for software projects in various industries. The requirements management tools currently used are not local and have high license costs. In this context, our designed system provides a local, accessible, and cost-effective alternative.

The project aimed to localize IBM Doors and create a CSEK Requirements management system integrated with two-way traceability, change management, and reporting features. In addition, the system's primary goals are security, scalability, and a user-friendly interface. The system, which provides accessibility with its web-based structure, can be easily used through a modern browser.

On the other hand, the system, which is built with microservice architecture, provides a structure that is open to future development and improvements. In the future, the system will be able to handle the integration of artificial intelligence, various reporting formats, and user testing.

In conclusion, the CSEK Requirements Management System provides a local, low-cost solution that can be used in software projects, effectively addressing the requirements management process. It is intended to be an improved requirements management tool that can be used in various industries in the future.

PROJECT PLAN



LITERATURE REVIEW

Abstract

Project success in software and systems engineering is largely dependent on efficient requirements management. A number of alternative tools have surfaced as a result of cost, complexity, or particular project requirements, however IBM DOORS has been a well-known tool for tracking, evaluating, and managing complicated requirements. The structure and functionality of IBM DOORS are examined in this paper, along with its reporting features, requirement trace models, and change management integration. A comparison of alternatives like Primavera, Microsoft Project, and specialized tools like RAT and QuARS is also carried out in order to evaluate their advantages and disadvantages in terms of enabling requirements engineering. Additionally, to improve efficiency and accuracy in requirement analysis, the paper suggests an AI-based method that combines Artificial Neural Networks and Natural Language Processing to automate the conversion of user needs into system specifications. Even though handling complicated linguistic structures can be challenging, the AI-based approach has the potential to greatly enhance the software development process' demand analysis step.

Introduction

Requirements management is a foundational discipline in software and systems engineering, ensuring that all project stakeholders' needs are effectively captured, documented, and tracked throughout the lifecycle of a project. IBM DOORS has long been a leading solution for managing complex requirements, offering robust features such as requirement traceability, version control, and automated reporting. However, due to its high cost and complexity, many organizations seek alternative tools to meet their specific project needs. In this report, we explore the features of IBM DOORS and examine several alternatives, focusing on their strengths and limitations in various project contexts. Additionally, we introduce an innovative approach that integrates Artificial Intelligence to automate the transformation of user requirements into system specifications, highlighting the potential benefits and challenges associated with this technology. By combining insights from industry-standard tools and AI techniques, this report aims to provide a comprehensive understanding of the current landscape in requirements management and offer new avenues for enhancing the efficiency and accuracy of requirement analysis.

Navigating Complexity: The Essential Role of Requirements Management in Modern Projects

Requirements management is a important and critical discipline in software and systems engineering, offering a structured approach for documenting, tracking, and adapting requirements across a project’s lifecycle. Requirements management is essential for ensuring that all project stakeholders’ needs are met, reducing the risk of misalignment or project scope issues that can arise when requirements are unclear or inconsistent. In order to reduce risks, eliminate scope creep, and enhance project outcomes, effective requirements management promotes traceability, upholds consistency, and encourages strong documentation.

The role of requirements management becomes significant in large and complex projects, where maintaining control over numerous, interrelated requirements is essential. As a result, requirements management systems have become extremely effective supporters, enabling teams to approach complicated requirements methodically and guaranteeing coordination amongst distant development teams. According to Kühn, Hoffmann, and Weber, requirements management tools provide significant features that facilitate and assist requirements management procedures, meeting a number of essential requirements:

- **Requirements Structuring and Traceability:** Linking requirements to other project artifacts, such design documents, testing specifications, and implementation records, is a crucial feature of requirements management solutions. Project teams can track changes through each phase, comprehend and manage dependencies, and guarantee consistency throughout development phases thanks to this traceability. In order to improve visibility and lower the possibility of discrepancies, the tools provide bidirectional traceability, which is essential for determining the source of each requirement and its influence on later phases. (Kühn et al., 2004).
- **Version Control and Change Management:** Requirements management solutions help with version control, which makes it simpler to monitor changes over time because updates are frequently needed, particularly in agile or iterative development contexts. Kühn claim that these tools keep thorough change histories that differentiate between significant and small alterations in addition to tracking changes at the level of individual requirements. The level of documentation supports a strong audit trail and helps prevent the loss of important information by enabling stakeholders to go back to earlier versions if needed. Change requests can be handled methodically thanks to integrated workflows, which also encourage accountability and offer real-time status information.
- **Document Generation and Reporting:** Requirements management tools simplify documentation by automatically generating reports, specifications, and other essential documents from the requirements. Document consistency and dependability are ensured by this automation, which also saves time and lowers human mistake. According to Kühn, document generation is especially useful in sectors where thorough documentation is required for regulatory compliance or as a component of agreements with outside parties like suppliers in the automotive industry.
- **Scalability and Adaptability:** Requirements management tools are designed to be highly scalable, catering to the diverse and growing needs of large projects. These tools

can manage vast amounts of data, and their adaptability allows them to be customized to fit specific project requirements, processes, and organizational needs. Kühn note that requirements management software may be set up to support domain-specific needs, which is especially advantageous for sectors like aerospace and automotive that have unique project workflows or specialized regulatory standards.

- **Enhanced Collaboration and User Accessibility:** Collaboration is fundamental to successful requirements management, particularly in globally distributed teams. Requirements management software includes features that enhance teamwork, such as user-specific views and centralized databases, and allows team members to work on various project components simultaneously while remaining consistent. Additionally, some solutions provide web-based access, which improves accessibility and lowers administrative overhead by enabling infrequent users or outside contributors to interact with requirements without requiring full software installations. (Kühn et al., 2004).

The requirements management process has been revolutionized by the incorporation of requirements management tools into project management workflows. These technologies offer a simplified, automated way to handle complicated requirements, preserve uniformity, and encourage teamwork. Integration features that enable requirements management technologies to work in unison with other development tools, such configuration management and testing software, are becoming more and more important as these tools advance. This integration supports end-to-end traceability and ensures that each requirement is continuously linked to its corresponding artifacts throughout the product lifecycle.

Research indicates that by decreasing errors, guaranteeing requirements consistency, and promoting adherence to legal requirements, requirements management technologies, when used properly, can greatly improve project outcomes. These solutions are now essential in industries like aerospace and automotive, where big project teams and frequent changes call for a more structured approach to requirements management. In order to better support complicated project needs, requirements management solutions will continue to be developed with an emphasis on accessibility, scalability, and integration with artificial intelligence for predictive analytics.

Overview of IBM DOORS Features and Structure

IBM DOORS (Dynamic Object-Oriented Requirements System) is powerful software designed to manage and track requirements in large-scale projects. It helps organize requirements in a clear project and folder structure. DOORS offers a requirement trace model that includes modules, linking, and specific folder, module, and linking models to ensure that requirements are connected and easy to trace. In addition, its reporting and data export features allow project teams to analyze and share important information efficiently. Together, these features make DOORS a flexible, traceable, and effective tool for requirement management.

• 1. Project and Folder Structure

IBM DOORS projects and folders serve as containers for information, typically containing formal modules and link modules. Folders are hierarchical (parent and child folders) and are listed alphabetically by default. In smaller projects, placing all requirement modules in one folder may be sufficient.

In some projects, the requirements traceability hierarchy may differ from the physical product structure or bill of materials, often resembling a document structure more closely. If requirements are traditionally managed in a document-based model, this structure may be replicated in DOORS, though complex document structures may not easily translate to DOORS' object-oriented database.

To meet traceability and reporting needs, the structure must be modeled and tested. For this, the ReqMAPS team uses Object Role Modeling (ORM) methodology to model project and

folder structures, attributes, views, and link sets. Each sub-folder in the requirements folder structure contains information at the same level within the linking model.

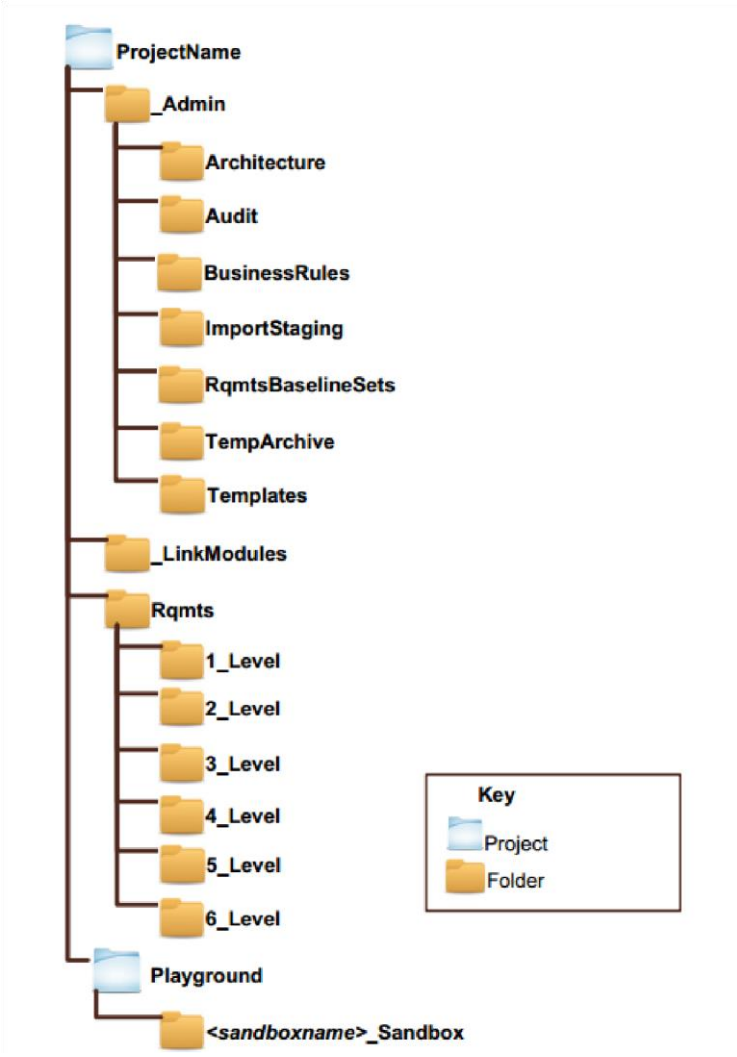


Figure 1: The Standard Folder Structure

- ## 2. Requirement Trace Model

2.1. **Modules:** Whether requirements are created directly in DOORS or imported from other files, a module/linking model should be established to support linking, traceability, and reporting. Creating requirements in DOORS with a defined information structure helps clarify which modules should be linked and ensures requirements are built to support relationships, such as satisfies relationships. Reviewing all requirement information and relationships within a model also simplifies validating the linking model.

2.2. **Linking:** In the traceability hierarchy, requirements are allocated downward and traced upward. For traceability reports to function correctly, levels should not be skipped. The same link set is reused across levels to enable full traceability. The direction of linking is essential because linking data is stored in the source module; thus, the source module requires editing permissions, while read-only access is enough for the target module. Different types of relationships, such as between requirement modules and modules like Qualification or Document Reference, can use link sets in opposite directions. All link sets should be created in DOORS before any linking; otherwise, DOORS will create a default link module, which should be removed using an audit script after confirming no links were created with it.

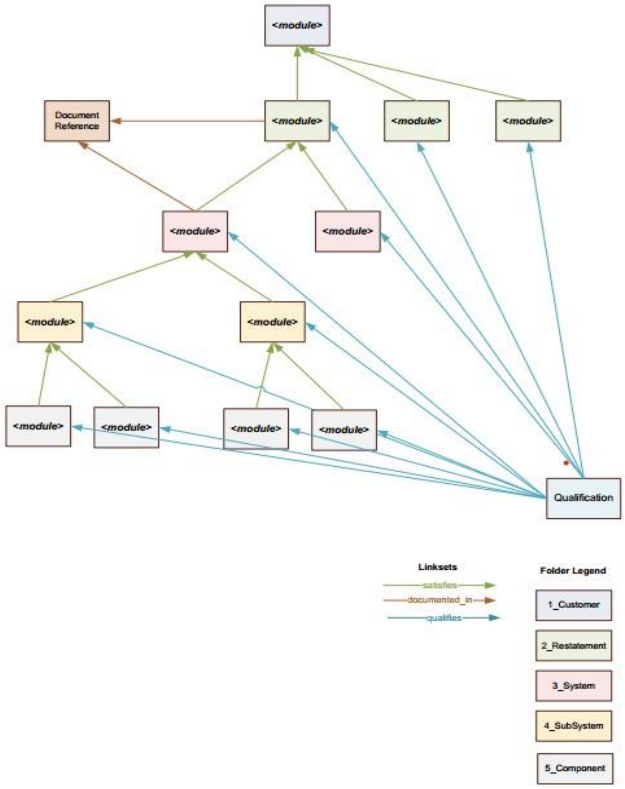


Figure 2: Linking Model Example

2.3. **Folder, Module, and Linking Models:** Two general examples of folder, module, and link set models are provided. The 1_Level folder might contain customer requirements, while the 2_Level folder could hold system requirements. To maintain

trace integrity, link sets should never skip levels. The examples illustrate traditional vertical structures and flattened structures using the satisfies linking model, suitable for cases where the document set does not fit traditional traceability.

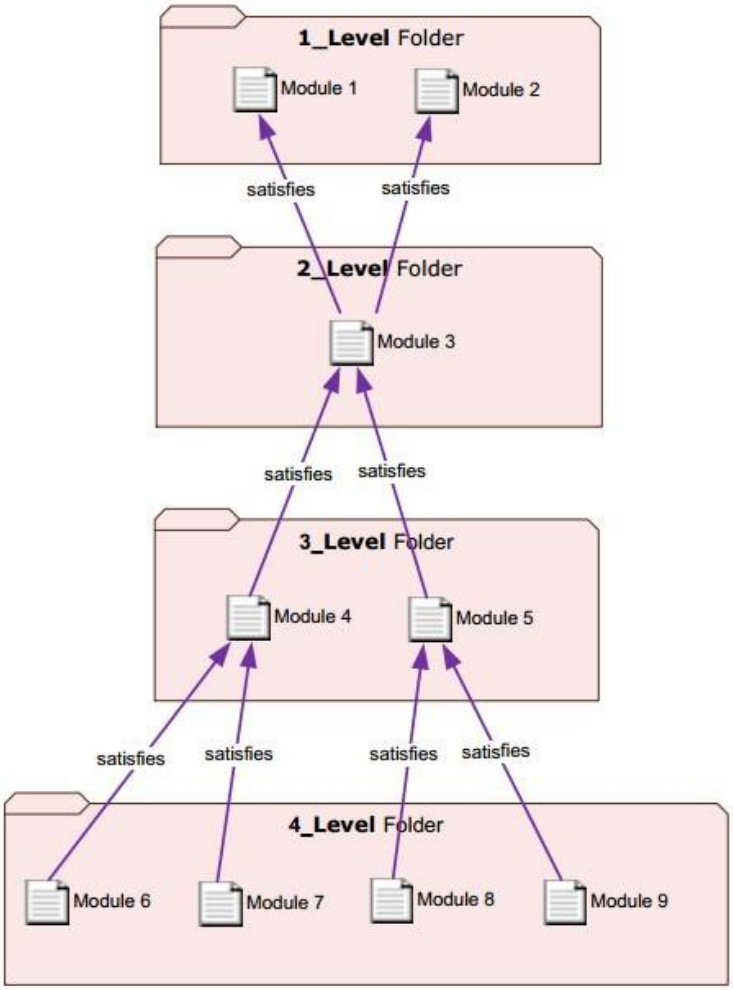


Figure 3 Folder, Module, and Link set Model for the satisfies Vertical Structure

• 3. Change Management

IBM DOORS integrates change management with traceability, enabling effective tracking of updates to requirements. When changes are made, the system tracks which requirements are affected, what the changes entail, and how these changes may impact the project scope. Documentation of changes and notification of relevant stakeholders are supported by DOORS' robust traceability structure. Users can view the history of changes and understand which modules, requirements, or links are impacted by each update. This ensures that all changes can be reviewed and approved before they impact project requirements, enhancing the reliability of change management.

- ### 4. Reporting and Formatting

IBM DOORS provides powerful options for exporting data, such as requirements and traceability links, into MS Word format, enabling users to share project details with stakeholders and maintain clear, accurate documentation. Customizable report templates help tailor reports to specific needs, and traceability information can be included to enhance report completeness. This integration facilitates easy tracking of project progress through detailed documentation.

When exporting data from DOORS to MS Word, formatting can be influenced by the method used. If DOORS' internal tools or custom scripts are employed, the data is sent in a DOORS-specified format, such as indented paragraphs. While this may be suitable for some, it may cause formatting inconsistencies with elements like tabs, bullets, or Rich Text Formatting (RTF), which may not be fully compatible with MS Word.

