CANKAYA UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

SOFTWARE REQUIREMENTS SPECIFICATION

CSEK REQUIREMENTS MANAGEMENT SYSTEM

KIVILCIM IŞIK 202011006

ERAY EMİR 202011016

SARPER ERBAR 202011001

CAN METE BOZAR 202011052

CONTENTS

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.

1.5. 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

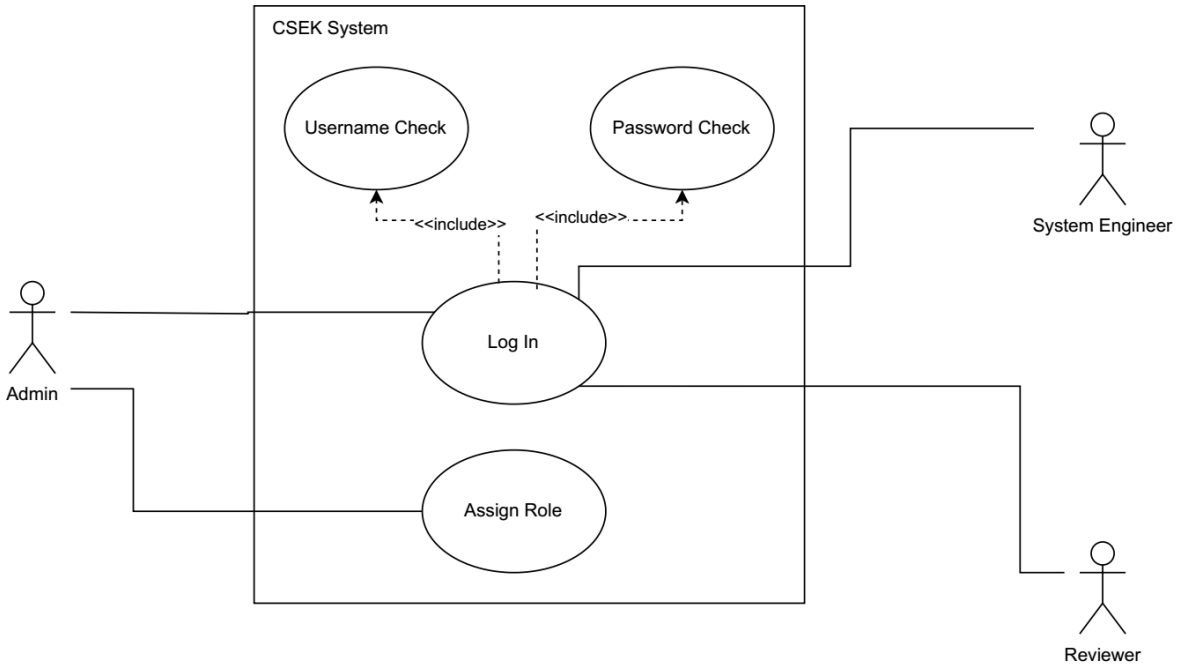

Figure 1- Use Case-1

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.

