University Management System Architecture

About arc42

arc42, the template for documentation of software and system architecture.

Template Version 8.2 EN. (based upon AsciiDoc version), January 2023

This version of the template contains some help and explanations. It is used for familiarization with arc42 and the understanding of the concepts. For documentation of your own system you use better the plain version.

Table of Contents

Introduction and Goals

Describes the relevant requirements and the driving forces that software architects and the development team must consider. These include:

- Underlying business goals,
- Essential features,
- Essential functional requirements,
- Quality goals for the architecture, and

- Relevant stakeholders and their expectations.

Requirements Overview

Contents

The University Management System (UMS) is designed to streamline and consolidate various academic and administrative processes within a university setting. It provides a unified platform for students, faculty, and administrative staff to manage academic activities efficiently.

Motivation

From the point of view of end-users, the system aims to enhance support for academic and administrative activities, improve data accuracy, and increase overall efficiency.

Form

The main functionalities include:

Feature	Description	Actor
Student Enrollment	Students can enroll in courses and manage their schedules.	Students
Attendance Tracking	Faculty can record and monitor student attendance digitally.	Faculty Members
Resource Access	Students have access to course materials and assignments online.	Students
Grade Management	Faculty can input and manage grades, and provide feedback.	Faculty Members
Report Generation	Administrators can generate reports on academic performance.	Administrators

Quality Goals

Contents

The top quality goals for the architecture, whose fulfillment is of highest importance to the major stakeholders.

Motivation

Understanding the quality goals is crucial as they influence fundamental architectural decisions. Clear and concrete quality goals ensure the architecture meets stakeholders' expectations.

Form

Quality Goal	Scenario
Usability	New users can perform key tasks without training within 15 minutes.
Maintainability	Developers can add new features with minimal impact on existing codebase.

Performance	The system can handle 1,000 concurrent users without performance degradation.
Security	Sensitive data is protected through encryption and secure authentication mechanisms.
Scalability	The system can scale to accommodate increasing numbers of users and data volume.
Compliance	The system complies with GDPR regulations for data protection.

Stakeholders

Contents

An explicit overview of stakeholders of the system, including their roles, contact information, and expectations.

Motivation

Identifying stakeholders ensures all parties involved or affected by the system are considered, avoiding surprises during development.

Form

Role/Name	Contact	Expectations
Name 1, Dean	contact1@university.edu	Improved administrative efficiency and data-driven decision-making.
Name 2, IT Director	contact2@university.edu	A scalable, secure system built on reliable technologies like C# and .NET 9.
Name 3, Registrar	contact3@university.edu	Streamlined enrollment processes and accurate student records.
Name 4, Faculty Rep	contact4@university.edu	Simplified class management and grading tools.
Name 5, Student Rep	contact5@university.edu	User-friendly interface for course registration and accessing resources.
Data Protection Officer	dpo@university.edu	Ensure compliance with GDPR regulations.

Architecture Constraints

Contents

Any requirements that constrain software architects in their freedom of design and implementation decisions or decisions about the development process. These constraints sometimes go beyond individual systems and are valid for the whole organization.

Motivation

Architects need to know exactly where they are free in their design decisions and where they must adhere to constraints. Constraints must always be dealt with; they may be negotiable, though.

Form

Technical Constraints

Constraint	Explanation
Technology Stack	Use C# with .NET 9 for modern, cross-platform support and access to the latest features.
UI Framework	Utilize Blazor WebAssembly for building rich, interactive web applications using `.razor` pages.
Database System	Use Microsoft SQL Server for relational data storage.
Authentication Mechanism	Integrate with existing LDAP for user authentication to leverage current infrastructure.

Organizational Constraints

Constraint	Explanation
Development Process	Follow Agile methodologies with iterative development and continuous feedback.
Coding Standards	Adhere to internal coding standards and best practices for C# and .NET development.
Deployment Environment	Host applications on university-managed servers within the EU to comply with data regulations.

Regulatory Constraints

Constraint	Explanation
GDPR Compliance	Ensure all data handling complies with GDPR requirements for data protection and privacy.