To address these issues, the Reporting and Publishing Engine (RPE) can be used to export DOORS data with more advanced formatting. By utilizing the att.RPEStyle attribute, which corresponds to MS Word's paragraph styles, RPE can instruct MS Word on how to format the content. In this approach, DOORS objects are not pre-formatted, allowing RPE to apply more sophisticated formatting and enabling quick adjustments. This method offers greater flexibility and control over the final appearance of the report, overcoming the limitations of DOORS' native formatting options.

Market Research: IBM DOORS Alternative Applications

In the field of requirements engineering, IBM DOORS has long been a prominent tool for managing, analyzing, and tracking requirements in complex projects. However, while DOORS is comprehensive, some organizations seek alternatives due to its cost, complexity, or limitations in specific project contexts. This part explores several alternative tools to IBM DOORS, examining their capabilities in requirements management, project scheduling, and requirements analysis. Each tool is reviewed for its strengths and functionalities, but we also highlight critical shortcomings—whether in traceability, integration, usability, or costeffectiveness—that may impact their suitability as an alternative for IBM DOORS in various engineering environments.

1. Primavera

Primavera is primarily a project management tool that is frequently adapted for requirements and project scheduling, particularly in large-scale projects where multi-project planning and tracking are needed. Key features include:

- **Scheduling and Tracking:** Primavera uses the Critical Path Method (CPM) to calculate project schedules based on activity duration and interdependencies. This tool provides flexibility in adjusting project timelines and viewing progress.
- **Resource Management:** The software supports detailed management of labor, material, and non-labor resources, making it suitable for projects with complex resource allocations.

- **Comprehensive Reporting:** Users can generate a variety of reports, enhancing transparency and enabling detailed performance tracking.

Drawbacks:

- **High Cost**: Primavera is relatively expensive compared to other tools, which can be a barrier for small to mid-sized projects or companies.
- **Limited Issue Tracking**: Users report limitations in defect and issue tracking, especially in the early stages. Many users noted the need to dedicate separate sprints just for resolving bugs and issues.
- **Complexity and Usability**: Primavera's extensive features can make it cumbersome and difficult to navigate, especially for those not deeply trained in project management.

2. Microsoft Project

Microsoft Project is a versatile tool commonly used in requirements and project management due to its intuitive interface and functionality. It is well-suited for planning, scheduling, resource allocation, and tracking project progress. Key functionalities include:

- **Milestone and Task Management:** The tool enables users to define milestones, track project progress against set timelines, and adjust schedules based on project needs.
- **Resource Allocation and Management:** Microsoft Project allows users to assign resources, manage workloads, and track time, giving project managers a clear view of resource utilization.
- **Gantt Chart Visualization:** The tool's Gantt chart feature provides a clear, graphical representation of project timelines, dependencies, and task status.

Drawbacks:

- **Rigid Scheduling Constraints**: New users may inadvertently apply constraints to tasks, leading to inflexible scheduling. These constraints, particularly when placed on too many tasks, can restrict the ability to adapt to changing project demands.
- **Tracking Limitations for Short Tasks**: Tracking tasks that span only minutes or hours may be impractical, as the tool is more suited for longer-duration projects and milestones.

- **Limited Integration with Other RE Tools**: Microsoft Project lacks strong integration with specialized RE tools, making it challenging to manage requirements tracking and traceability effectively within the same system.

3. GanttProject

GanttProject is an open-source, cross-platform tool mainly designed for task scheduling and resource management. Known for its lightweight interface and ease of use, GanttProject is popular in smaller projects or educational settings. Key features include:

- **Gantt and PERT Charting:** It provides Gantt charts for project scheduling and PERT charts for dependency analysis, both useful for visualizing project timelines and relationships between tasks.
- **Resource Allocation:** While basic, the resource allocation feature allows for effective task management across teams.
- **Work Breakdown Structure (WBS):** GanttProject offers a WBS feature that breaks down complex projects into manageable tasks, facilitating project planning.

Drawbacks:

- **Limited Features for Large-Scale Projects**: GanttProject lacks advanced features, such as intricate reporting and automated tracking, which may make it less effective for large-scale or enterprise-level projects.
- **Basic Resource Management**: Resource management capabilities are limited and may not support complex resource allocation needs as effectively as other RE tools like Microsoft Project.
- **Compatibility and Export Issues**: Although the software supports export to PDF and HTML, interoperability with other RE and project management tools remains limited, potentially creating issues for projects that require extensive reporting.

4. Requirements Analysis Tool (RAT)

Developed by Accenture, the Requirements Analysis Tool (RAT) is specifically designed to analyze requirements documents by performing both syntactic and semantic analysis. RAT combines structured content extraction with domain-specific ontologies, making it highly useful for identifying ambiguities and inconsistencies within requirements. Key features include:

- **Controlled Syntax and User-Defined Glossaries:** RAT enforces structured syntax and user-defined glossaries, which improve consistency and clarity across requirements documentation.
- **Semantic and Syntactic Analysis:** RAT utilizes a semantic engine to perform deep semantic analysis, identifying gaps, conflicts, and dependencies among requirements, which reduces the potential for misinterpretation or ambiguity.
- **Problem Phrase Glossary:** RAT has an extensive glossary of problematic phrases, helping to detect vague terms that could lead to misinterpretations (e.g., ambiguous timeframes like "daily" or terms like "quickly").

Drawbacks:

- **High Dependency on Glossary Maintenance**: While effective, RAT's reliance on user-defined glossaries means it requires continuous maintenance to stay relevant to specific project needs and terminologies.
- **Complex Setup for Non-Technical Users**: RAT's features may be challenging for nontechnical users due to its emphasis on structured syntax and controlled language, making it less accessible for stakeholders without a technical background.
- **Limited Integration with RE and Project Management Tools**: RAT primarily focuses on requirements analysis and lacks robust integration with tools for broader project management activities, such as scheduling and task tracking.

5. QuARS (Quality Analyzer for Requirements Specifications)

QuARS is a requirements analysis tool known for its support in improving the quality of requirements specifications. It focuses on identifying linguistic and syntactic issues within requirements documents, which helps organizations enhance the clarity and consistency of their requirements. Key features include:

- **Linguistic Analysis:** QuARS evaluates the language used in requirements to identify ambiguities, inconsistencies, and incompleteness, which often lead to misinterpretations during development.

- **Support for Requirements Quality Control:** The tool offers built-in metrics for assessing the quality of requirements, allowing users to measure aspects like clarity, precision, and consistency.
- **Phrasal Analysis:** It uses specific algorithms to scan for problematic phrases, making it especially useful in projects where requirements are captured in natural language.

Drawbacks:

- **Limited Scope of Analysis**: QuARS is restricted to linguistic and syntactic analysis, lacking deeper semantic analysis capabilities. This means it may not detect complex interdependencies between requirements effectively.
- **No Native Traceability Features**: QuARS does not support requirements traceability natively, which is a crucial feature for managing changes and tracking requirement lifecycle.
- **Dependency on Well-Written Requirements**: The effectiveness of QuARS depends significantly on the initial quality of requirements. If requirements are already wellstructured, the tool's benefit may be limited.

In comparison to IBM DOORS, which provides extensive support for requirements management, traceability, and collaboration in large projects, many alternative tools focus on specific aspects of requirements engineering, such as project management (Primavera, Microsoft Project), linguistic analysis (QuARS), or requirements analysis (RAT). While these tools offer unique advantages, they often lack the comprehensive traceability and integration that DOORS provides.
Each of the tools reviewed has limitations that could affect its suitability depending on project size, complexity, and specific RE needs:

- **For Project Management-Centric Needs**: Primavera and Microsoft Project offer robust scheduling, resource management, and milestone tracking. However, they lack detailed requirements analysis features and may not integrate seamlessly with traceability needs.
- **For Requirements Quality and Analysis**: Tools like RAT and QuARS provide strong analytical capabilities but are often limited by lack of integration with broader project management tools. These tools are suitable for ensuring requirements clarity and consistency but may require additional tools for full project lifecycle support.

- **For Lightweight Project Management**: GanttProject is a good choice for simpler projects but lacks the advanced features necessary for complex requirements management.

From User Requirements to System Specifications: An AI-Based Approach to Requirement Analysis

In this project, we aim to leverage artificial intelligence (AI) technology to automatically generate system requirements from user requirements. The methods and results derived from reviewing similar research are summarized in this section of the report.

This study proposes an approach for automatically extracting key use-case components—specifically actors and actions—from software requirement documents written in natural language. By combining Artificial Neural Networks (ANN) and Natural Language Processing (NLP) techniques, the goal is to enhance the speed and accuracy of requirement analysis. The study utilizes the GATE (General Architecture for Text Engineering) platform for linguistic analysis, followed by classification of extracted features using ANN.

Objective

In software development, requirement documents play a crucial role in conveying detailed information about system functionality. However, manually extracting and processing this information is time-consuming and error-prone. The aim of this study is to automate the detection of actors and actions expressed in natural language within these documents, thus improving efficiency.

Methodology

The methodology integrates NLP and ANN for requirement analysis, divided into four key stages:

Linguistic Analysis

- *Tokenization*: The text is broken down into tokens (e.g., words, sentences), which serve as input for further analysis.

- *Syntax Analysis:* The grammatical structure of sentences is analyzed using tools like GATE and Stanford-CoreNLP.

- *Semantic Analysis*: Actions and their associated actors are analyzed based on semantic relationships to identify use-case components.

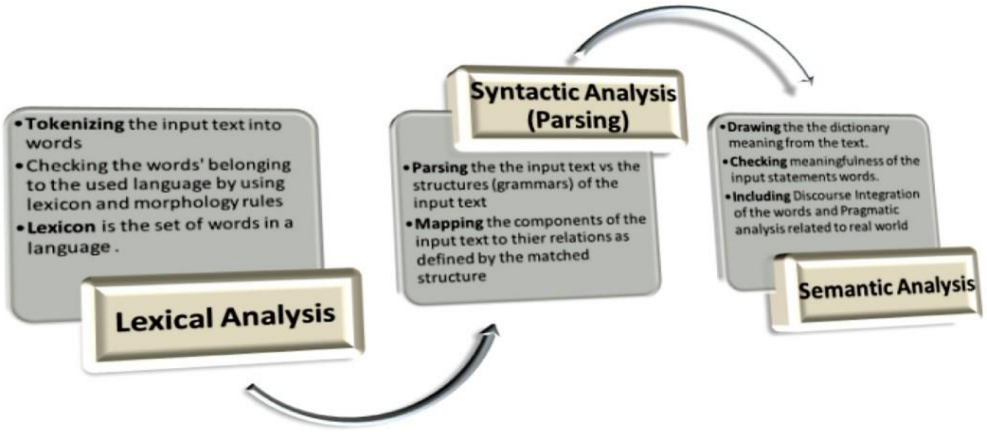


Figure 4 NLP components

Pre-Processing

- Tokens, categories (e.g., nouns, verbs), and dependencies are encoded numerically to prepare them for classification by the ANN.

Classification with ANN

- A Back-Propagation Neural Network (BPNN) classifies linguistic components into use-case labels (actor, action, other).

- The network includes an input layer, hidden layers, and an output layer, with training carried out using MATLAB's neural network tools.

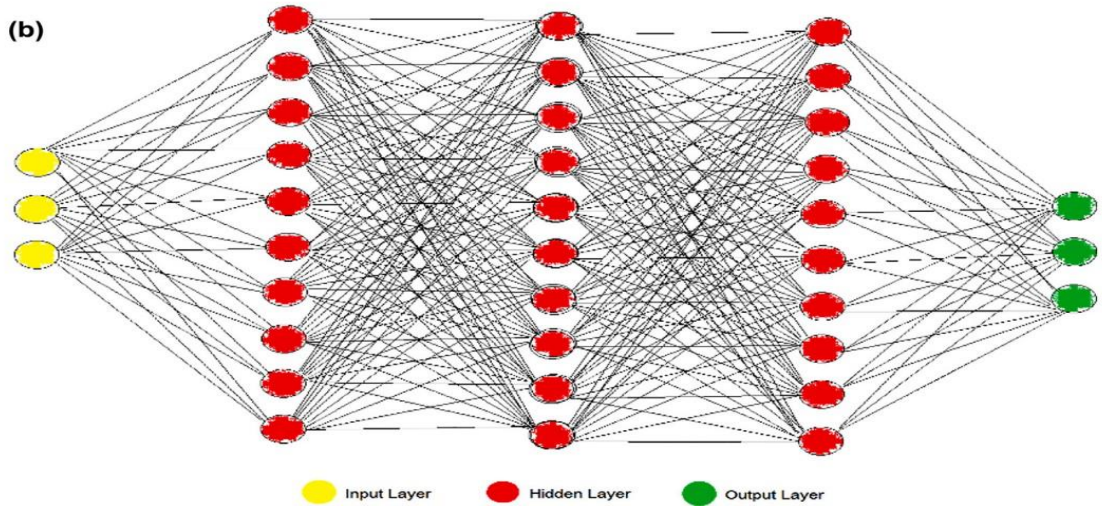


Figure 5 BPNN topology of IT4RE

Evaluation

- The network's performance is assessed using precision, recall, and F-measure metrics to evaluate the accuracy of classification.

Results and Discussion

The results indicated that precision varied between 17% and 63%, while recall ranged from 5% to 100%. The average F-measure was 55%, suggesting that the method performed less effectively with complex sentence structures and specialized vocabulary due to limitations in NLP tools. However, this approach demonstrates an efficient and accurate method for requirement analysis by integrating NLP and ANN techniques. Enhancing NLP tools and model parameters is necessary to improve accuracy, especially for handling complex linguistic structures.

Adaptation of this Approach for Transforming User Requirements to System Requirements

This combined ANN and NLP approach can be adapted to automatically convert user requirements into system requirements by interpreting and specifying them in a more technical, system-level language. The process would involve the following steps:

- *Extended Linguistic Analysis:* User requirements, often expressed in high-level terms, would be analyzed for key concepts and inferences. Using Concept Mapping, general terms are mapped to specific system functions or modules, while Inference is used to derive more detailed system requirements.

- *Pre-Processing and Encoding*: Key terms and dependencies are encoded in a technical language, facilitating the definition of system-level dependencies and features.

- *Mapping User Requirements to System Requirements:* A customized ANN model would categorize user requirements into system requirement categories such as functionality, performance, and security. Specific rules guide the transformation process based on recognized patterns.

- *System Requirement Generation:* The model generates detailed system requirements, extracting technical specifications and organizing them into functional modules, thereby providing a structured transition from user requirements to system design.

This method, combining ANN and NLP, offers a promising solution for automating the conversion of user requirements into system requirements. However, to achieve greater accuracy, especially in handling complex language structures, further improvements in NLP tools and model parameters are needed. This approach has the potential to significantly enhance the efficiency of requirement analysis processes in software development projects.

Conclusion

Effective requirements management is essential for the success of complex software and systems engineering projects. IBM DOORS has proven to be a comprehensive solution for handling the intricacies of requirements traceability, version control, and reporting. However, due to its cost and complexity, many organizations turn to alternative tools such as Primavera, Microsoft Project, and specialized analysis tools like RAT and QuARS, each with distinct advantages and drawbacks. While these alternatives may be suitable for specific project needs, they often lack the comprehensive traceability and integration that IBM DOORS offers. Furthermore, the application of Artificial Intelligence, specifically Natural Language Processing (NLP) and Artificial Neural Networks (ANN), presents an exciting opportunity to automate the conversion of user requirements into system requirements. This AI-based approach, though not without challenges in handling complex linguistic structures, shows promise in enhancing the speed and accuracy of requirement analysis, offering significant improvements in project efficiency. Continued advancements in AI and NLP technologies will further refine this approach, enabling more seamless transitions from user requirements to system specifications in the future.

SOFTWARE REQUIREMENT SPESIFACATION

1.Introduction

1.1. Project Purpose

The purpose of CSEK Requirements Management System is to develop a new, local requirements management system. One of the most important tools in this field, IBM DOORS, has served as an inspiration for the localization of our project. IBM DOORS is a requirements management tool with a significant role in the defense industry and high licensing costs. The primary goal of our project is to localize this critical tool, for various sectors, and provide a more accessible, customizable, and cost-effective alternative suitable for our country.

1.2. Project Scope

The aim of this project is to develop a local requirements management system solution for the defense sector. Widely used technologies like IBM DOORS present operational and budgetary challenges due to high licensing costs and limited accessibility in regional defense projects. Our project aims to provide a more cost-effective, flexible, and locally developed alternative to these existing technologies.