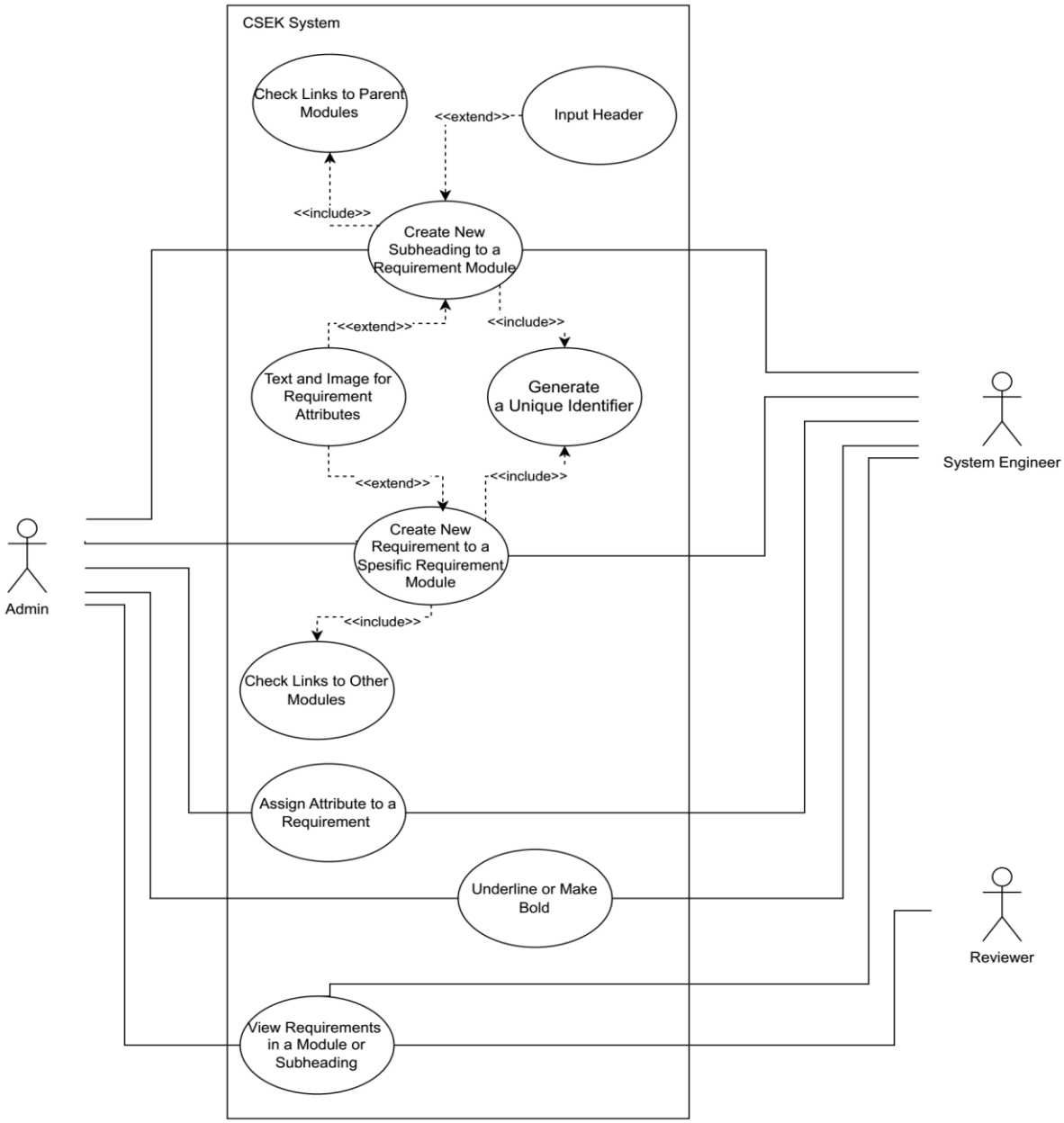

Figure 2 Use Case-2

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.