System Scope and Context

Business Context

Contents

Specification of all communication partners (users, IT-systems, etc.) with explanations of domain-specific inputs and outputs or interfaces.

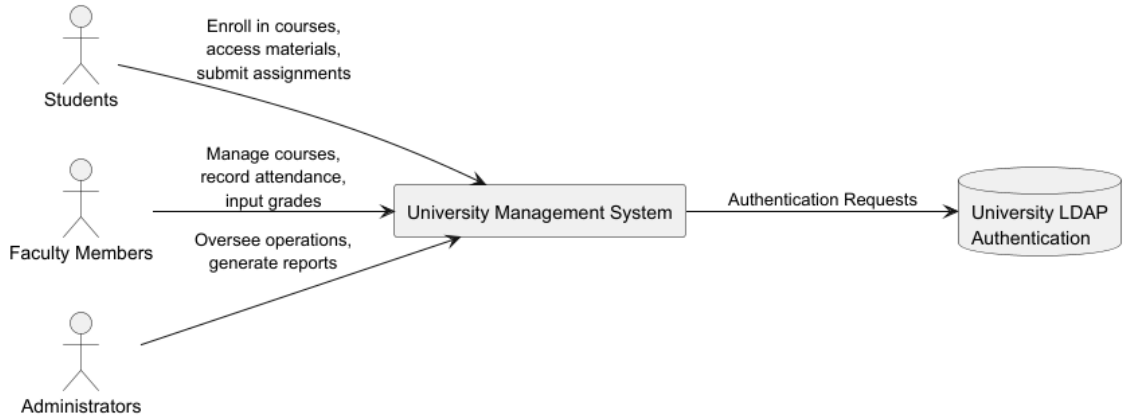
Motivation

All stakeholders should understand which data are exchanged with the environment of the system.

Form

Context Diagram

The context diagram shows the interaction between the actors and the University Management System.



Explanation of External Domain Interfaces

- **Students**: Enroll in courses, access materials, submit assignments.
- **Faculty Members**: Manage courses, record attendance, input grades.
- **Administrators**: Oversee system operations, generate reports.
- **University LDAP Authentication**: Provides user authentication services.

Technical Context

Contents

Technical interfaces (channels and transmission media) linking the system to its environment, and mapping of domain-specific input/output to these channels.

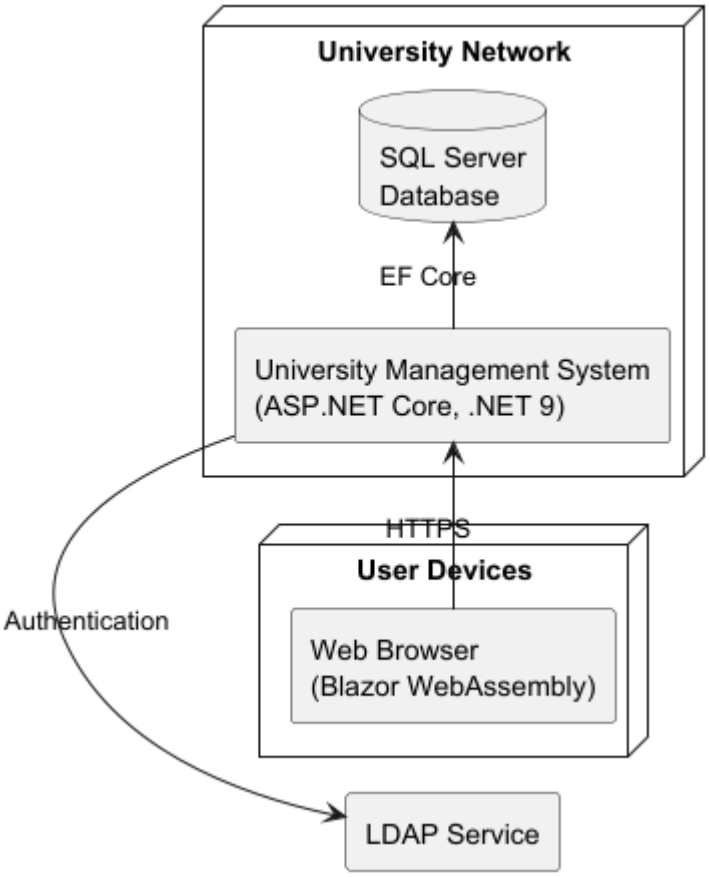
Motivation

Many stakeholders make architectural decisions based on the technical interfaces between the system and its context.

