table of contents

Template :revnumber: 8.2 EN :revdate: January 2023 :revremark: (based upon AsciiDoc version) :toc-title: Table of Contents

About arc42

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

Template Version {revnumber}. {revremark}, {revdate}

1. Introduction and Goals

Housing system is an administrative system for a landlord that rents houses. It’s main goals are to facilitate the landlord to administrate the followings: * rental objects * tennants * service charge billings The project should also give the tennants a portal to check their daily consumptions of heating energy and water. …

This is a project of Alexander Rothschild, Hantao Zhou, Marco Campione.

1.1. Requirements Overview

Contents

The main purpose of Housing system is keeping track of all the rental objects, tennants informations, billings, consumptions and sensor management so that the landlord has all the informations updated. Another purpose is to create a portal for the tennants to pat for the rent and the services

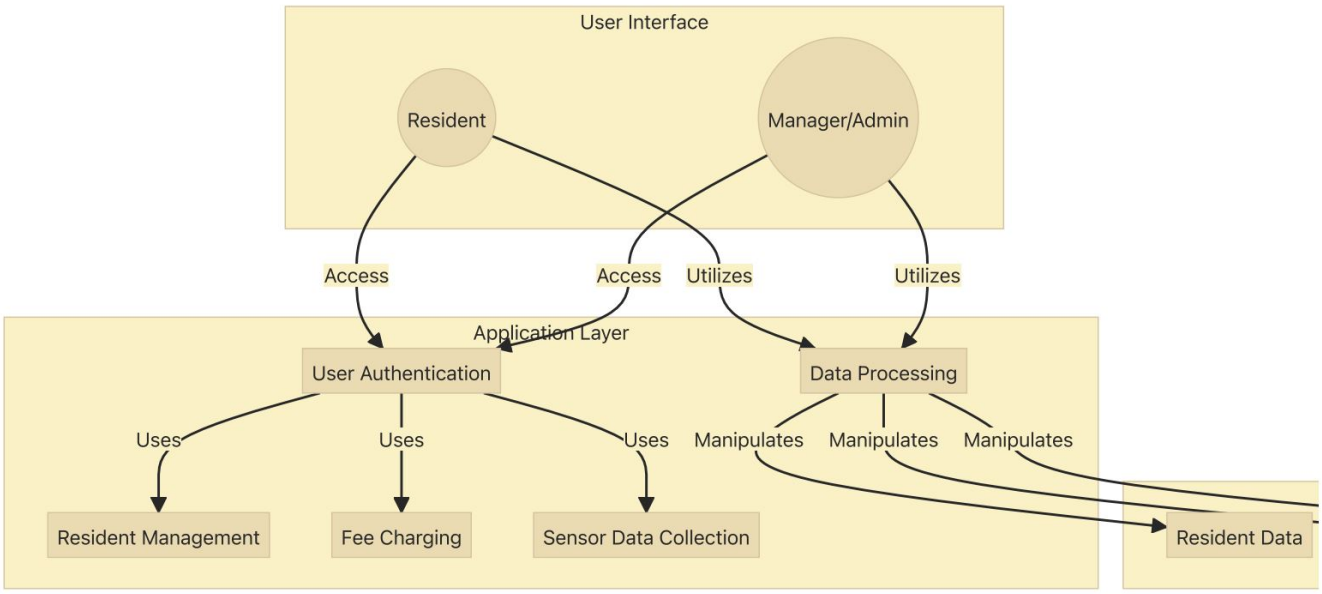
Motivation

The landlord asked for a system that helps him to keep track of everything he think it’s important for the administration of his rental objects. He asked for this project because right now he is working with Excel sheets and he is keeping every information updated "manually". This project aims to shift this manual work to an automated system.

1.2. Basic ideas and Brainstorming

A software design architecture to process all the info related to heat fee, including functions of collecting info from sensors, managing when the resident moves out, charging fees etc. Interact mainly with two types of user, resident as normal users and managers and system maintainers as admins.

We used mermaid to create a diagram Create a coarse boxes and arrows diagram (only most basic components) to give yourself a better first overview of the system’s requirements.



Use Case ID and Name	UC001 - Record and Charge Heating Fees
Actors	Tenant, Property Manager
Description	This use case involves recording and charging heating fees for rented houses. The system tracks the usage of heating resources in a house and calculates the corresponding fees for the tenant.
Preconditions	1. Tenant has an active rental agreement. 2. The heating system is installed and operational.
Postconditions	1. Heating fees are recorded for the specified period. 2. Tenant receives a bill for the heating charges.
Invariants	None
Basic Workflow	1. Tenant adjusts the heating preferences. 2. System records heating resource consumption. 3. System calculates heating fees based on usage. 4. Tenant receives a bill for the heating charges.
Alternative Workflow	- If the heating system malfunctions, notify property manager and suspend fee calculation until the issue is resolved. - If tenant disputes the heating charges, initiate a review process and adjust fees if necessary.
Risks	1. Malfunction of heating system. 2. Disputes between tenant and property manager regarding fee calculation.
Quality Goals	1. Accuracy in calculating heating fees. 2. Timely generation and delivery of heating fee bills.
Special Requirements	1. The system should support multiple methods of heating (e.g., central heating, electric heating) for accurate fee calculation. 2. Heating fee bills should be clear and itemized for easy understanding by the tenant.