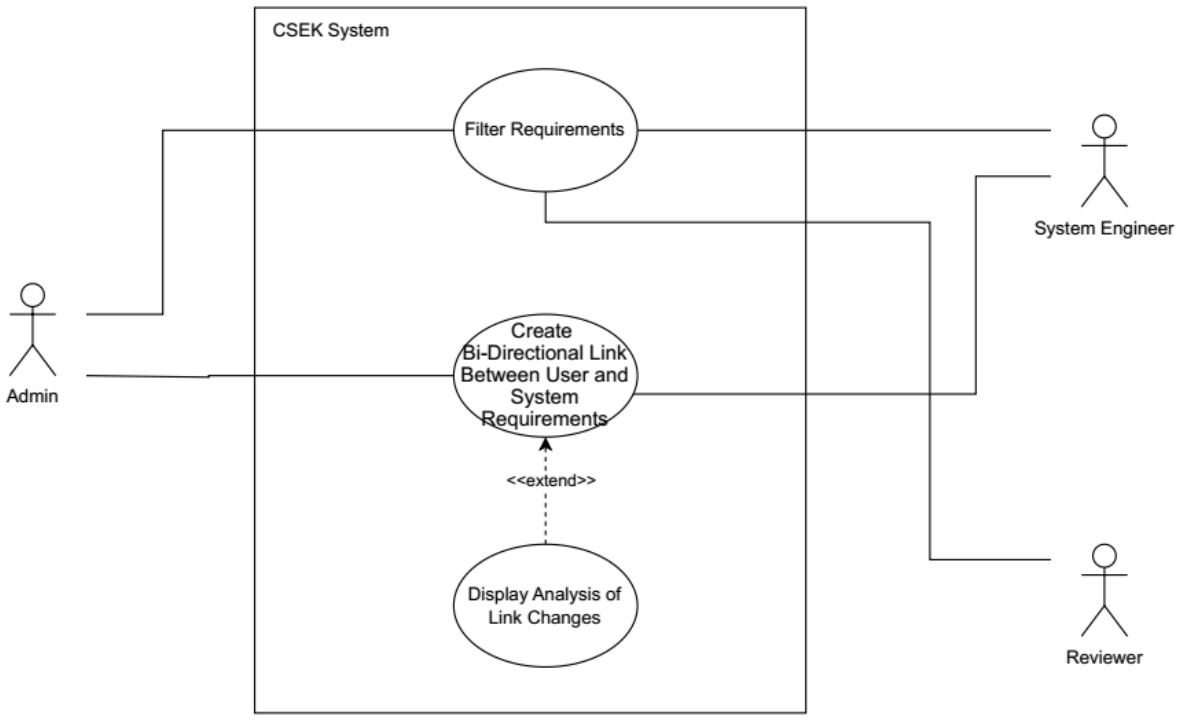

Figure 3 Use Case-3

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.

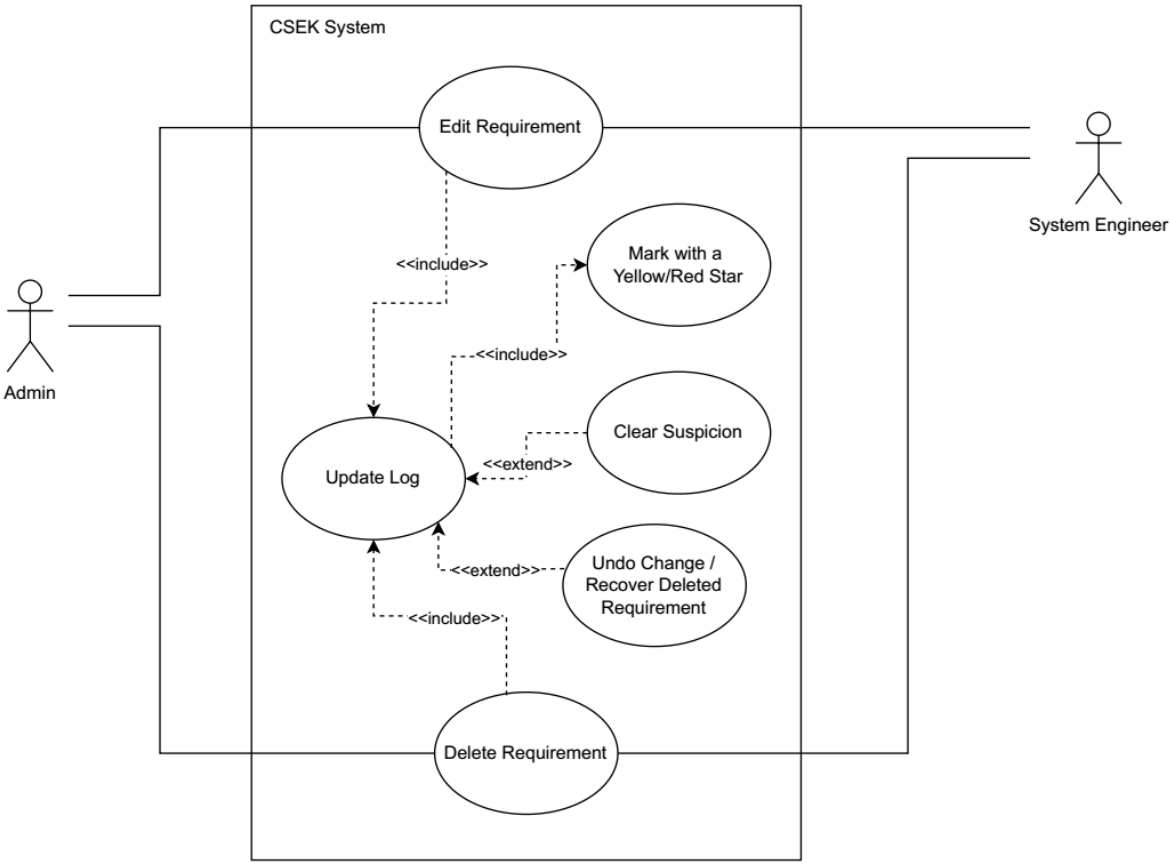

Figure 4 Use Case-4

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.