Form

Technical Context Diagram

The technical context diagram shows the technical components involved, such as the user devices, university network, and communication interfaces.



Explanation of Technical Interfaces

- **Web Clients (Blazor WebAssembly)**: Browser-based access using `.razor` pages.
- **Database Connection**: Secure access to SQL Server database via Entity Framework Core.
- **LDAP Authentication**: Integration with the university's LDAP service for authentication.
- **HTTPS Protocol**: All communications occur over secure HTTPS connections.

Solution Strategy

Contents

A short summary and explanation of the fundamental decisions and solution strategies that shape the system architecture.

- **Technology Decisions**:
  - Use C# with .NET 9 for server and client development.
  - Utilize Blazor WebAssembly for a unified C# development experience.
- **Top-Level Decomposition**:
  - Implement a **Layered Architecture** separating concerns into Presentation, Business Logic, and Data Access layers.
- **Achieving Key Quality Goals**:

    - **Usability**: Design intuitive UI with Blazor components and responsive design principles.
    - **Maintainability**: Apply SOLID principles and modular design for easy feature additions and code management.
    - **Performance**: Leverage client-side processing with Blazor WebAssembly to reduce server load.
    - **Security**: Implement robust authentication, authorization, and data protection mechanisms.
    - **Compliance**: Integrate GDPR-compliant data handling practices throughout the system.
- **Organizational Decisions**:
    - Adopt Agile development methodologies with continuous integration and deployment pipelines.

Motivation

These decisions form the cornerstones for the architecture. They are the foundation for many other detailed decisions or implementation rules.

Form

Keep the explanations of such key decisions short. Motivate what was decided and why it was decided that way, based upon the problem statement, quality goals, and key constraints. Refer to details in the following sections.

Building Block View

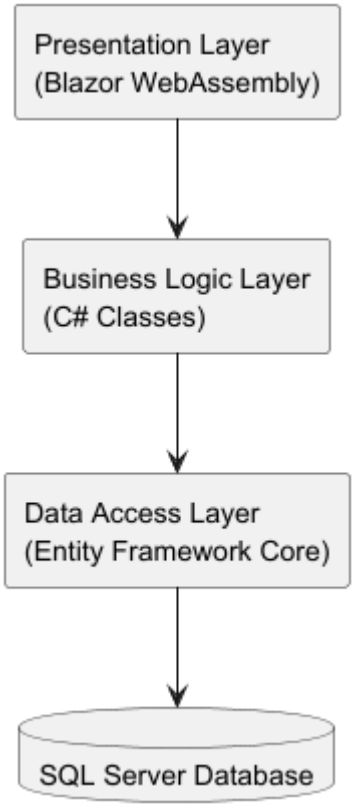
Content

The building block view shows the static decomposition of the system into building blocks (modules, components, classes, etc.) as well as their dependencies.

Infrastructure Level 1

Overview Diagram

This diagram shows the building blocks of the University Management System, including the Presentation, Business Logic, and Data Access layers.



Motivation

Maintain an overview of the source code by making its structure understandable through abstraction. This allows communication with stakeholders on an abstract level without disclosing implementation details.

Form

The building block view is a hierarchical collection of black boxes and white boxes and their descriptions.

Whitebox Overall System

Motivation

A layered architecture promotes separation of concerns, enhancing maintainability, scalability, and testability.

Contained Building Blocks

Name	Responsibility
Presentation Layer	Manages user interactions via Blazor components running in the browser.
Business Logic Layer	Encapsulates core application logic and business rules.