Use Case ID and Name	UC001 - Record and Charge Heating Fees
Assumptions	1. Tenants are responsible for their heating charges as per the rental agreement. 2. The heating system is equipped with meters or sensors to measure consumption accurately.

Use Case ID and Name	UC002 - Register New Tenant
Actors	Landlord, New Tenant
Description	This use case involves the landlord registering a new tenant in the system when a property is rented out to a new occupant.
Preconditions	1. The landlord is logged into the property management system. 2. The property is available for rent.
Postconditions	1. The new tenant’s information is added to the system. 2. The rental agreement is generated and stored.
Invariants	None
Basic Workflow	1. Landlord logs into the property management system. 2. Landlord accesses the ``New Tenant Registration'' feature. 3. Landlord enters the new tenant’s information (name, contact details, etc.). 4. Landlord specifies the terms of the rental agreement (duration, rent amount, etc.). 5. System validates the information provided. 6. System generates a unique identifier for the new tenant and stores their details. 7. System generates a rental agreement document. 8. Landlord reviews and confirms the registration.
Alternative Workflow	- If the information provided is incomplete or invalid, the system prompts the landlord to correct the details. - If the rental agreement terms need negotiation, the landlord can enter a negotiation phase with the new tenant.
Risks	1. Incomplete or inaccurate tenant information. 2. Miscommunication regarding rental agreement terms.
Quality Goals	1. Accurate and complete recording of tenant information. 2. Efficient generation and storage of rental agreements.

Use Case ID and Name	UC002 - Register New Tenant
Special Requirements	1. The system should support the attachment of legal documents related to the rental agreement. 2. Notifications should be sent to both the landlord and the new tenant upon successful registration.
Assumptions	1. The landlord has the legal authority to rent out the property. 2. The property is in a suitable condition for occupancy.

Use Case ID and Name	UC003 - Handle Tenant Rent Delay
Actors	Property Manager, Tenant
Description	This use case involves the property manager addressing a situation where a tenant is delaying on paying rent. The goal is to initiate communication and resolve the issue in a timely and fair manner.
Preconditions	1. Tenant has not paid the rent on the due date. 2. Property manager is logged into the property management system.
Postconditions	1. Communication with the tenant regarding the rent delay is documented. 2. A plan for resolution is agreed upon, which may include late fees or a revised payment schedule.
Invariants	None
Basic Workflow	1. Property manager receives a notification or identifies that a tenant's rent is overdue. 2. Property manager accesses tenant information in the property management system. 3. Property manager initiates communication with the tenant through the system, inquiring about the reason for the delay. 4. Tenant responds with the reason for the delay. 5. Property manager reviews the situation and determines appropriate actions, which may include imposing late fees or negotiating a new payment schedule. 6. Property manager updates the system with the details of the communication and any agreed-upon resolution. 7. If the issue persists, the property manager may escalate the matter following the established protocol.

Use Case ID and Name	UC003 - Handle Tenant Rent Delay
Alternative Workflow	- If the tenant provides a valid reason for the delay (e.g., unexpected financial hardship), the property manager may work with the tenant to establish a temporary solution. - If the tenant is unresponsive, the property manager may escalate the issue by sending formal notices or involving legal channels as per the rental agreement.
Risks	1. Miscommunication between the property manager and tenant. 2. Tenant disputes regarding late fees or resolution terms.
Quality Goals	1. Timely and clear communication regarding rent delays. 2. Fair and consistent application of policies for resolving rent delays.
Special Requirements	1. The system should support the documentation of all communication related to rent delays. 2. Notifications to both parties should be clear and provide relevant information.
Assumptions	1. The rental agreement includes terms and policies regarding rent payments and late fees. 2. Both parties are expected to communicate through the system for transparency.

Use Case ID and Name	UC004 - Update System Security
Actors	System Administrator
Description	This use case involves the system administrator updating the security measures of the system to mitigate potential risks and ensure the protection of sensitive information and resources.
Preconditions	1. The system administrator has proper access rights. 2. Identified security vulnerabilities or a routine security update schedule.
Postconditions	1. The system’s security measures are updated. 2. Documentation of the security update is recorded.
Invariants	None