The system will offer configurable role-based access control based on user needs. It will allow users with roles such as Administrator, System Engineer, and Reviewer to access only the areas relevant to their specific responsibilities. This approach will ensure a secure system while minimizing potential errors. Additionally, the system will enable precise organization, categorization, and efficient management of requirements across multiple projects.

The system will provide bidirectional traceability of requirements, allowing users to quickly assess how one requirement impacts others. In the context of change management, the system will ensure traceability of all changes, including revisions and deletions. Users will also benefit from reporting features that allow the export of requirements in various formats, such as MS Word and PDF.

The completed system will not only offer a cost-effective solution for defense projects but will also add significant value in terms of data security and customization. As a locally developed software solution, it will comply with national security standards and meet the technological requirements of defense projects. Our project can be positioned as a critical step in ensuring the sustainability and cost-efficiency of the defense sector.

Although artificial intelligence integration was considered for the generation of system requirements in the initial planning stages, it would be a better approach to leave this part as a part of the system that can be developed later due to the difficulties of finding datasets for model training.

In addition to this, the system is open to future development in terms of features that are not included in our current project design, such as user input of tests of system requirements, tasking feature where project managers can assign which tasks to be performed by whom, and adding different formats to the reporting process in the export section.

1.3. Glossary

- IBM DOORS:

A software tool used for requirements management, widely preferred in the defense industry and known for its high license fees.

- Role-Based Access Control (RBAC):

An access control mechanism for determining and managing the authorization of users in the system according to their roles.

- Bidirectional Traceability:

Bidirectional linking of requirements so that it is possible to quickly analyze how one requirement affects others.

- Baseline:

A structure that is saved as a fixed version of the requirements at a given moment. Allows changes to be tracked.

- Microservices Architecture:

Software architecture that allows systems to be composed of independent, small and modular services.

- Keycloak:

An open source tool for authentication and authorization.

- REST API:

A protocol used for communication between applications. Works in accordance with OpenAPI standards.

- Netflix Eureka:

A service discovery tool that allows dynamic registration and discovery of microservices.

- gRPC:

A communication protocol that enables fast and low latency communication between microservices.

- Kafka:

A messaging tool for event-based communication and data processing.

- Redis:

A caching system used to reduce database load and ensure that frequently accessed data is stored in the cache.

- OAuth 2.0:

An open standard protocol used for authorization.

- AES-256:

An encryption algorithm used to protect data.

1.4. Overview of This Document

This document has been prepared to detail the requirements for the development of our project named CSEK Requirements Management System. This project focuses on addressing the issue of most requirement management tools used in various industries being non-local and having high licensing fees. Our project targets a user base that manages projects by utilizing requirement management tools used in various sectors.

The Introduction section includes the purpose, scope, glossary, references, and this part, which is the overview section. The purpose of the Introduction section is to provide a brief definition of the project and explain its objectives. The Overall Description section includes topics such as the software's general perspective, user characteristics, and constraints. The Specific Requirements section details functional, performance, and interface requirements. The Supporting Information section provides additional explanations. Overall, this document is designed as a guide for developers.

2. Overall Description

2.1. Product Perspective

CSEK Requirements Management System provides a standalone software solution designed to efficiently manage, track and organize project requirements. The system aims to solve the problems of existing requirements management tools such as high license costs, local incompatibilities and complexity. By providing users with a customizable and economical tool, it plans to reduce dependency on traditional software in the industry.

It has a user-friendly and web-based interface, so users can access the system from any modern browser. It offers an intuitive menu that makes it easy to use and allows users to easily navigate between requirements, reports, baselines and historical records. It also provides a visual indicator next to changed or unapproved requirements, allowing users to easily track changes and approval processes. The system provides clear feedback on all actions taken by users.

The CSEK Requirements Management System provides a stable environment with extensive user assistance. The system is intended to run on strong server hardware capable of supporting up to 500 users simultaneously. It also supports cloud-based infrastructure services (such as AWS), which offer scalability and flexibility. The software utilizes a variety of innovative software interfaces. The system uses Keycloak for authentication and authorization, allowing users to log in securely. Role-based access control stops users from gaining unwanted access, making the environment more secure. REST APIs that follow OpenAPI standards facilitate communication between user and system services, as well as external applications. Netflix Eureka enables dynamic registration and discovery of microservices, making it possible for the system to have a modular and flexible structure. Simultaneous communication between microservices is provided using gRPC. The communication between user requirements, system requirements and sub-header requirements microservices is realized through this infrastructure. Kafka provides synchronized communication between requirements microservices and snapshot services, enabling efficient event-driven communication. Using a PostgreSQL 13 or higher database, all requirements and associated change history are securely stored. Redis improves performance and reduces database load by caching frequently accessed data. With comprehensive traceability, it is possible to create links between all user, system and sub-header requirements and easily trace the source of each requirement or the areas it affects. Furthermore, the system enables users to create customized reports according to their needs.

Requirements can be exported in Word and PDF formats while maintaining their hierarchical structure. Users can get results in a short time thanks to the system's fast reporting feature.

Finally, CSEK Requirements Management System stands out as a cost-effective and high-performance solution. Easily adaptable to local needs, it offers a software that can be ideal for most levels of organizations, from small and medium-sized enterprises to large corporations. The system enables organizations to manage their requirements in an organized, reliable and efficient manner.

2.2. User Characteristics

The system offers functionality based on three primary roles: Admin, System Engineer, and Reviewer.

1.Administrator:

- Required Features:

Ability to assign and edit user roles.
Ability to manage user access, add or remove users from the system.
Ability to create, edit, and manage project requirements.
Ability to define user permissions to ensure system security.
Ability to create and manage project baselines.

- Access:

Full access to all system functionalities (create, edit, delete requirements, manage users).
Authority to manage user roles, permissions, and requirements.
Authority to create, manage, and delete project baselines.
Authority to ensure system security and data integrity.

2.System Engineer

- Required Features:

Ability to create, edit, track, and manage user, system, and subheading requirements.
Ability to link requirements for traceability.
Ability to assign attributes to requirements to organize them effectively.

- Access:

Access to create, edit, manage, and track requirements.
Authority to establish links between requirements for traceability.
Authority to update and delete requirements (validation is required).

3.Reviewer

- Required Features:

Ability to view all types of requirements (user, system, subheading).
Ability to review the accuracy and quality of requirements.
Ability to export requirements in formats such as PDF and Word (reporting).

- Access:

Read-only access to all requirements.
Authority to report and export requirements.
No authority to edit or delete content.
These assignments and roles will provide the opportunity to minimize errors and create a secure environment in the system.

2.3. General Constraints

- Hardware Capacity:

The system is optimized to support a maximum of 500 concurrent users at the same time. Additional server resources will be required for more users.

- Access Control:

Only authenticated and authorized users can access the system.

- Connection Requirement:

The system can only be used with a continuous internet connection.

- Project Duration:

The system is scheduled for delivery within 6 months.

- Language Support:

The first release will only provide support for certain languages. Other languages will be developed in subsequent releases.

- Platform Dependency:

The system will be web-based only and no mobile application support will be provided.

- Legal Restrictions:

The system must comply with the local legal regulations of the countries.
These clauses guarantee the completion of the project within the specified limits and compliance with certain standards.

2.4. General Assumptions

- User Access:

All users will have an internet connection and a modern web browser to access the system.

- User Technical Knowledge:

Users are assumed to have basic computer skills.

- System Load:

The system is optimized to support a maximum of 500 simultaneous users. It is assumed that the number of users will not exceed this limit.

- Data Quality:

It is assumed that the data to be entered into the system is accurate, complete and in the appropriate format. It is assumed that no incorrect or incomplete data will be entered.

- Integration:

The system will work seamlessly with other systems in accordance with the specified API standards and integration components.

- Legal Compliance:

It is assumed that the legal regulations regarding data privacy and security in the regions where the system will be implemented will not change during the project period.

- Project Schedule:

It is assumed that the project team will adhere to the planned timeline and that there will be no major delays during the project.
These assumptions represent the basic principles underlying the planning and implementation phases of the project.

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

- The system shall provide a web-based user interface accessible via modern browsers.
- The system shall provide an intuitive navigation menu to allow users to easily navigate between requirements, reports, baselines, and history logs.
- The system shall provide a visual indicator (e.g., a star icon) next to modified or unapproved requirements, allowing users to track changes and approvals.
- The system shall provide clear visual feedback for user actions.
- The system shall display error messages in clear, understandable language when an action cannot be completed.

3.1.2. Hardware Interfaces

- The system shall run on server hardware with sufficient CPU, memory, and storage to support up to 500 concurrent users.
- The system shall support cloud-based infrastructure services for deployment and scaling (eg. AWS).

3.1.3. Software Interfaces

- The system shall integrate with Keycloak for authentication and authorization, leveraging its IAM capabilities.
- The system shall use REST APIs for communication between user and system services and external applications, as defined by OpenAPI specifications.
- The system shall use Netflix Eureka for service discovery to enable dynamic registration and discovery of microservices.
- The system shall use gRPC for synchronous inter-service communication between user requirements microservice, system requirements microservice, and subheading requirements microservice.
- The system shall use Kafka for synchronous communication between requirement microservices and snapshot services for efficient event-driven communication.
- The system shall integrate with Redis for caching frequently accessed data to improve performance and reduce database load.
- The system shall integrate with PostgreSQL 13 or higher for database management and storage.

3.1.4. Communications Interfaces

- The system shall ensure secure communication for all API requests via OAuth 2.0 authentication.
- The system shall leverage Netflix Eureka for service discovery, enabling seamless communication between microservices by dynamically locating and accessing available services.
- The system shall use gRPC for efficient and low-latency communication between user, system, and subheading requirement microservices.
- The system shall use Kafka for communication between requirement microservices and snapshot services, supporting event-driven architecture.
- The system shall use Redis for caching purposes to improve performance and reduce latency during inter-service communication.

3.2. Functional Requirements

3.2.1 Login Requirements

FR 1.1:

- The system shall support three distinct user roles: Admin, System Engineer, and Reviewer.
    - Users must log in with a role-based account to access the system.

FR 1.2:

- The system shall require a valid username and password for authentication.
    - All users must provide credentials to access the system.

FR 1.3:

- The system shall allow admins to assign user roles (Admin, System Engineer, Reviewer) during user registration.
    - Admins are responsible for managing user access and roles.

FR 1.4:

- The system shall restrict access to system features based on the user's assigned role.
    - Different roles (Admin, System Engineer, Reviewer) have access to specific system functionalities.

3.2.2. User Requirement Management

FR 2.1:

- The system shall allow system engineers to create new user requirements.

FR 2.2:

- The system shall automatically generate a unique identifier for each user requirement based on its type (e.g., UR-001).
    - This facilitates easy reference and management of requirements.

FR 2.3:

- The system shall allow system engineers to input input text, numerical values and dates for the user requirement.

FR 2.4:

- The system shall allow system engineers to add images to a requirement description.
    - Supports enhanced documentation of requirements with visual elements.

FR 2.5:

- The system shall allow system engineers to assign each user requirement to a specific attribute.
    - Helps in better definition and understanding of requirements.

FR 2.6:

- The system shall allow system engineers to underline important points and bold text when needed.
    - System engineers can apply bold or underline styles to selected text.

FR 2.7:

- The system shall allow reviewers to view all user requirements in a read-only mode.
    - Reviewers can access and review requirements without modifying them.

3.2.3. System Requirements Management

FR 3.1:

- The system shall allow system engineers to create new system requirements.

FR 3.2:

- The system shall automatically generate a unique identifier for each system requirement based on its type (e.g., SR-001).
    - This facilitates easy reference and management of requirements.

FR 3.3:

- The system shall allow system engineers to input input text, numerical values and dates for the system requirement.

FR 3.4:

- The system shall allow system engineers to use embedded images within system requirements.
    - Ensures detailed technical specifications can be documented.

FR 3.5:

- The system shall allow system engineers to assign each system requirement to a specific attribute.
    - Helps in better definition and understanding of requirements.

FR 3.6:

- The system shall validate required field before saving a new system requirement.
    - Mandatory field include User requirement link.

FR 3.7:

- The system shall allow reviewers to view all system requirements in a read-only mode.
    - Reviewers can access and review requirements without modifying them.

3.2.4. Subheading Requirements Management

FR 4.1:

- The system shall allow system engineers to divide system requirements into subheadings, such as Functional Requirements, Non-Functional Requirements.
    - Provides better organization and clarity by categorizing requirements under appropriate subheadings.

FR 4.2:

- The system shall allow system engineers to create new subheading requirements.

FR 4.3:

- The system shall require system engineers to select whether a subheading requirement is functional or non-functional during creation.
    - Ensures proper classification of subheading requirements for traceability and organization.

FR 4.4:

- The system shall automatically generate a unique identifier for each subheading requirement based on its type (e.g., FR-001, NFR-001).
    - This facilitates easy reference and management of requirements.

FR 4.5:

- The system shall allow system engineers to manually input a header (e.g., Hardware, Interface) for the subheading requirement after classification.
    - Provides flexibility for system engineers to define relevant headers based on the requirement context.
    - The system shall allow system engineers to leave the header field empty.

FR 4.6:

- The system shall allow system engineers to input input text, numerical values and dates for the subheading requirement.

FR 4.7:

- The system shall allow system engineers to use embedded images within subheading requirements.
    - Ensures detailed technical specifications can be documented.

FR 4.8:

- The system shall allow subheading requirements to be linked to their parent system requirements.
    - Links between subheading and system requirements provide traceability and context.

FR 4.9:

- The system shall allow system engineers to assign each subheading requirement to a specific attribute.
    - Helps in better definition and understanding of requirements.

FR 4.10:

- The system shall validate required field before saving a new subheading requirement.
    - Mandatory field include system requirement link.

FR 4.11:

- The system shall allow reviewers to view all subheading requirements in a read-only mode.
    - Reviewers can access and review requirements without modifying them.

3.2.5. Traceability Requirements

FR 5.1:

- The system shall allow system engineers to create bi-directional links between user requirements and system requirements.
    - Users can trace from user requirements to system requirements and vice versa.

FR 5.2:

- The system shall allow system engineers to create bi-directional links between system requirements and subheading requirements.
    - Users can trace from system requirements to subheading requirements and vice versa.

FR 5.3:

- The system shall display an impact analysis when a requirement is linked or unlinked.
    - Shows how changes affect related requirements.

FR 5.4:

- The system shall allow users to filter requirements based on their traceability status (e.g., linked or unlinked).
    - Helps identify orphaned, missing or unlinked requirements.

3.2.6. Change Management Requirements

FR 6.1:

- The system shall allow system engineers to edit existing requirements.
    - System engineers with appropriate permissions can modify requirement details such as description, priority, and attributes.

FR 6.2:

- The system shall allow system engineers to delete existing requirements.
    - Deletion shall require confirmation and appropriate permissions by admin to prevent accidental removal.

FR 6.3:

- The system shall maintain a detailed history of every change, including edits and deletions, for each requirement.
    - The history log shall include the user who made the change, timestamp, and a summary of the modification.

FR 6.4:

- The system shall display a yellow colored star icon next to a requirement that has been edited but not yet reviewed.
    - The special indicates that the requirement needs stakeholder review.

FR 6.5:

- The system shall display a red colored star icon on all related requirements when a linked requirement is edited or deleted.
    - Ensures stakeholders are aware of potential impacts on related requirements.

FR 6.6:

- The system shall provide a "Clear Suspicion" button to remove the special star icon once the changes have been reviewed and acknowledged.
    - Allows users to mark changes as reviewed and finalize the requirement.

FR 6.7:

- The system shall provide an option to recover or revert a deleted requirement from the history log.
    - Helps recover accidentally deleted requirements or review historical data.

FR 6.8:

- The system shall track changes made to images within a requirement.
    - History logs should include changes to visual elements.

3.2.7. Release Requirements

FR 7.1:

- The system shall allow admin to create a baseline of the entire project.
    - A baseline will serve as a frozen version of the requirements at a specific point in time.

FR 7.2:

- The system shall provide versioning for baselines, allowing multiple baselines to be created and stored.
    - Each baseline will be assigned a unique version identifier for easy tracking and comparison.

FR 7.3:

- The system shall display to users for comparing baselines to identify changes between versions.
    - Requirements releases will be displayed together column by column for easy comparison.

FR 7.4:

- The system shall allow admin to delete obsolete baselines after proper review.
    - Baselines that are no longer needed can be removed to optimize storage space.

3.2.8. Reporting Requirements

FR 8.1:

- The system shall allow users to export user system and subheading requirements in a hierarchical structure.

    - Ensures that requirements are exported with their relationships intact, displaying parent-child links.

FR 8.2:

- The system shall provide users with the option to export requirements in either a Document, Table, or Book format.