Data Access Layer	Handles data persistence and retrieval using Entity Framework Core.
Database	Stores persistent data in Microsoft SQL Server.

Important Interfaces

- **API Endpoints**: RESTful APIs facilitating communication between client and server.
- **Data Models**: Shared data contracts between layers ensuring consistency.
- **Authentication Interface**: Integration with LDAP for user authentication.

Runtime View

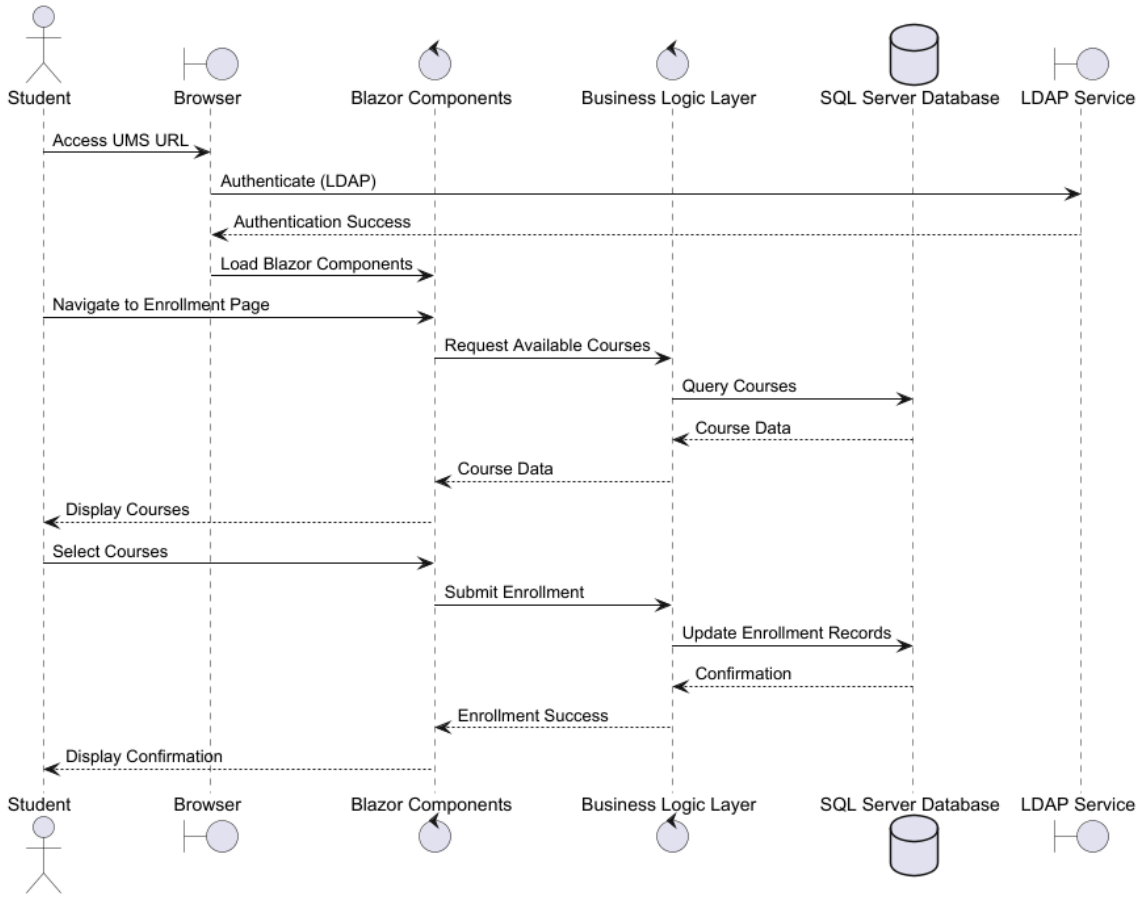
Contents

The runtime view describes concrete behavior and interactions of the system's building blocks in the form of scenarios.