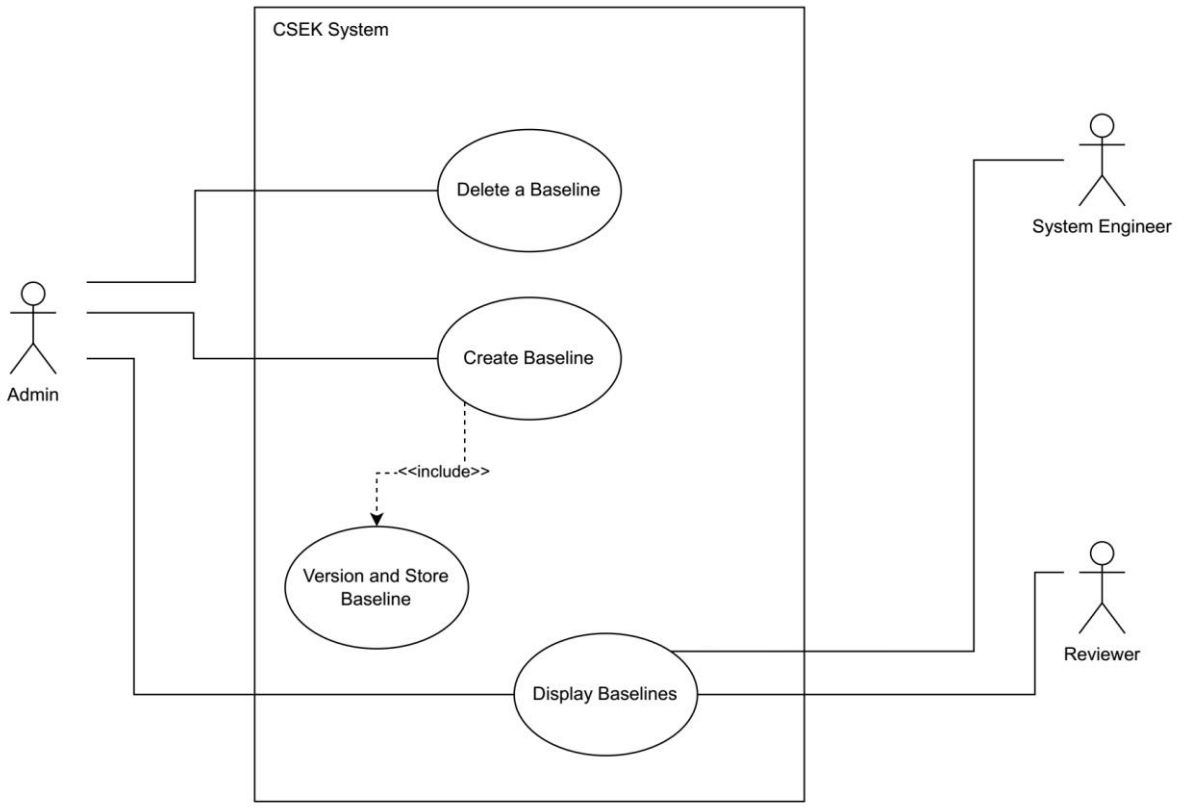

Figure 5 Use Case-5

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.