    - Users can choose the format that best suits their reporting needs:
        - Document: Requirements are presented in a narrative format.
        - Table: Requirements are presented in a tabular format with rows and columns.
        - Book: Requirements are grouped into chapters or sections for structured documentation.

FR 8.3:

- The system shall allow users to export requirements to MS Word format.

    - Supports the creation of professional, editable documents.

FR 8.4:

- The system shall allow users to export requirements to PDF format.
    - Ensures that requirements can be shared as read-only, professional reports.

FR 8.5:

- The system shall preserve the hierarchical structure of requirements in both Word and PDF exports.
    - Ensures that parent-child relationships are clearly indicated in exported documents.

FR 8.6:

- The system shall allow users to select which requirements (user, system or subheading) to export.
    - Users can choose to export only user requirements, only system requirements, or both.

FR 8.7:

- The system shall provide a preview option before exporting the document.
    - Users can review the export before generating the final file to ensure accuracy.

FR 8.8:

- The system shall allow users to export visual elements (images) along with text in the exported document.
    - Ensures that all embedded visual elements are included in the final report.

3.2.9. Database Requirements

FR 9.1:

- The system shall allow system engineers and admins to save all modules to the PostgreSQL database.
    - Ensures that all modules(user, system and subheading requirements) are securely stored and accessible for future reference.

FR 9.2:

- The system shall store a history of all changes made to requirements, including timestamps and user details, in PostgreSQL.
    - Provides traceability and accountability for all modifications.

FR 9.3:

- The system shall allow admins to save all baselines to the PostgreSQL database.
    - Enables secure storage of baseline configurations for version control and traceability.

3.3. Non-Functional Requirements

3.3.1. Performance

NFR 1.1:

- The system shall provide a response time of less than 2 seconds for all database queries involving requirements retrieval.
    - Ensures that users experience minimal delays when interacting with the system.

NFR 1.2:

- The system shall save all modules(user, system and subheading requirements) to the PostgreSQL database within 5 seconds for projects containing up to 1000 requirements.
    - Provides efficient storage of entire modules, even for large projects.

NFR 1.3:

- The system shall save baselines to the PostgreSQL database within 5 seconds for projects.
    - Ensures quick and reliable saving of baseline configurations for version control and traceability.

NFR 1.4:

- The system shall generate reports in less than 10 seconds for projects containing up to 1000 requirements.
    - Ensures efficient report generation for large projects.

3.3.2. Security

NFR 2.1:

- The system shall use API Gateway to manage and secure all authentication and authorization requests via REST APIs.

NFR 2.2:

- The system shall integrate with Keycloak to handle identity and access management, supporting role-based access control (RBAC) for Admin, System Engineer, and Reviewer roles.

NFR 2.3:

- The PostgreSQL database shall encrypt data at rest using AES-256 symmetric encryption to ensure data confidentiality.

NFR 2.4:

- The PostgreSQL database shall use TLS 1.3 to encrypt data in transit between the database and the application.

3.3.3. Reliability

NFR 3.1:

- The system shall be available 99.9% of the time, excluding scheduled maintenance periods.
    - Guarantees high availability to minimize downtime.

NFR 3.2:

- The system shall use Netflix Eureka to dynamically reroute traffic to healthy microservices during failures or downtime.

NFR 3.3:

- The system shall use write-ahead logging (WAL) for PostgreSQL to ensure data durability and recoverability in case of a crash.

3.3.4. Usability

NFR 4.1:

- The system shall provide an intuitive user interface (UI) that allows users to create, edit, link, and review requirements with minimal training.

NFR 4.2:

- The system shall support keyboard shortcuts for common actions such as saving, editing, and navigating between requirements.

NFR 4.3:

- The system shall ensure that all actions (e.g., saving, linking, or exporting) are performed with a maximum of four clicks.

NFR 4.4:

- The system shall display error messages in plain language, providing clear instructions on how to resolve the issue.

3.3.5. Maintainability

NFR 5.1:

- The system shall support the seamless addition or removal of microservices without disrupting other services, utilizing Netflix Eureka for dynamic service discovery and communication.

NFR 5.2:

- The system shall ensure that all microservices are independently deployable and maintainable.

3.4. Use Cases

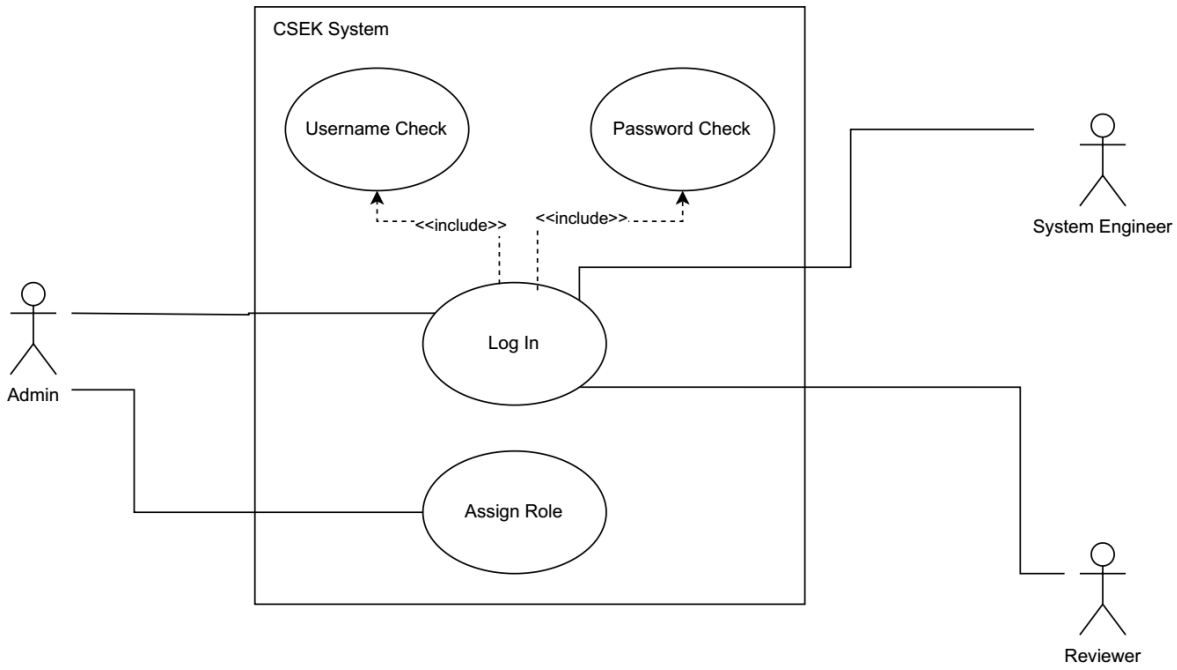


Figure 1- Use Case-1

UC-1.1: Log In
Description: Users must log in with a role-based account to access the system.
Actors: Admin, System Engineer, Reviewer
Preconditions: Valid username and password input.
Postconditions: User enters to the system.

UC-1.2: Assign Role
Description: The system shall allow admins to assign user roles (Admin, System Engineer, Reviewer) during user registration.
Actors: Admin
Preconditions: Must login to the system as Admin role.
Postconditions: Role assignes to the target user.

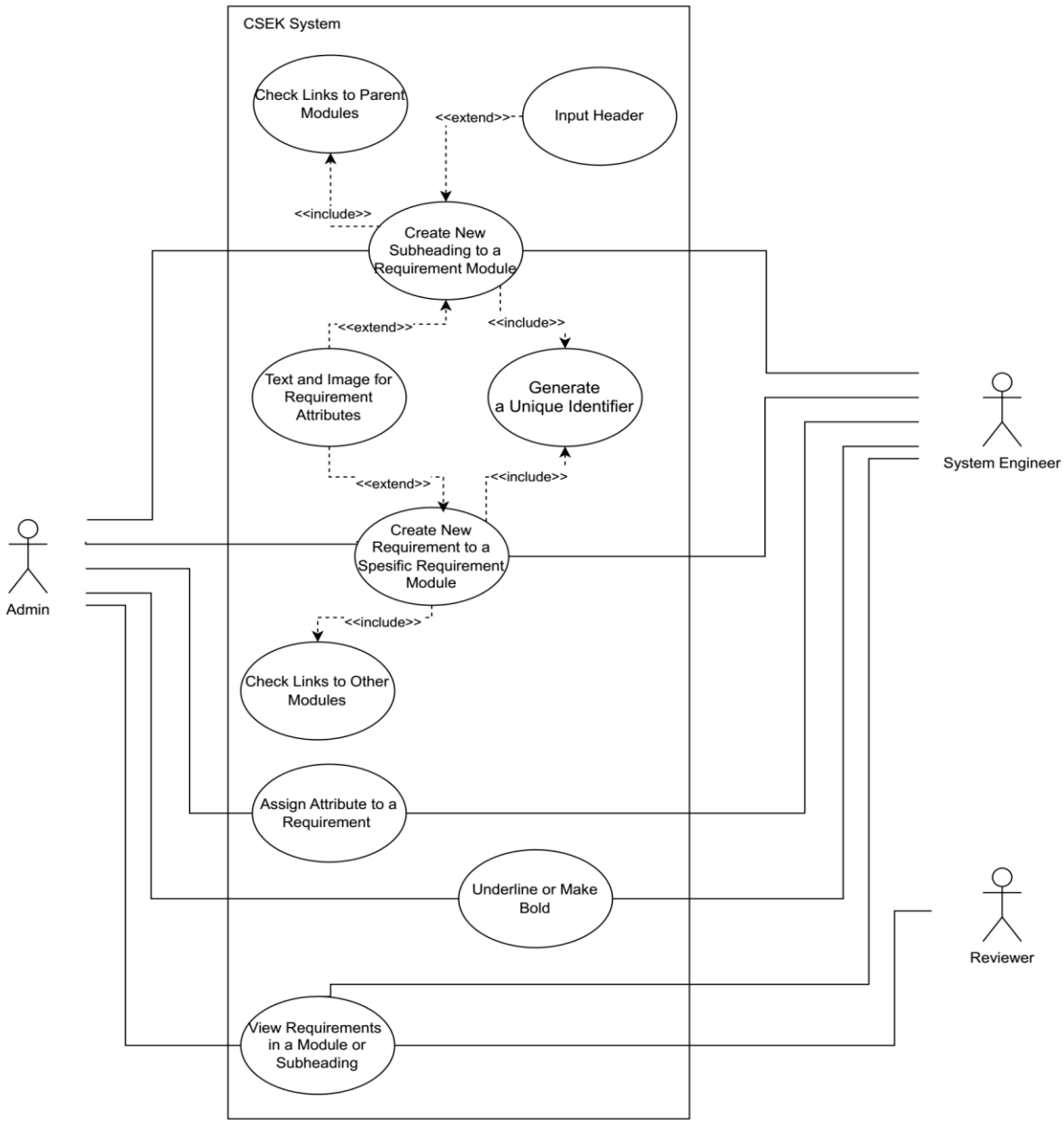


Figure 2 Use Case-2

UC-2.1: Create New Requirement to a Spesific Requirement Module
Description: The system shall allow system engineers to create new user or system requirement to relevant requirement module.
Actors: Admin, System Engineer
Preconditions: Input text or image format data for attributes.
Postconditions: Check links to other requirement module types and genereate a unique identifier to created requirement.

UC-2.2: Create New Subheading to a Requirement Module
Description: The system shall allow system engineers to divide system requirements into subheadings, such as Functional Requirements, Non-Functional Requirements.
Actors: Admin, System Engineer

Preconditions: Input text or image format data for attributes. Or input a header (e.g., Hardware, Interface) for the subheading requirement after classification.
Postconditions: Check links to the parent requirement module and genereate a unique identifier to created subheading.

UC-2.3: Assign Attribute to a Requirement
Description: The system shall allow system engineers to assign each requirement to a specific attribute.
Actors: Admin, System Engineer
Preconditions: Requirement must be defined already.
Postconditions: The system adds new attribute to requirement as a column.

UC-2.4: Underline or Make Bold
Description: The system shall allow system engineers to underline important points and bold text when needed.
Actors: Admin, System Engineer
Preconditions: Requirement must be defined already.
Postconditions: The system underlines or applies bold in selected parts on a requirement.

UC-2.5: View Requirements in a Module or Subheading
Description: The system shall allow reviewers to view all user requirements in a read-only mode.
Actors: Admin, System Engineer, Reviewer
Preconditions: Requirements must be defined already.
Postconditions: Reviewers accesses and reviews requirements without modifying them.

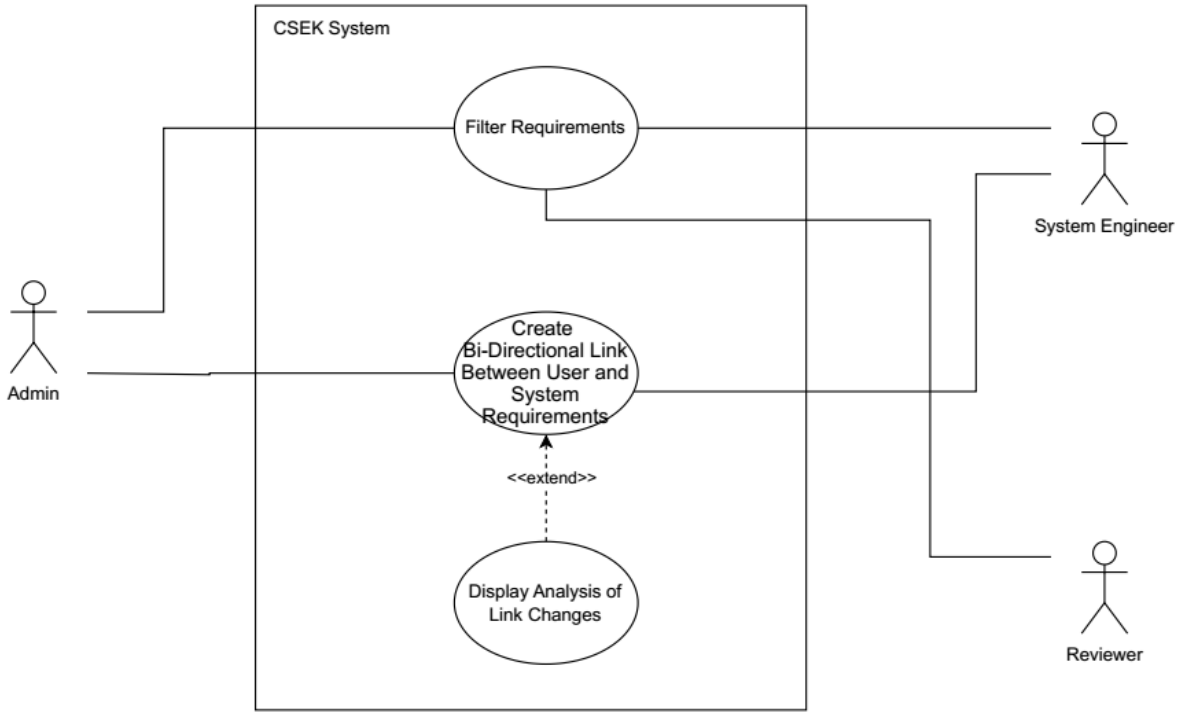


Figure 3 Use Case-3

UC-3.1: Create Bi-Directional Link Between User And System Requirements
Description: The system shall allow system engineers to create bi-directional links between different modules.
Actors: Admin, System Engineer
Preconditions: Requirements must be defined already.
Postconditions: The system shall display an impact analysis when a requirement is linked or unlinked.

UC-3.2: Filter Requirements
Description: The system shall allow users to filter requirements based on their traceability status (e.g., linked or unlinked).
Actors: Admin, System Engineer, Reviewer
Preconditions: Requirements must be defined already.
Postconditions: The system applies filter to analysis.