Student Enrollment Scenario

Student Enrollment Scenario

This sequence diagram illustrates the flow for student enrollment, showing how the different components interact during the process.



Notable Aspects

- **Client-Side Rendering**: Enhances user experience with responsive UI.
- **Secure Communication**: All data exchanges occur over HTTPS with proper authentication tokens.
- **Asynchronous Operations**: Improves performance and scalability.

Deployment View

Content

The deployment view describes:

1. Technical infrastructure used to execute the system, including infrastructure elements like environments, computers, processors, channels, and network topologies.
2. Mapping of software building blocks to those infrastructure elements.

Motivation

- **Scalability**: Web servers can be scaled horizontally to handle increased load.
- **Security**: Secure communications and data storage comply with GDPR.

Quality and/or Performance Features

- **Client-Side Processing**: Reduces server workload, enhancing scalability.
- **Efficient Data Access**: Optimized queries and caching strategies improve performance.

Mapping of Building Blocks to Infrastructure

- **Presentation Layer**: Deployed to the web server and executed in user browsers as Blazor WebAssembly apps.
- **Business Logic Layer**: Runs on the web server within the ASP.NET Core application.
- **Data Access Layer**: Operates on the web server, accessing the database server.
- **Database**: Hosted on a dedicated SQL Server within the university's data center.

Cross-cutting Concepts

Content

This section describes overall, principal regulations and solution ideas that are relevant in multiple parts of the system.

Motivation

Concepts form the basis for conceptual integrity (consistency, homogeneity) of the architecture. Thus, they are an important contribution to achieving the inner qualities of the system.

Form

- Concept papers with any kind of structure.
- Cross-cutting model excerpts or scenarios using notations of the architecture views.
- Sample implementations, especially for technical concepts.
- Reference to typical usage of standard frameworks.

Security Concepts

Explanation

- **Authentication and Authorization**:
  - Use ASP.NET Core Identity with LDAP integration.
  - Implement role-based access control (RBAC).
- **Data Protection**:
  - Encrypt sensitive data at rest and in transit.
  - Use HTTPS for all communications.
- **Input Validation**:
  - Sanitize user inputs to prevent SQL injection, XSS, and other attacks.

UI/UX Concepts

Explanation

- **Blazor Components**:
  - Create reusable `.razor` components for consistency.
  - Follow responsive design principles for compatibility with various devices.
- **Accessibility**:
  - Ensure compliance with WCAG 2.1 standards.

Exception Handling and Logging

Explanation

- **Global Exception Handling**:
  - Implement middleware to catch and handle exceptions gracefully.
- **Logging Framework**:
  - Use Serilog for structured logging.
  - Store logs securely and comply with GDPR regarding personal data in logs.

Configuration Management

Explanation

- **AppSettings**:
  - Use `appsettings.json` files with environment-specific overrides.
- **Secrets Management**:
  - Securely store sensitive configurations using Azure Key Vault or similar services.
- **CI/CD**:
  - Automate builds, tests, and deployments using tools like GitHub Actions or Azure DevOps.

Development Concepts

Explanation

- **Coding Standards**:

    - Adhere to Microsoft's C# coding conventions.
- **Testing Strategy**:
    - Unit tests with xUnit.
    - Integration tests for API endpoints.
    - Use bUnit for Blazor component testing.
- **Version Control**:
    - Use Git for source code management with branching strategies.

Architecture Decisions

Contents

Important, expensive, large-scale, or risky architecture decisions including rationales.

Motivation

Stakeholders of the system should be able to comprehend and retrace decisions.

Form

- ADR (Architecture Decision Records) for every important decision.

Decision 1: Use of Blazor WebAssembly with .NET 9

- **Decision**: Utilize Blazor WebAssembly for client-side web application development using .NET 9.
- **Rationale**:
    - Allows full-stack development with C#, reducing context switching.
    - Enhances performance with client-side rendering.
    - Access to the latest .NET 9 features and improvements.
- **Consequences**:
    - Requires developers to be proficient in Blazor and WebAssembly.
    - Larger initial load times due to WebAssembly payload.
    - Need to handle browser compatibility and limitations.