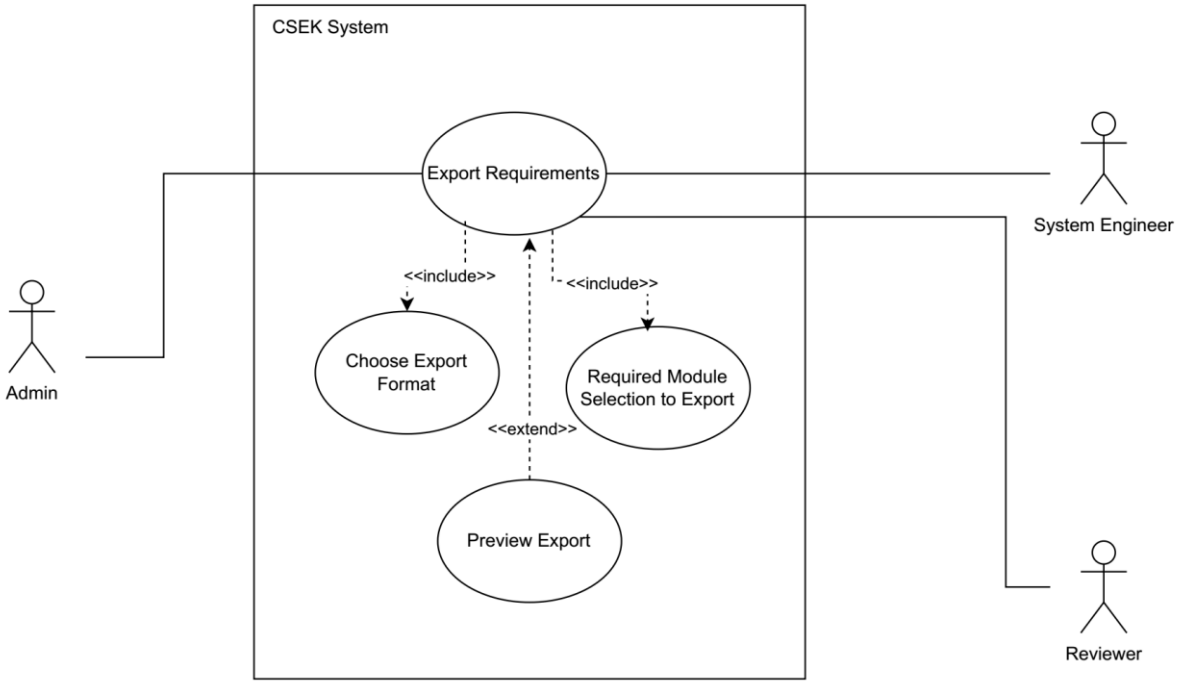

Figure 6 Use Case-6

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.

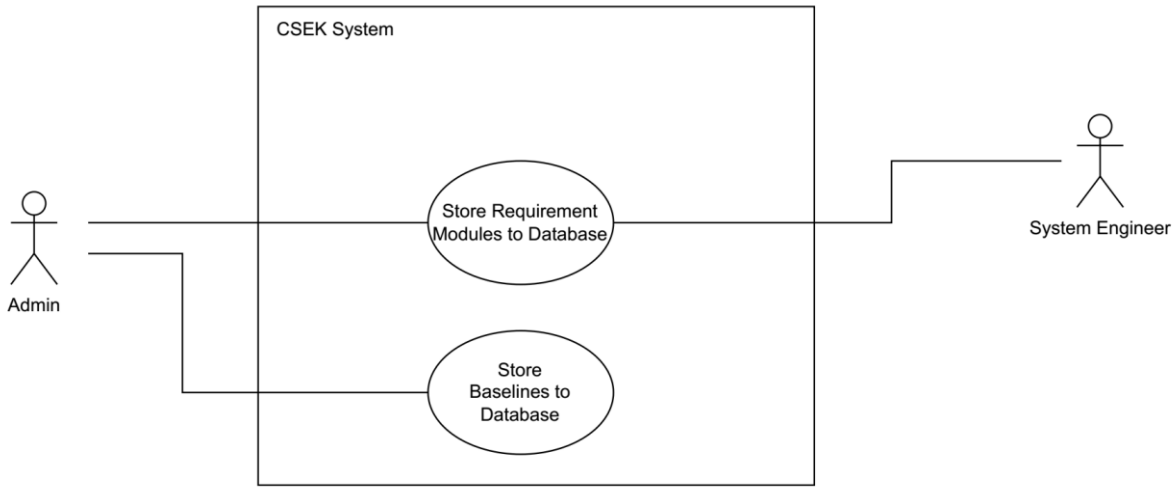

Figure 7 Use Case-7

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.

References

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

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

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

[4] Marcelo, & Marcelo. (2024, October 6). IBM Rational DOORS Yazılımına Genel Bakış | Eksiksiz Kılavuz. Visure Solutions. https://visuresolutions.com/tr/ibm-kap%C4%B1lar-k%C4%B1lavuzu/

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

[7] https://medium.com/devopsturkiye/redis-nedir-ne-i%CC%87%C5%9Fe-yarar-1a19ebbdb2b4