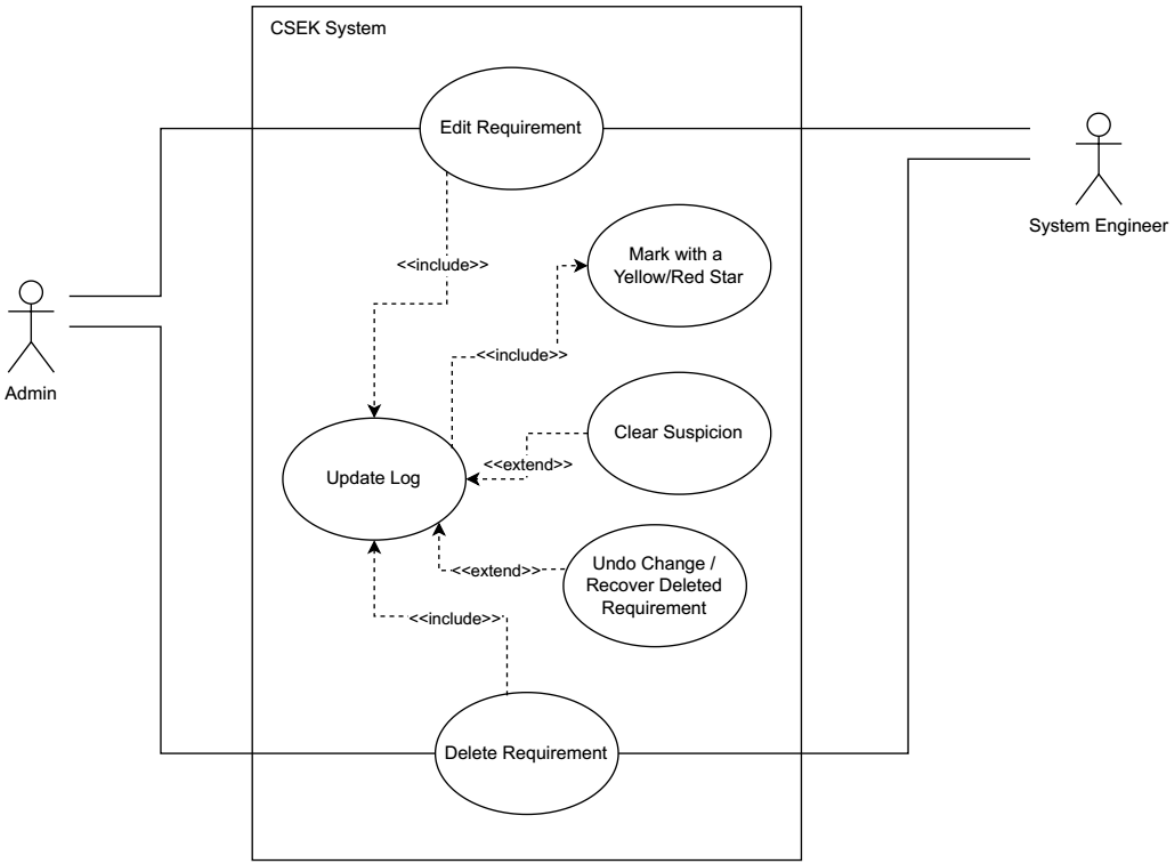


Figure 4 Use Case-4

UC-4.1: Edit Requirement
Description: The system shall allow system engineers to edit existing requirements.
Actors: Admin, System Engineer
Preconditions: Requirement must be defined already.
Postconditions: The system writes change to the history log and to display suspicion, marks relevant requirements with colored stars. Also can be undone.

UC-4.2: Delete Requirement
Description: The system shall allow system engineers to delete existing requirements.
Actors: Admin, System Engineer
Preconditions: Requirement must be defined already.
Postconditions: The system writes deletion to the history log and to display suspicion, marks relevant requirements with colored stars. Also deleted requirement can be recovered.

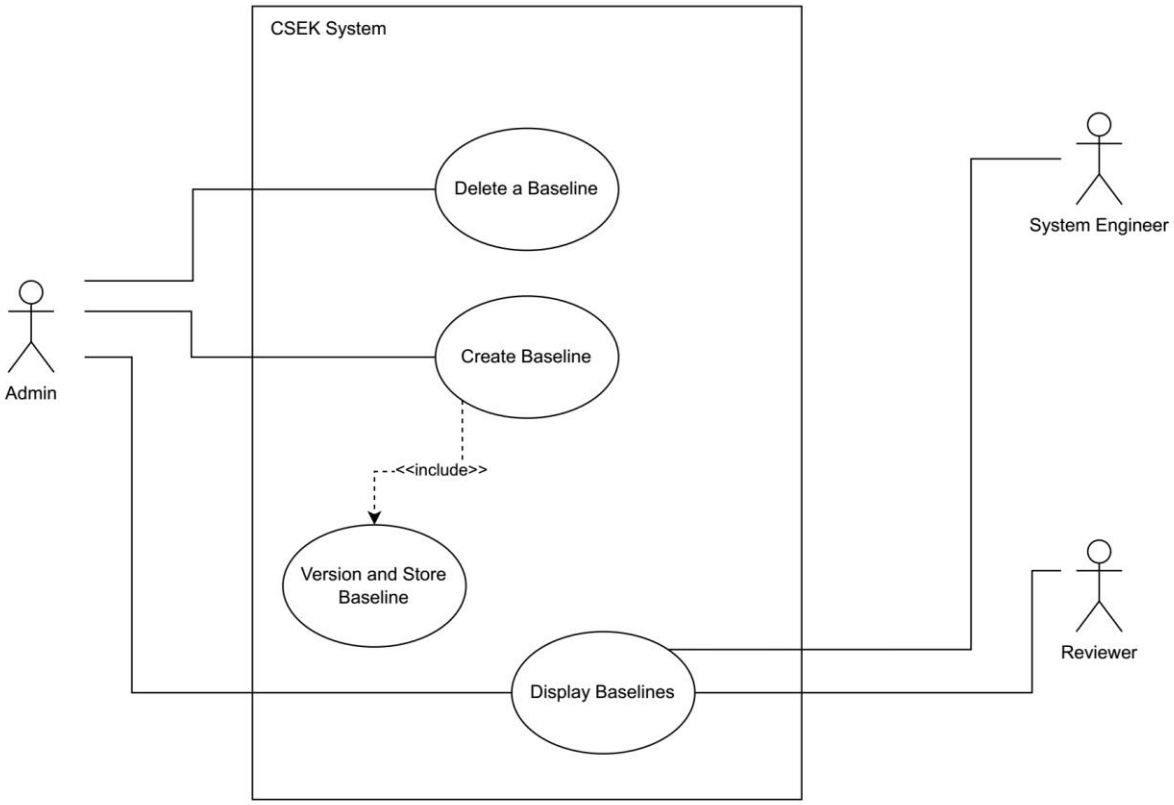


Figure 5 Use Case-5

UC-5.1: Create Baseline
Description: The system shall allow admin to create a frozen version of the requirements at a specific point in time of the entire project.
Actors: Admin
Preconditions: Must login to the system as Admin role.
Postconditions: The system shall provide versioning for baselines, allowing multiple baselines to be created and stored.

UC-5.2: Delete a Baseline
Description: The system shall allow admin to delete obsolete baselines after proper review.
Actors: Admin
Preconditions: Must login to the system as Admin role.
Postconditions: Selected baselines will be removed from storage.

UC-5.3: Display Baseline
Description: The system shall display to users for comparing baselines to identify changes between versions.
Actors: Admin, System Engineer, Reviewer
Preconditions: Baseline must be created already.
Postconditions: Requirements releases will be displayed together column by column for easy comparison.

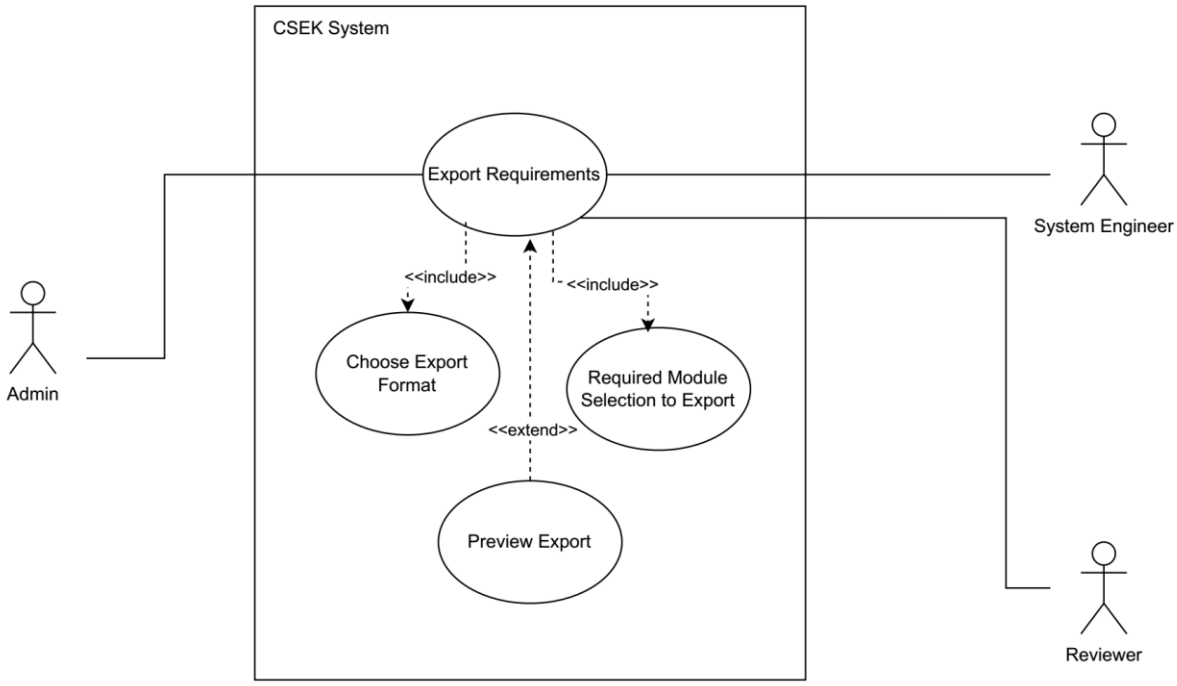


Figure 6 Use Case-6

UC-6.1: Export Requirements
Description: The system shall allow users to export requirement modules in a hierarchical structure.
Actors: Admin, System Engineer, Reviewer
Preconditions: The system allows user to select requirement module and export format. Also, provide a preview option before exporting the document.
Postconditions: The system exports selected modules as MS Word format.

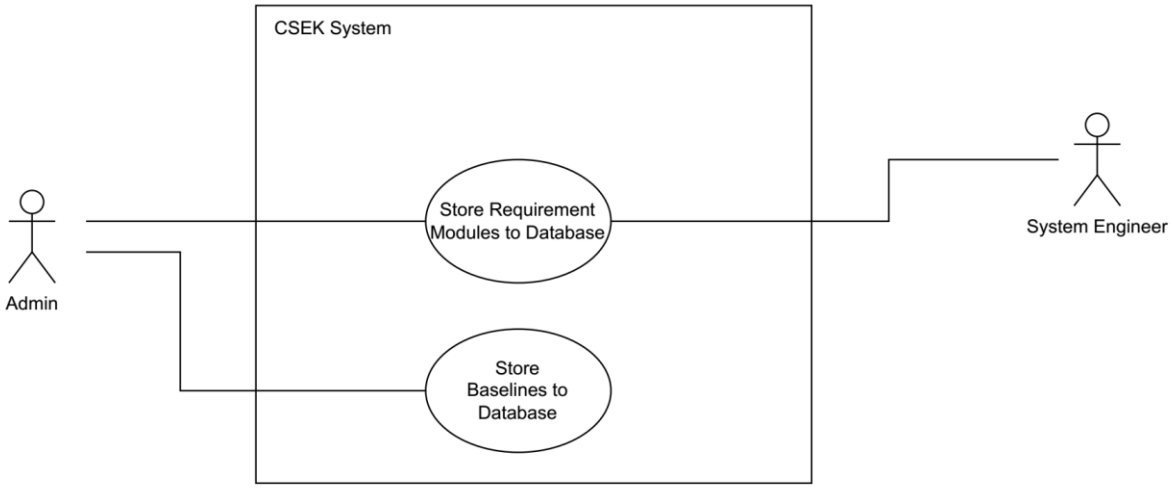


Figure 7 Use Case-7

UC-7.1: Store Requirement Modules to Database
Description: The system shall allow system engineers and admins to save all modules to the PostgreSQL database.
Actors: Admin, System Engineer
Preconditions: Must login to the system as Admin or System Engineer role.
Postconditions: Ensures that all modules (user, system and subheading requirements) are securely stored and accessible for future reference.

UC-7.2: Store Baselines to Database
Description: The system shall allow admins to save all baselines to the PostgreSQL database.
Actors: Admin, System Engineer
Preconditions: Must login to the system as Admin role.
Postconditions: Enables secure storage of baseline configurations for version control and traceability.

3.5. System Attributes

3.5.1. Portability

- The CSEK Requirements Management System has been developed as a web-based application and is optimized to work in modern browsers.
- The system works in a way that is compatible with any operating system, as only one browser is required.

3.5.2. Performance

- Requirements queries and database operations should be completed in maximum 2 seconds.
- Reporting processes are optimized to take a maximum of 10 seconds for projects with 1000 requirements.
- Writing to the system database should take a maximum of 5 seconds.

3.5.3. Usability

- The user interface is designed to make the system easy to learn and use.
- The system allows users to create, edit and delete requirements with a maximum of 4 clicks.
- As a result of incorrect operations, the user is presented with clear error messages and suggested solutions.
- The system menu allows users to easily navigate between requirements, reports and historical records.

3.5.4. Adaptability

- The system includes modules that can be adapted to different user needs.
- Microservice architecture is used to add new features or functionalities in the future.
- No adaptation of existing data is required because the system focuses only on static data management.

3.5.5. Scalability

- The system is designed to support up to 500 simultaneous users.
- When more user load is required, it can be easily scaled using the system's cloud-based infrastructure.
- However, the system does not require scalability for tasks configured to be accessed by a single user (e.g. individual reporting).

4.Supporting Information

4.1. Change Log

At the beginning of the project, we thought of integrating artificial intelligence technology while planning. thanks to artificial intelligence technology, system requirements would be generated from user requirements. Then, considering the technologies to be used and the legal parts, we removed the artificial intelligence integration part from our project. The reasons for this are that we do not have a dataset that can train the artificial intelligence model in this regard, which is a limitation for us in the software part. At this point, finding companies dataset is a legal constraint.

SOFTWARE DESING DOCUMENT

1. Introduction

1.1. Purpose

The purpose of this document is to provide a detailed overview of the design of the CSEK Requirements Management System. This document forms the basis for the design of the system, which aims to meet the functional and technical requirements of the project and optimize the requirements management processes of users in various sectors.

The system aims to provide a cost-effective, customizable alternative to existing tools with high licensing costs, such as IBM Doors. The system must be designed to make it easy to monitor the impact of requirements on each other through bidirectional traceability. The rolebased access feature shall allow users to access only the features that they are authorized to access. In this manner, the system shall increase data security and minimize possible user errors. Also, the system shall be designed to provide simultaneous access for up to 500 users. The cloud-based structure must ensure scalability and create a suitable architecture for future expansions.

The project must be developed under national security standards to address the needs of different industries. In addition to satisfying security needs, this design shall allow requirements processes to be more organized and efficient. Users can easily create, track, edit, and report requirements. The system also must allow the creation of professional reports from requirements with export options in MS Word and PDF formats.

The design aims to ensure that the system is sustainable and developable in the long term by focusing on future expandability goals. The microservice architecture must facilitate adding new features and making changes to the system. This adaptive methodology must facilitate project sustainability and industrial adaptation.

This document for designing the CSEK Requirements Management System is a guide and reference for software engineers, project managers, and other stakeholders. It aims to help the project team achieve its goals by exhaustively discussing all technical and architectural aspects of the system design. This document provides the necessary technical infrastructure for the project's successful execution.

1.2. Scope

The CSEK Requirements Management System must be designed to provide a robust, safe, and cost-effective solution that can be used in various industries. This system must offer a localized, accessible alternative by addressing the usage challenges of existing requirements management tools such as IBM Doors.

The system must provide a comprehensive tool for managing user, system, and subheading requirements. It must allow users to perform basic tasks such as creating, editing, categorizing, and monitoring requirements. It must also allow users to benefit from advanced features such as bidirectional traceability, which

shall provide users with an understanding of how requirements affect each other. The system must also secure data with role-based access control.

The platform must be web-based and designed to run on browsers. This must allow the system to be accessible from anywhere with an internet connection. The system must adopt a modular microservice architecture and use advanced technologies such as Netflix Eureka for dynamic service discovery, Kafka for event-driven communication, and gRPC for inter-service communication. This architectural approach must ensure the system is highly scalable, adaptable, and future-proof.

The system must store data securely in the PostgreSQL database. Redis must be integrated to increase system performance and reduce database load. With Redis, frequently accessed data must be cached. The system must encrypt data at rest using AES-256 and data in transit using TLS 1.3 to ensure data security. This must protect sensitive information.

In terms of performance, the system must be optimized to support up to 500 concurrent users to satisfy the needs of organizations of different sizes. The system's cloud-based infrastructure must enable it to cater to requirements and users as they grow. Reporting features must also be important, allowing users to export their requirements in MS Word and PDF while maintaining their hierarchical structure. This feature must support the creation of professional documentation.