Decision 2: Adoption of Entity Framework Core

- **Decision**: Use Entity Framework Core for Object-Relational Mapping (ORM).
- **Rationale**:
    - Simplifies data access and reduces boilerplate code.
    - Supports LINQ for querying, enhancing developer productivity.
- **Consequences**:
    - Potential performance overhead if not optimized.
    - Requires understanding of EF Core's behaviors and configurations.

Decision 3: Integration with LDAP for Authentication

- **Decision**: Integrate with the university's existing LDAP service for authentication.
- **Rationale**:
    - Leverages existing infrastructure and user credentials.

    - Provides a unified authentication mechanism across university services.
- **Consequences**:
    - Complexity in integrating LDAP with ASP.NET Core Identity.
    - Potential issues with synchronization and user data consistency.

Quality Requirements

Content

This section contains all quality requirements as a quality tree with scenarios.

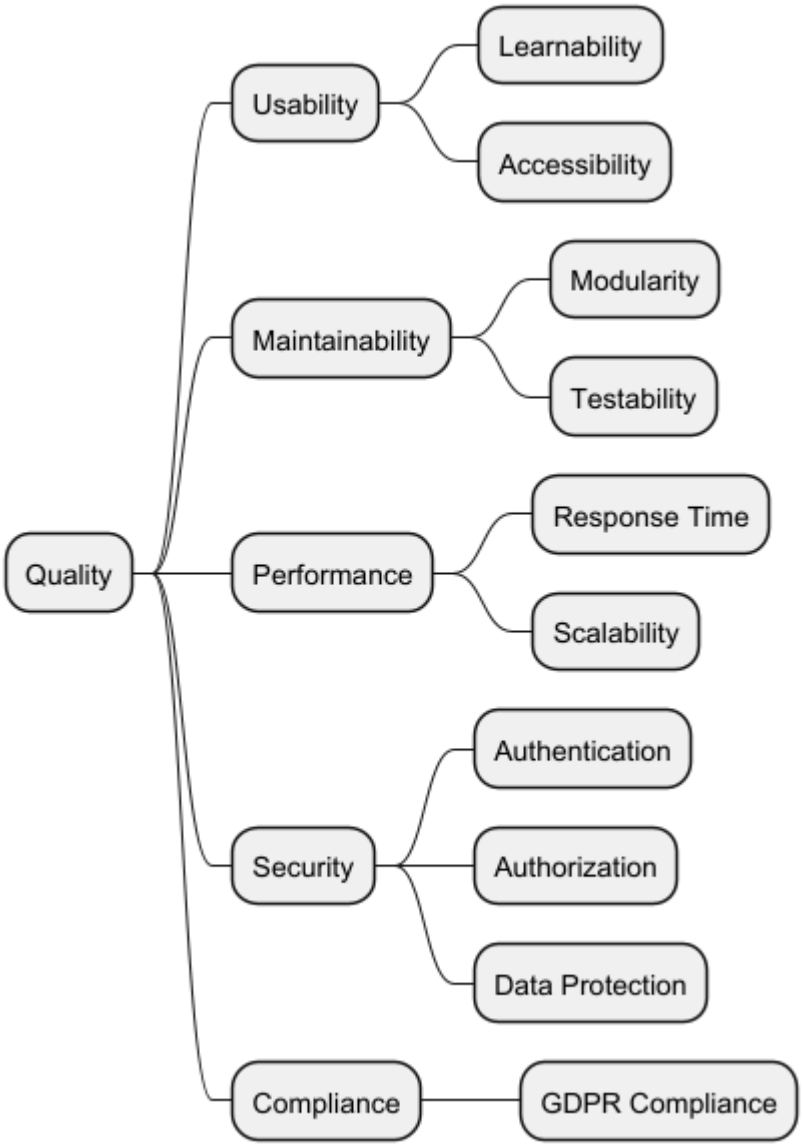
Motivation

Since quality requirements will have a significant influence on architectural decisions, it's essential to know what is important to stakeholders, expressed concretely and measurably.