The CSEK Requirements Management System must be designed by national security standards and provide an ideal option for various sectors. The system must focus on sustainability and adaptability, allowing for future improvements. Features such as adding additional formats to the reporting section, generating system requirements from user requirements by integrating artificial intelligence technology, allowing the user to enter requirements tests, and allowing the administrator to assign tasks to users must be left open for future development. Although these and similar features must be not included in the initial design, they can be added during the development phase without problems using the microservice architecture.

CSEK Requirements Management System must enable organizations to manage their requirements more efficiently, reliably, and customizable by providing a requirements management platform that is secure, scalable, and customizable to user needs. Addressing both technical and operational challenges must enable organizations to manage their requirements in a more efficient, reliable, and customizable way.

1.3. Glossary

- AES-256

  Advanced Encryption Standard with a 256-bit key length, used for securing data at rest.
- API Gateway

  A centralized entry point that routes user requests to various microservices, typically using REST protocols.

- Bidirectional Traceability

  A system feature that allows tracking of how requirements impact each other.
- Keycloak

  An open-source software solution for managing authentication and authorization processes.
- Microservice Architecture

  A software development architecture where an application is divided into independently deployable and manageable small services.
- Role-Based Access Control (RBAC)

  An access control mechanism where permissions are granted based on user roles.
- TLS1.3

  The latest version of the Transport Layer Security protocol ensures encrypted data transmission.
- Snapshot

  A saved state of a system at a specific point in time, is often used for backup and version control.

1.4. Overview of Document

This document is provided as a Software Design Document for the CSEK Software Requirements Management System. The system must be designed to develop a local requirements management tool for various sectors. The system must aim to address challenges such as the high licensing costs of existing tools, local compatibility issues, and limited customization options. In this context, the project must offer a secure and scalable solution that is both cost-effective and suitable for user needs.

This document describes the software design of the CSEK Requirements Management System comprehensively. The document details the architectural structure, design principles, modular approach and technologies to be used. The project must be developed using microservices architecture and optimized to satisfy security and performance requirements. The system must be designed to appeal to various organizations, with a user-friendly interface and robust data management features.

The main design objectives of the system include providing bi-directional traceability of user requirements, increasing the level of security with role-based access control, and providing detailed reporting features. The system, supported by technologies such as PostgreSQL database, Redis caching mechanism, and Kafka, must be structured to provide high performance and data security.

This document aims to give software engineers, project managers, and other stakeholders a clear understanding of the project's design and to provide guidance at each stage of the software development process. Readers can find the overall architecture of the system, its technical components, and the motivations underlying the design decisions in this document.

Introduction part describes the purpose of the document and the objectives of the system. The Design Approach section summarizes the methods, steps and general approaches used in the design process. The Architectural Design section describes the microservice architecture of the system, the technological infrastructure and the tools used. In the User Interface section, the interface that the user will encounter is visualized and explained.

1.5. Motivation

We are a group of senior computer engineering students who aim to develop a requirements management tool for various industries. As a group, we wanted to be interested in a field that would provide us with more sectoral knowledge, develop us using different technologies, and fill a gap in the industries.
The main motivation for developing the CSEK Requirements Management System is to address problems arising from the high cost and unsuitability of existing requirements management tools for local needs in various industries. This project aims to improve users' requirements management processes, increase traceability of changes, and improve operational efficiency. The motivation behind the project is to create sustainable value in various industries by providing an innovative and accessible solution to technical and operational challenges. As a team, since we are not very familiar with the technologies we will use in our project, we plan to go through a process in which we can progress by working to learn these technologies and getting the necessary support. In this way, we plan both to manage the project process healthily and to improve ourselves in these contexts.

2. Architecture Design

2.1. Design Approach

The design approach of the CSEK Requirements Management System includes several steps to manage the software development process in an organized way. This approach aims to satisfy both technical and operational needs, providing a flexible structure for future development and extensions. The design process includes the following steps:

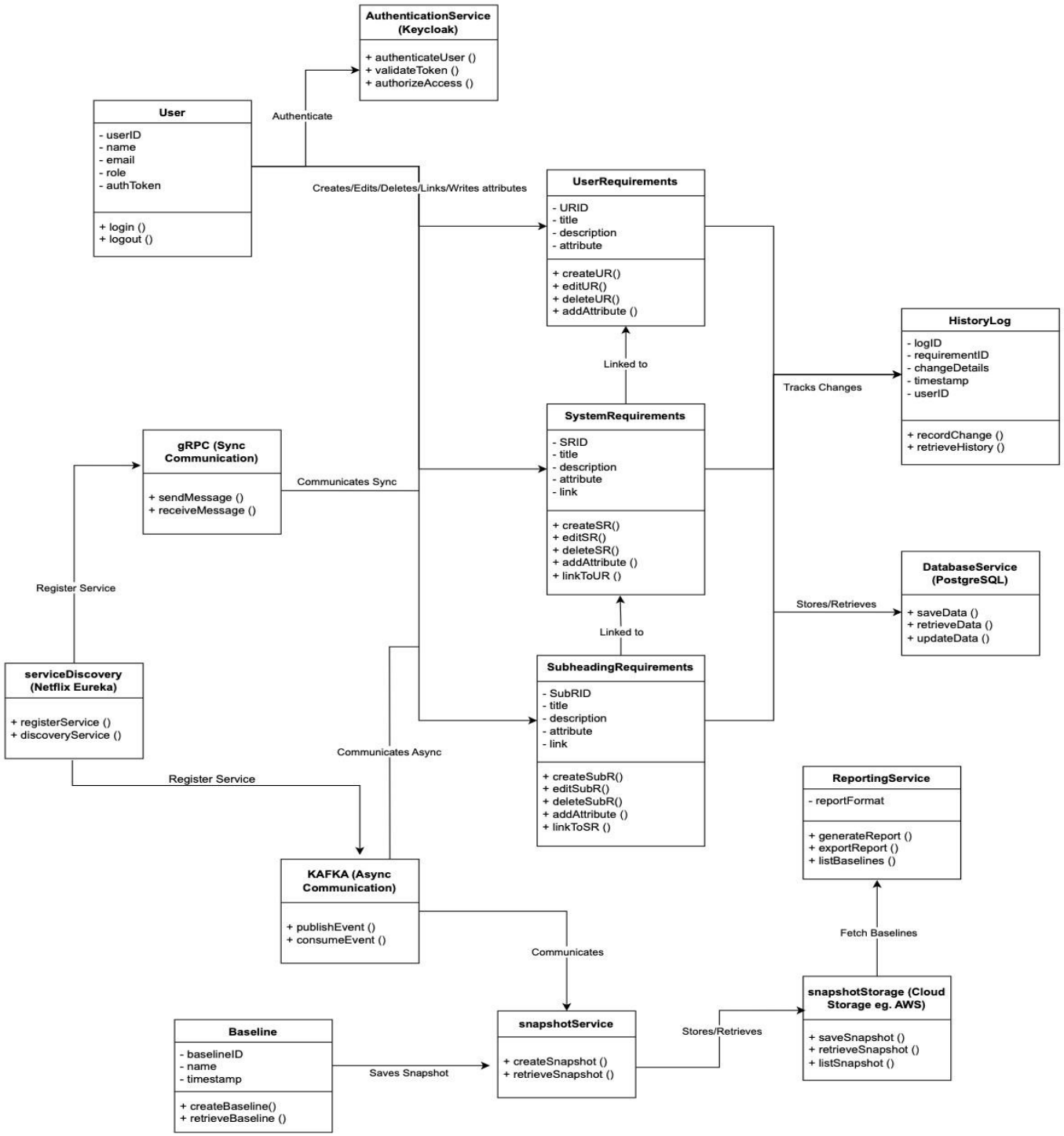
1. Literature Review: At the beginning of the project, a literature review was conducted to research the field and examine existing studies. In this context, the shortcomings of the existing studies and the aspects that could be improved were observed. In addition, the key features of our project were better recognized.
2. Requirements Gathering: This phase enabled the identification of user needs and requirements. This process focused on addressing the needs of a requirements management tool that could be used in various sectors. The needs of the user roles were analyzed separately. Features

such as bi-directional traceability, customizable reporting, and role-based access control were included in the project.

3. Software Architecture Design: Microservice architecture must be preferred as it is thought to be suitable for the requirements of the project. Each module must be designed as an independent service and Netflix Eureka, gRPC and Kafka must be used for interaction between them. This structure must increase the modularity of the system, facilitating both maintenance and extensibility.
4. Database Design: Data security and accessibility must be one of the important points in system design. A secure and scalable database infrastructure must be built using PostgreSQL. All requirements and user data must be stored securely in this database. Performance optimization must be ensured by using Redis for frequently accessing data. Also, data must be encrypted with AES-256 encryption at rest and TLS 1.3 protocol during transmission.
5. Performance Design: The system must be designed to meet performance needs with a cloud-based architecture. This must ensure scalability and high performance. In addition, it must be optimized to support 500 users simultaneously
6. User-Friendly Interface and Web-Based Design: The system must be designed with a user-friendly interface for administrators, system engineers and reviewers. The web-based structure must provide a high accessibility design that can be accessed from browsers.
7. Future Expandability: The system must be designed with a flexible architecture so that new features can be easily integrated. Future development steps may include AI-based requirements generation, more advanced reporting options, and different integration features.

CSEK's Requirements Management System must use this approach to deliver a design that is high-performing and tailored to the user's needs.

2.2. Class Diagram



This diagram illustrates the structure and interaction of various components within a CSEK Requirement Management System's architecture designed for managing requirements and snapshots, using modern communication and service technologies. It includes a User entity with authentication and authorization managed by AuthenticationService (Keycloak). ServiceDiscovery (Netflix Eureka) handles service registration and discovery. The system incorporates synchronous communication through gRPC and asynchronous communication via Kafka. Core modules like UserRequirements, SystemRequirements, and
SubheadingRequirements manage hierarchical requirement relationships, enabling creation, editing, deletion, and linking functionalities. The HistoryLog tracks changes, while Baseline and snapshotService focus on capturing and managing system snapshots stored in snapshot storage (e.g., AWS). Additionally, DatabaseService (PostgreSQL) provides CRUD operations, and ReportingService generates and exports reports.

2.3. ARCHITECTURAL DESIGN

• 2.3.1. Overview

The system uses a microservices architecture. This guarantees scalability. Maintenance and use of independent services Each microservice is designed according to clean architecture principles. It focuses on separating concerns. It provides a clear boundary between core business logic and external dependencies. The system is implemented on Kotlin and uses the Spring Boot framework, leveraging modern tools for caching, communication, and authentication.

• 2.3.2. Technology Stack

Programming Language: **Kotlin**
Framework: **Spring Boot**
Database: **PostgreSQL** (for requirements and user data storage)
Caching: **Redis** (for improving performance) Service Communication:

gRPC: For synchronous communication among User Requirements, the System Requirements, and Subheading Requirements services.

Kafka: For asynchronous communication between Requirements Microservices and Snapshot Service.

Service Discovery: **Netflix Eureka** (to manage microservices and their dynamic locations).
Authentication & Authorization: **Keycloak** integrated via a REST API Gateway.

Snapshot Storage: Snapshots are stored in a cloud storage solution (e.g., AWS).

- ### 2.3.3. Microservices

1. User Requirement Service

Handles operations related to user requirements. Communicates with other requirement services via gRPC, and communicates with snapshot service via Kafka.

2. System Requirement Service

Manages system requirements and links to user requirements. Communicates with other requirement services via gRPC, and communicates with snapshot service via Kafka.

3. Subheading Requirement Service

Manages subheading requirements and links to system requirements. Communicates with other requirement services via gRPC, and communicates with snapshot service via Kafka.

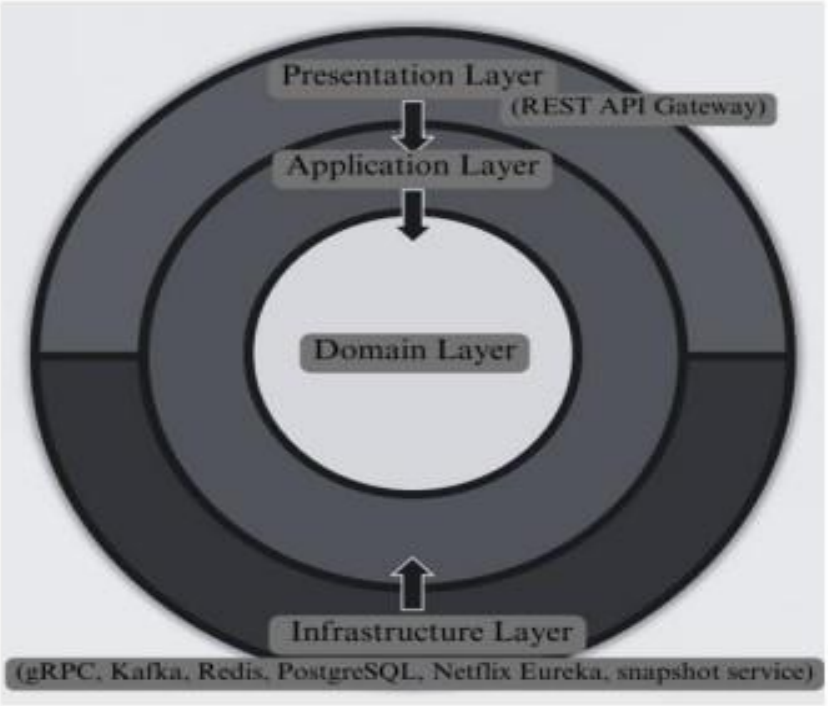
4. Snapshot Service

Stores snapshots in cloud storage and interacts with requirements services for versioning using Kafka.

5. Reporting Service

Generates reports by retrieving data from the snapshot storage.

- ### 2.3.4. Clean Architecture Diagram



1. Presentation Layer: This is the layer where users interact with the system through interfaces like a web application or API.
2. Application Layer: Manages the application's use cases by coordinating between the Presentation Layer and the Domain Layer. It handles the business process logic but does not include detailed business rules.
3. Domain Layer: Contains the core business logic and rules of the application, independent of external systems or technologies.
4. Infrastructure Layer: Provides technical details and implementations that support the application but are not part of the core business logic.

- ### 2.3.5. Service Communication

1. API Gateway (REST):
   - Centralized entry point for the application, handling user requests through RESTful communication and routing them to appropriate microservices.
   - Integrated with Keycloak for user authentication and authorization using REST protocols.
2. gRPC:
   - Facilitates synchronous communication among the User Requirements, System Requirements, and Subheading Requirements services to ensure efficient data exchange.
3. Kafka:
   - Manages asynchronous messaging between Requirements Microservices and the Snapshot Service, ensuring reliability in data exchange.
4. Redis:
   - Acts as a caching layer to optimize database queries and improve overall system performance. 5. Netflix Eureka:
   - Provides service discovery, allowing microservices to locate and communicate with each other dynamically.

2.4. Activity Diagram

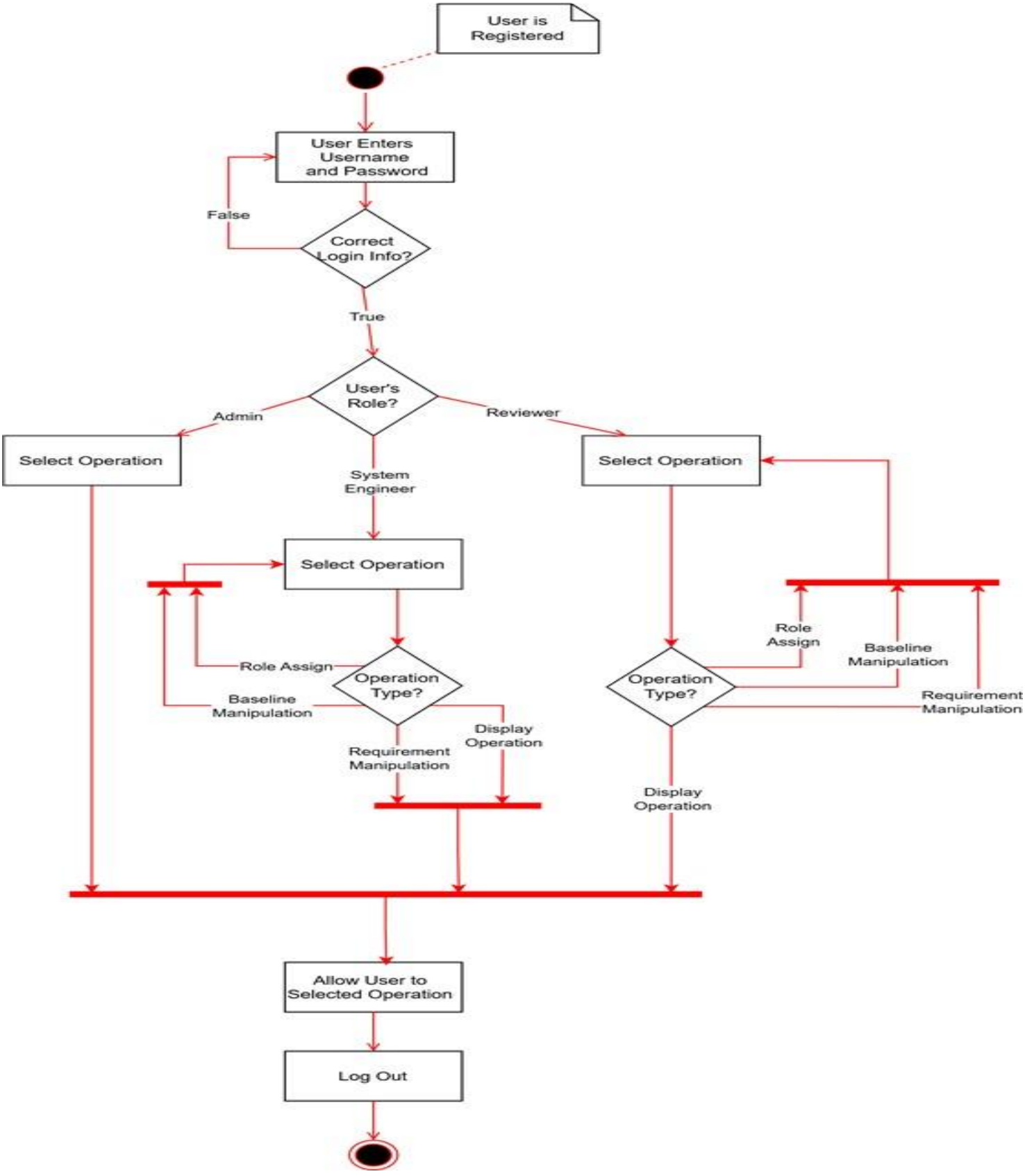
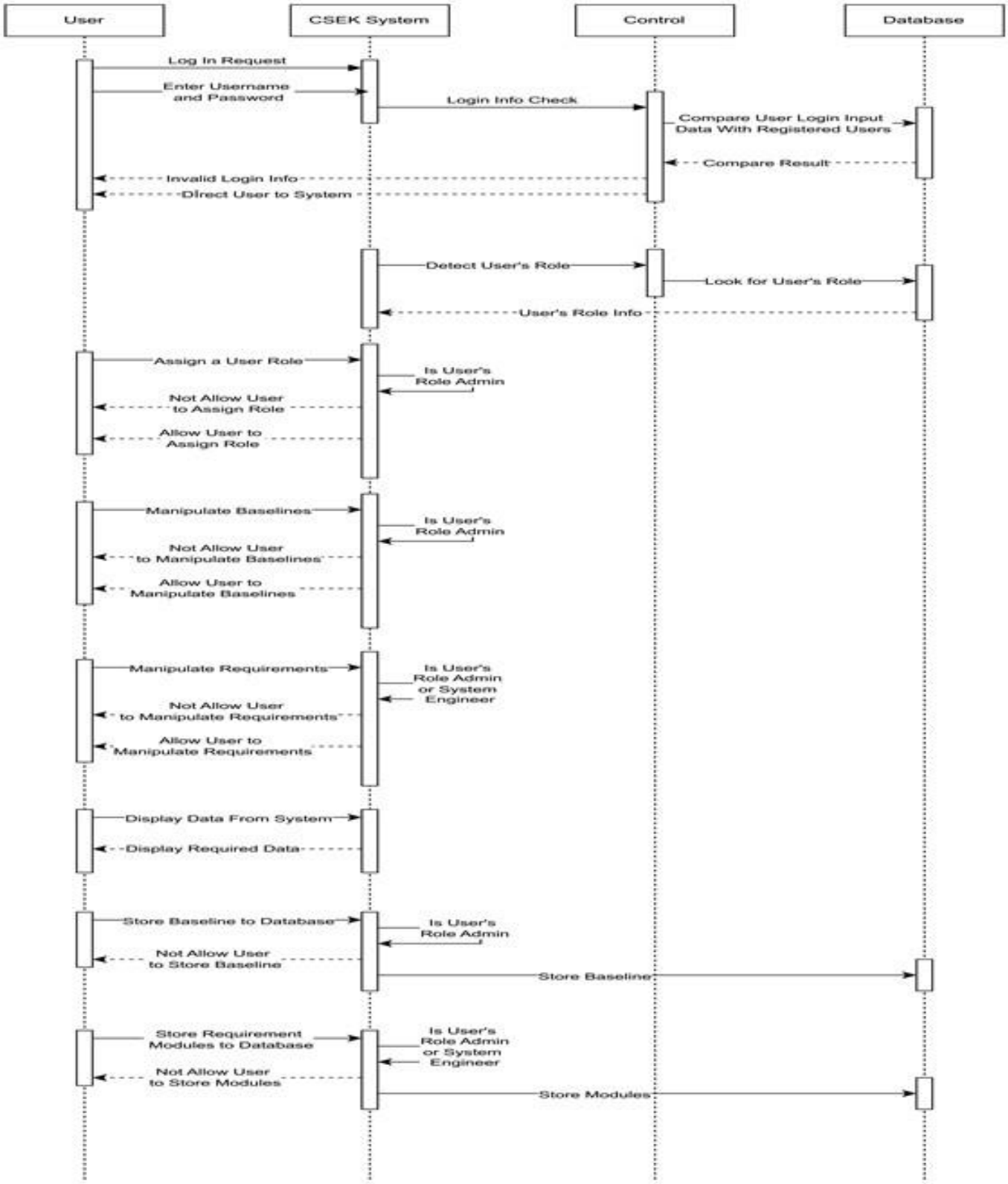


*Figure i*s an activity diagram illustrating how our program operates in a scenario where a user logs into the program, performs an operation, and then logs out. When a user successfully logs in, their role is checked. If the user's role does not have permission to access the desired operation, access is denied. While users with the Admin role have access to all operations, the access for System Engineer and Reviewer roles is more restricted. The Reviewer can only perform view operations on existing items in the system, whereas the System Engineer can also make modifications related to requirements in addition to the Reviewer’s capabilities.

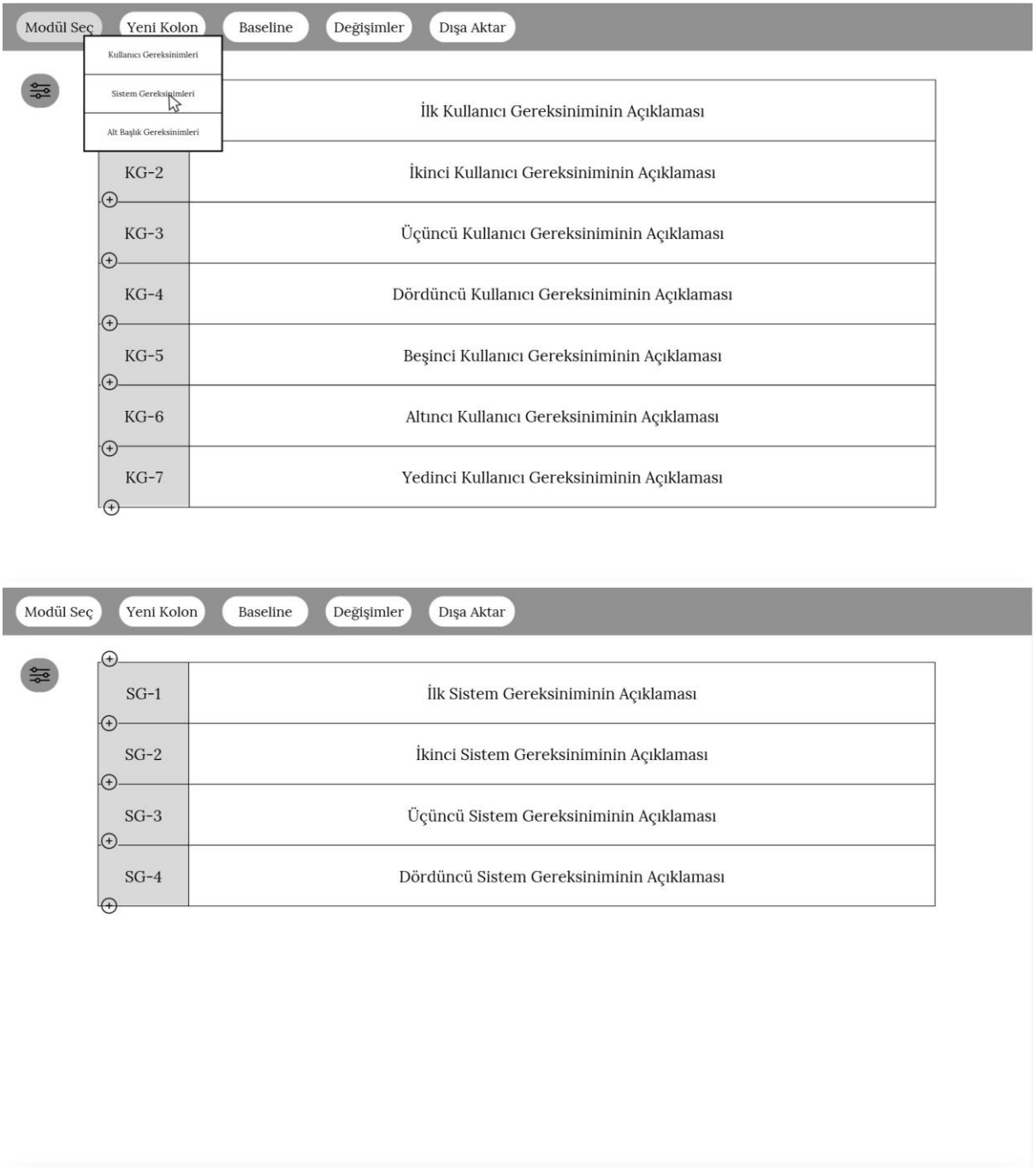
2.5. Sequence Diagram



The sequence diagram in *Figure* outlines the interaction between a user and the system, beginning with a login request where the entered username and password are validated against the database. Upon successful authentication, the system retrieves the user's role to determine access rights. Users with the "Admin" role can assign roles, manipulate baselines, and store them in the database, while both "Admin" and "System Engineer" roles can manipulate requirements and store requirement modules. Users without the necessary roles are restricted from performing these operations. Additionally, all users can view data stored in the system.

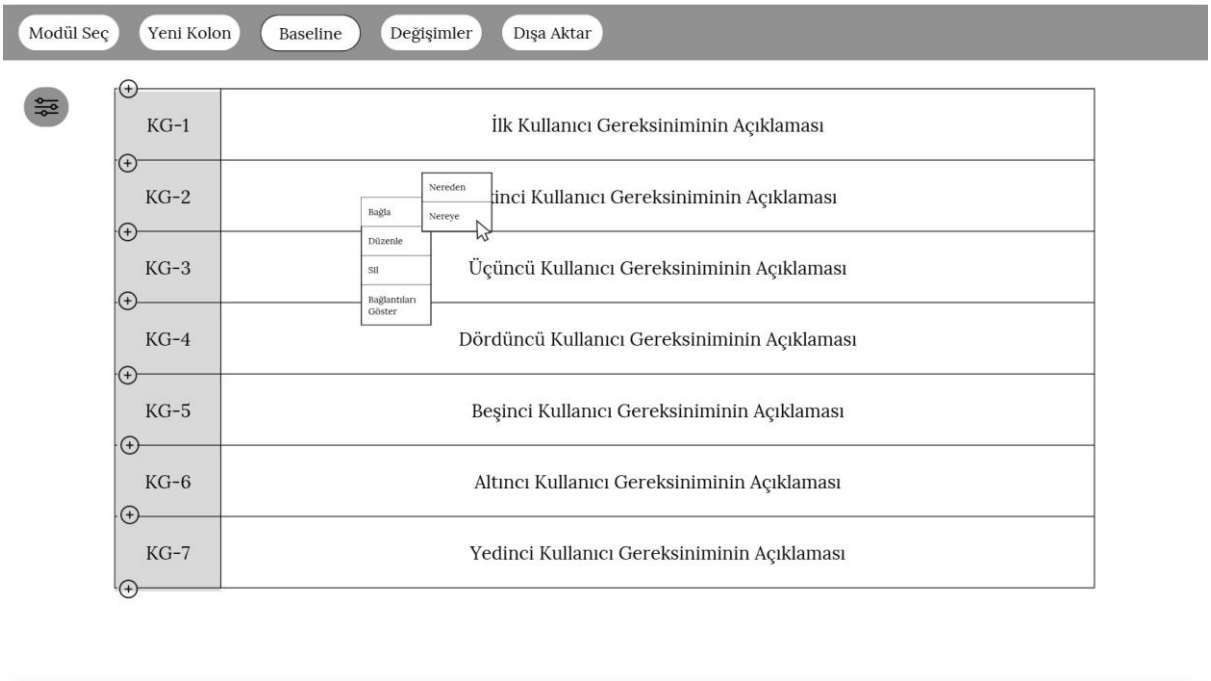
3.User Interface Design

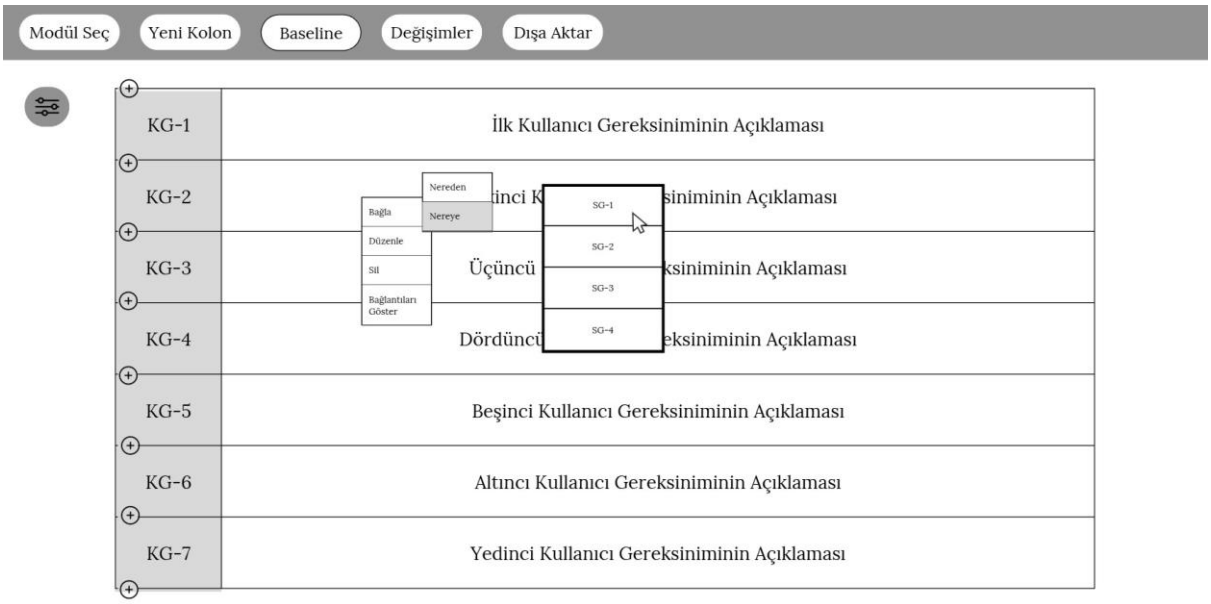
On this page User Reqiurements are displayed in a sequential order.