Quality Tree

Quality Tree Diagram

The quality tree diagram presents the quality attributes of the system, such as usability, maintainability, and performance.



Motivation

The tree structure provides an overview for a large number of quality requirements.

Quality Scenarios

Contents

Concretization of quality requirements using scenarios.

Usability Scenario

- **Scenario ID**: US-01
- **Description**: A new student can navigate the system and enroll in courses within 15 minutes without prior training.

- **Stimulus**: A first-time user accesses the system.
- **Response**: The user successfully completes course enrollment within the time frame.
- **Measurement**: Time taken by usability test participants.

Performance Scenario

- **Scenario ID**: PE-01
- **Description**: The system responds to user actions within 2 seconds under normal load.
- **Stimulus**: User performs standard operations during peak hours.
- **Response**: System processes and responds within the specified time.
- **Measurement**: Average response time from server logs.

Security Scenario

- **Scenario ID**: SE-01
- **Description**: Unauthorized access attempts are detected and logged; accounts are locked after 5 failed attempts.
- **Stimulus**: Multiple failed login attempts by an unauthorized user.
- **Response**: Account is locked, and the event is logged for review.
- **Measurement**: Security audit logs.

Compliance Scenario

- **Scenario ID**: CO-01
- **Description**: User data is handled in compliance with GDPR requirements.
- **Stimulus**: User requests data deletion.
- **Response**: User data is anonymized or deleted within 30 days.
- **Measurement**: Time taken to fulfill data deletion requests.

Motivation

Scenarios make quality requirements concrete and allow for easier measurement and evaluation.

Risks and Technical Debts

Contents

A list of identified technical risks or technical debts, ordered by priority.

Motivation

Risk management is essential for project success. Identifying and addressing risks and technical debts early can prevent future issues.

Form

List of risks and/or technical debts, including suggested measures to minimize, mitigate, or avoid risks or reduce technical debts.

1. **Learning Curve with Blazor WebAssembly and .NET 9**
   - **Risk**: Developers may need time to adapt to new technologies.
   - **Mitigation**: Provide training sessions and access to learning resources.
2. **Browser Compatibility Issues**
   - **Risk**: Inconsistent behavior across different browsers due to WebAssembly support.

- **Mitigation**: Conduct thorough cross-browser testing and implement polyfills if necessary.

3. **LDAP Integration Challenges**
   - **Risk**: Potential difficulties integrating LDAP with ASP.NET Core Identity.
   - **Mitigation**: Allocate additional time for integration testing and involve LDAP experts.
4. **Performance Overhead**
   - **Risk**: Large initial download size of Blazor WebAssembly apps affecting load times.
   - **Mitigation**: Optimize assets, enable compression, and implement lazy loading.
5. **GDPR Compliance Complexity**
   - **Risk**: Ensuring all data processing complies with GDPR may be challenging.
   - **Mitigation**: Consult with the Data Protection Officer and perform regular compliance audits.

Glossary

Contents

The most important domain and technical terms that stakeholders use when discussing the system.

Motivation

Clear definitions ensure all stakeholders have an identical understanding of terms, avoiding confusion due to synonyms or homonyms.

Form

A table with columns for terms and their definitions.

Term	Definition
Blazor WebAssembly	A framework for building interactive client-side web applications using C# and WebAssembly.
`.razor` Pages	Files containing Blazor components combining HTML markup with C# code.
Entity Framework Core	An open-source ORM framework for .NET applications.
LDAP	Lightweight Directory Access Protocol used for accessing and maintaining distributed directory information services.
GDPR	General Data Protection Regulation; EU law on data protection and privacy.
ORM (Object-Relational Mapping)	A programming technique for converting data between incompatible systems using object-oriented programming languages.
SOLID Principles	A set of design principles for object-oriented programming aimed at making software designs more understandable, flexible, and maintainable.

CI/CD	Continuous Integration and Continuous Deployment; practices that automate software building, testing, and deployment.