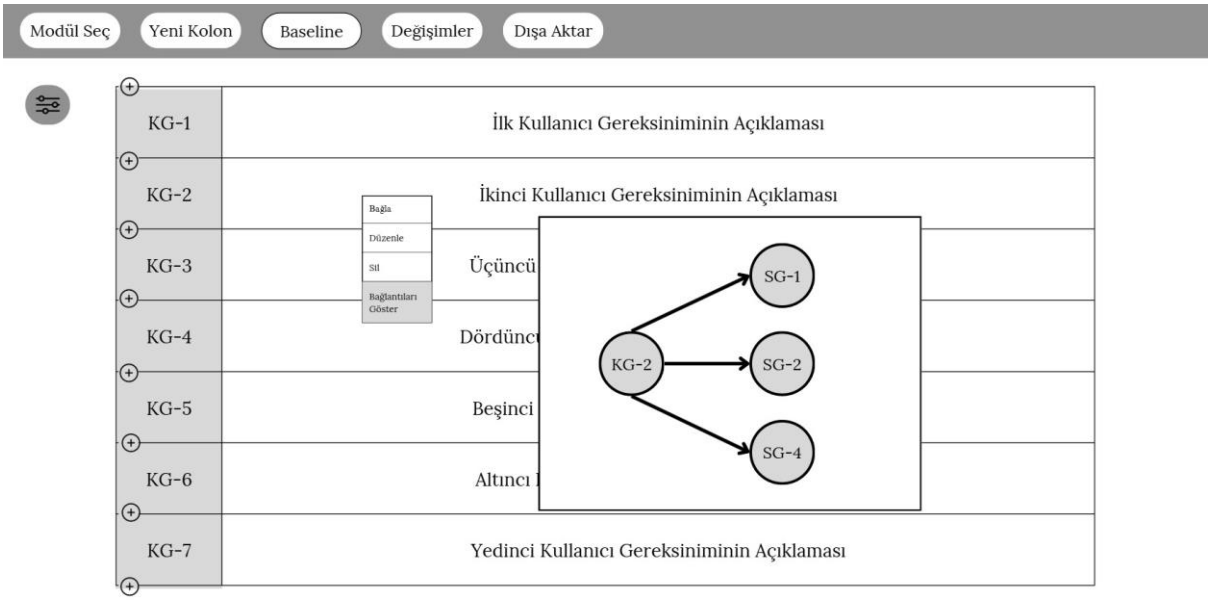
System requirements can be seen with the Select Module option


Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması
Bağla
Düzenle
Sil
Bağlantıları Göster
KG-3
Üçüncü Kullanıcı Gereksiniminin Açıklaması
KG-4
Dördüncü Kullanıcı Gereksiniminin Açıklaması
KG-5
Beşinci Kullanıcı Gereksiniminin Açıklaması
KG-6
Altıncı Kullanıcı Gereksiniminin Açıklaması
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması

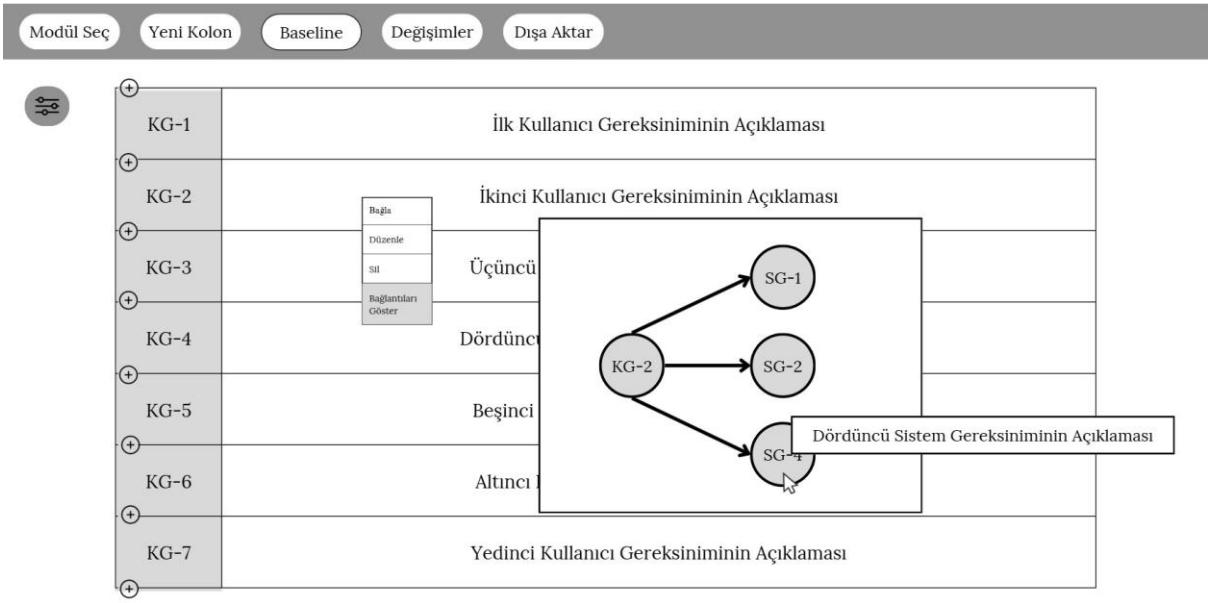

Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
Nereden
Nereye
Bağla
Düzenle
Sil
Bağlantıları Göster
KG-3
Üçüncü Kullanıcı Gereksiniminin Açıklaması
KG-4
Dördüncü Kullanıcı Gereksiniminin Açıklaması
KG-5
Beşinci Kullanıcı Gereksiniminin Açıklaması
KG-6
Altıncı Kullanıcı Gereksiniminin Açıklaması
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması



From the right-click menu, the user can specify where the requirement comes from and where it will go.



Tracebility can be done with the Show Connections option.


Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması
KG-3
Üçüncü
KG-4
Dördünc
KG-5
Beşinci
KG-6
Altıncı
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması
Bağla
Düzenle
Sil
Bağlantıları Göster
SG-1
SG-2
SG-4
Dördüncü Sistem Gereksiniminin Açıklaması

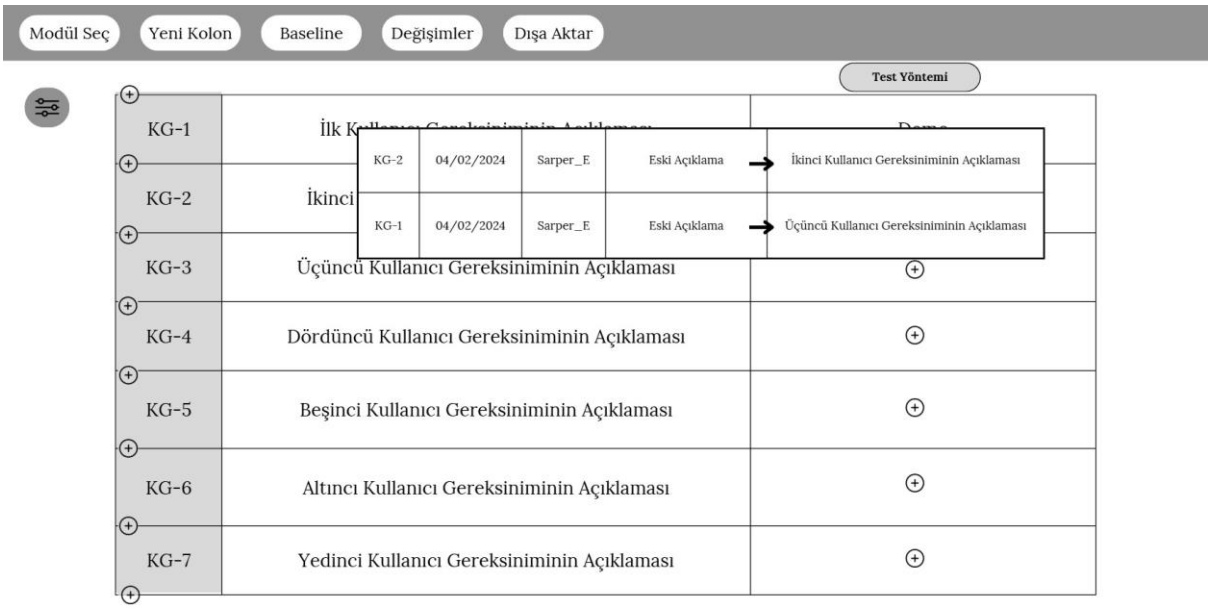

Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması
KG-3
Üçüncü Kullanıcı Gereksiniminin Açıklaması
KG-4
Dördüncü Kullanıcı Gereksiniminin Açıklaması
KG-5
Beşinci Kullanıcı Gereksiniminin Açıklaması
KG-6
Altıncı Kullanıcı Gereksiniminin Açıklaması
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması


Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
Sütun Başlığı Giriniz: |
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması
KG-3
Üçüncü Kullanıcı Gereksiniminin Açıklaması
KG-4
Dördüncü Kullanıcı Gereksiniminin Açıklaması
KG-5
Beşinci Kullanıcı Gereksiniminin Açıklaması
KG-6
Altıncı Kullanıcı Gereksiniminin Açıklaması
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması

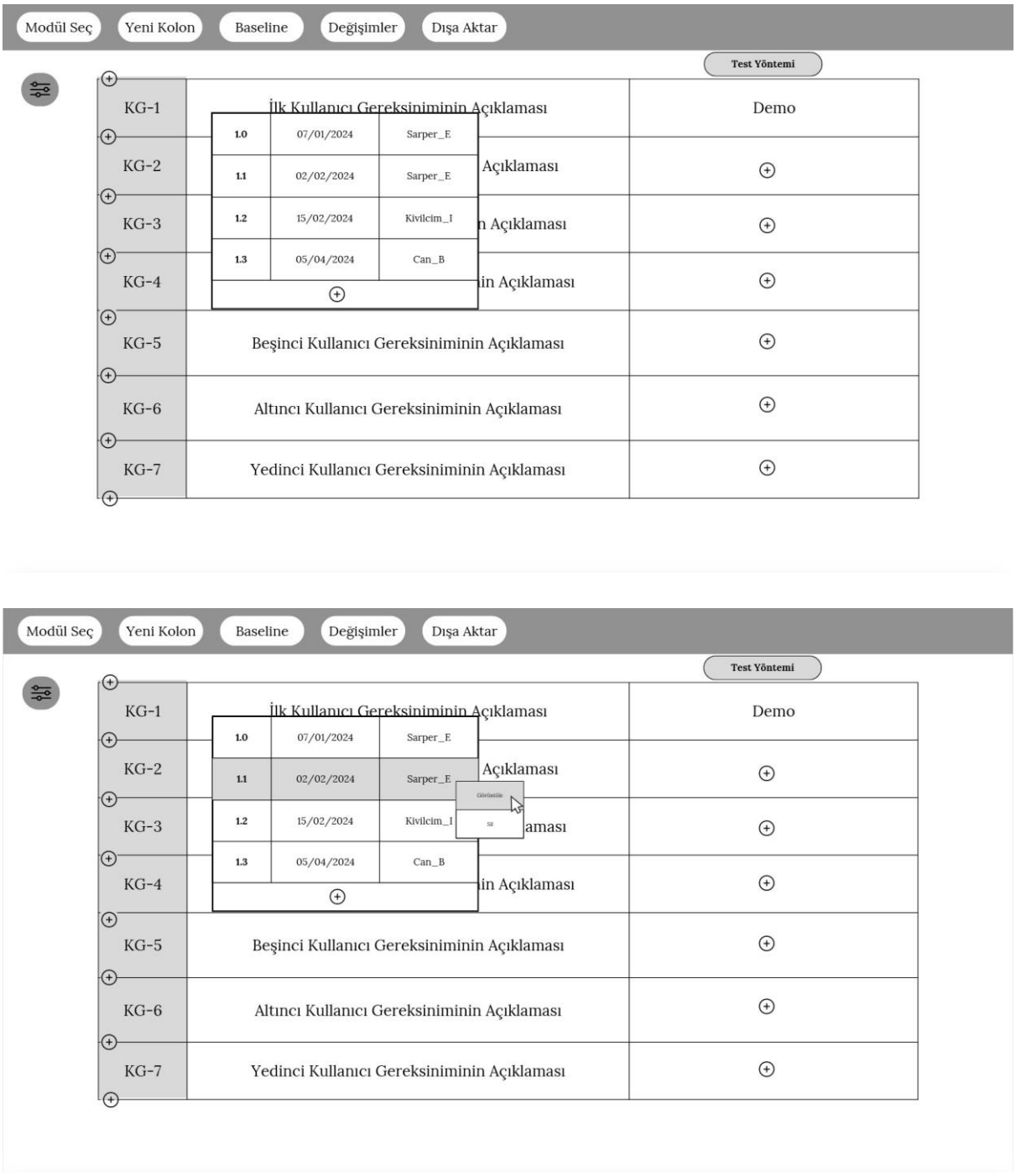

Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
Sütun Başlığı Giriniz: Test Yöntemi
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması
KG-3
Üçüncü Kullanıcı Gereksiniminin Açıklaması
KG-4
Dördüncü Kullanıcı Gereksiniminin Açıklaması
KG-5
Beşinci Kullanıcı Gereksiniminin Açıklaması
KG-6
Altıncı Kullanıcı Gereksiniminin Açıklaması
KG-7
Yedinci Kullanıcı Gereksiniminin Açıklaması

New Column option creates a new column with the entered title.

By pressing the + key, the rows in the forming column can be filled.



Changes made can be checked with the Changes option.



With the Baseline option, previous records are displayed and the current situation can be compared with the desired past record.


Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması


Modül Seç
Yeni Kolon
Baseline
Değişimler
Dışa Aktar
KG-1
İlk Kullanıcı Gereksiniminin Açıklaması
KG-2
İkinci Kullanıcı Gereksiniminin Açıklaması

The + sign above the requirements allows a new requirement to be added to the desired line.

CONCLUSION

CSEK Requirements Management System has been developed as a local, cost-effective, customizable requirements management tool. Designed to be built using different software technologies, this tool provides an accessible and functional solution. It is aimed to localize IBM DOORS, which is a frequently used tool in requirements management, and to avoid its disadvantages.

Our project is planned to implement bidirectional traceability, role-based access control, a user-friendly interface, and reporting features to improve the requirements management process. In addition, the fact that the system provides a secure structure must ensure increased usability in various sectors.

According to this report, the system is designed to allow users to manage requirements in an efficient manner. In addition, the system is intended to be open to future development and improvements using a microservice architecture.

As a result, the CSEK Requirements Management System should provide a solution that is effective in software development processes and appropriate to user needs. It is aimed to contribute to software development processes in various industries as a sustainable and developable system in the long term.

REFERENCES

[1] Aragon, K. M., Eaton, S. M., McCornack, M. T., Shannon, S. A., & Sandia National Laboratories. (2014). Generalized information Architecture for managing requirements in IBM's Rational DOORS® application. In SANDIA REPORT (No. SAND2014-20563). Sandia National Laboratories. http://www.osti.gov/bridge

[2] Kuutti, T. (2019). Comparing Requirements Management Tools – IBM Rational DOORS & HP ALM. In Metropolia University of Applied Sciences, Metropolia University of Applied Sciences.

[3] Al-Hroob, A., Imam, A. T., & Al-Heisa, R. (2018). The use of artificial neural networks for extracting actions and actors from requirements document. Information and Software Technology, 101, 1–15. https://doi.org/10.1016/j.infsof.2018.04.010

[4] Hoffmann, M., Kühn, N., Weber, M., DaimlerChrysler AG Research & Technology, Bittner, M., & TU Berlin, Fak. IV, ISTI, SWT. (2004). Requirements for Requirements Management Tools. In Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04) (pp. 1090-705X) [Conference-proceeding]. IEEE.

[5] Verma, K., Kass, A., & Accenture Technology Labs. (2008). Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In ISWC 2008 (Vol. 5318, pp. 751–763) [Journal-article]. Springer-Verlag Berlin Heidelberg.

[6] Sajad, M., Sadiq, M., CRIDS (Center for Research in Distributed and Supercomputing) RIU, Naveed, K., Iqbal, M. S., CRIDS (Center for Research in Distributed and Supercomputing) RIU, & School of Computer Science, Anhui University, Hefei, China. (2016). Software Project Management: Tools assessment, Comparison and suggestions for future development. In IJCSNS International Journal of Computer Science and Network Security (pp. 31–32).

[7] draw.io - free flowchart maker and diagrams online. (n.d.). https://app.diagrams.net/

[8] Engineering Requirements Management DOORS. (n.d.). https://www.ibm.com/docs/en/engineering-lifecycle-managementsuite/doors/9.7.0?topic=overview

[9] Kafka Nedir? - Apache Kafka'ya Ayrıntılı Bakış - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/tr/what-is/apache-kafka/

[10] Marcelo, & Marcelo. (2024, October 6). IBM Rational DOORS Yazılımına Genel Bakış | Eksiksiz Kılavuz. Visure Solutions. https://visuresolutions.com/tr/ibm- kap%C4%B1lark%C4%B1lavuzu/

[11] gRPC. (n.d.). gRPC. https://grpc.io/

[12] https://medium.com/devopsturkiye/redis-nedir-ne-i%CC%87%C5%9Fe-yarar1a19ebbdb2b4

[13] *Rules to clean architecture*. (n.d.).

https://www.ssw.com.au/rules/rules-to-better-cleanarchitecture/.

[14] *UML Class Diagram Tutorial*. (n.d.).

https://www.visual-paradigm.com/guide/umlunified-modeling-language/uml-class-diagram-tutorial/

[15] https://www.canva.com/

[16] *How to Write a Software Design Document (SDD)*. (n.d.).

https://www.nuclino.com/articles/software-design-document