Use Case ID and Name	UC004 - Update System Security
Basic Workflow	1. System administrator identifies the need for a security update, either through routine checks or the discovery of vulnerabilities. 2. System administrator logs into the system with appropriate credentials. 3. System administrator accesses the security settings and configurations. 4. System administrator applies the necessary updates, patches, or configuration changes to address identified vulnerabilities or enhance security. 5. System administrator tests the updated security measures to ensure they do not disrupt system functionality. 6. System administrator documents the details of the security update, including the changes made and any testing outcomes. 7. If the update is successful, the system administrator notifies relevant stakeholders about the security enhancement.
Alternative Workflow	- If the security update requires system downtime, the system administrator coordinates with relevant parties to minimize disruption. - If the update reveals unforeseen issues or conflicts, the system administrator may need to roll back the changes and investigate the cause before reapplying the update.
Risks	1. Potential system downtime during the update. 2. Unforeseen issues or conflicts arising from the security update.
Quality Goals	1. Minimize system downtime during security updates. 2. Ensure that security updates do not introduce new vulnerabilities.
Special Requirements	1. The system should support rollback mechanisms in case of issues with the security update. 2. Detailed documentation of security updates should be maintained for audit and compliance purposes.
Assumptions	1. The system administrator has a thorough understanding of the system's architecture and security requirements. 2. Relevant stakeholders are informed about the scheduled security update.

Use Case ID and Name	UC005 - Handle Broken Heat Sensor
Actors	Tenant, Maintenance Personnel
Description	This use case involves the process of identifying and resolving a broken heat sensor in a rented home to ensure the proper functioning of the heating system.

Use Case ID and Name	UC005 - Handle Broken Heat Sensor
Preconditions	1. Tenant notices an issue with the heating system or reports a lack of accurate temperature readings. 2. Maintenance personnel have access to the home and the necessary tools for sensor replacement.
Postconditions	1. The broken heat sensor is replaced. 2. The heating system is functioning correctly.
Invariants	None
Basic Workflow	1. Tenant notices a discrepancy in the temperature readings or experiences issues with the heating system. 2. Tenant reports the issue to the property management system. 3. Property management system logs a maintenance request and notifies maintenance personnel. 4. Maintenance personnel schedule a visit to the home. 5. Maintenance personnel assess the heat sensor and confirm it is malfunctioning. 6. Maintenance personnel replace the broken heat sensor with a new one. 7. Maintenance personnel test the heating system to ensure it is functioning correctly. 8. Maintenance personnel update the property management system with details of the resolution. 9. Tenant is notified that the issue has been resolved.
Alternative Workflow	- If the broken heat sensor is under warranty, maintenance personnel may contact the sensor manufacturer for a replacement. - If the replacement of the heat sensor requires a specialized technician, maintenance personnel may coordinate with external service providers.
Risks	1. Delay in addressing the issue may lead to discomfort for the tenant. 2. Unavailability of the required replacement parts or sensors.
Quality Goals	1. Timely resolution of heating system issues. 2. Accurate and reliable temperature readings after sensor replacement.
Special Requirements	1. The property management system should efficiently log and track maintenance requests. 2. Maintenance personnel should have access to replacement parts and sensors as needed.
Assumptions	1. Tenants are prompt in reporting issues with the heating system. 2. Maintenance personnel are adequately trained to handle sensor replacements.

Use Case ID and Name	UC005 - Handle Broken Heat Sensor
Actors	Tenant, Maintenance Personnel
Description	This use case involves the process of identifying and resolving a broken heat sensor in a rented home to ensure the proper functioning of the heating system.
Preconditions	1. Tenant notices an issue with the heating system or reports a lack of accurate temperature readings. 2. Maintenance personnel have access to the home and the necessary tools for sensor replacement.
Postconditions	1. The broken heat sensor is replaced. 2. The heating system is functioning correctly.
Invariants	The temperature readings from the heat sensor must be accurate and within an acceptable range.
Basic Workflow	1. Tenant notices a discrepancy in the temperature readings or experiences issues with the heating system. 2. Tenant reports the issue to the property management system. 3. Property management system logs a maintenance request and notifies maintenance personnel. 4. Maintenance personnel schedule a visit to the home. 5. Maintenance personnel assess the heat sensor and confirm it is malfunctioning. 6. Maintenance personnel replace the broken heat sensor with a new one. 7. Maintenance personnel test the heating system to ensure it is functioning correctly. 8. Maintenance personnel update the property management system with details of the resolution. 9. Tenant is notified that the issue has been resolved.
Alternative Workflow	- If the broken heat sensor is under warranty, maintenance personnel may contact the sensor manufacturer for a replacement. - If the replacement of the heat sensor requires a specialized technician, maintenance personnel may coordinate with external service providers.
Risks	1. Delay in addressing the issue may lead to discomfort for the tenant. 2. Unavailability of the required replacement parts or sensors.
Quality Goals	1. Timely resolution of heating system issues. 2. Accurate and reliable temperature readings after sensor replacement.
Special Requirements	1. The property management system should efficiently log and track maintenance requests. 2. Maintenance personnel should have access to replacement parts and sensors as needed.

Use Case ID and Name	UC005 - Handle Broken Heat Sensor
Assumptions	1. Tenants are prompt in reporting issues with the heating system. 2. Maintenance personnel are adequately trained to handle sensor replacements.

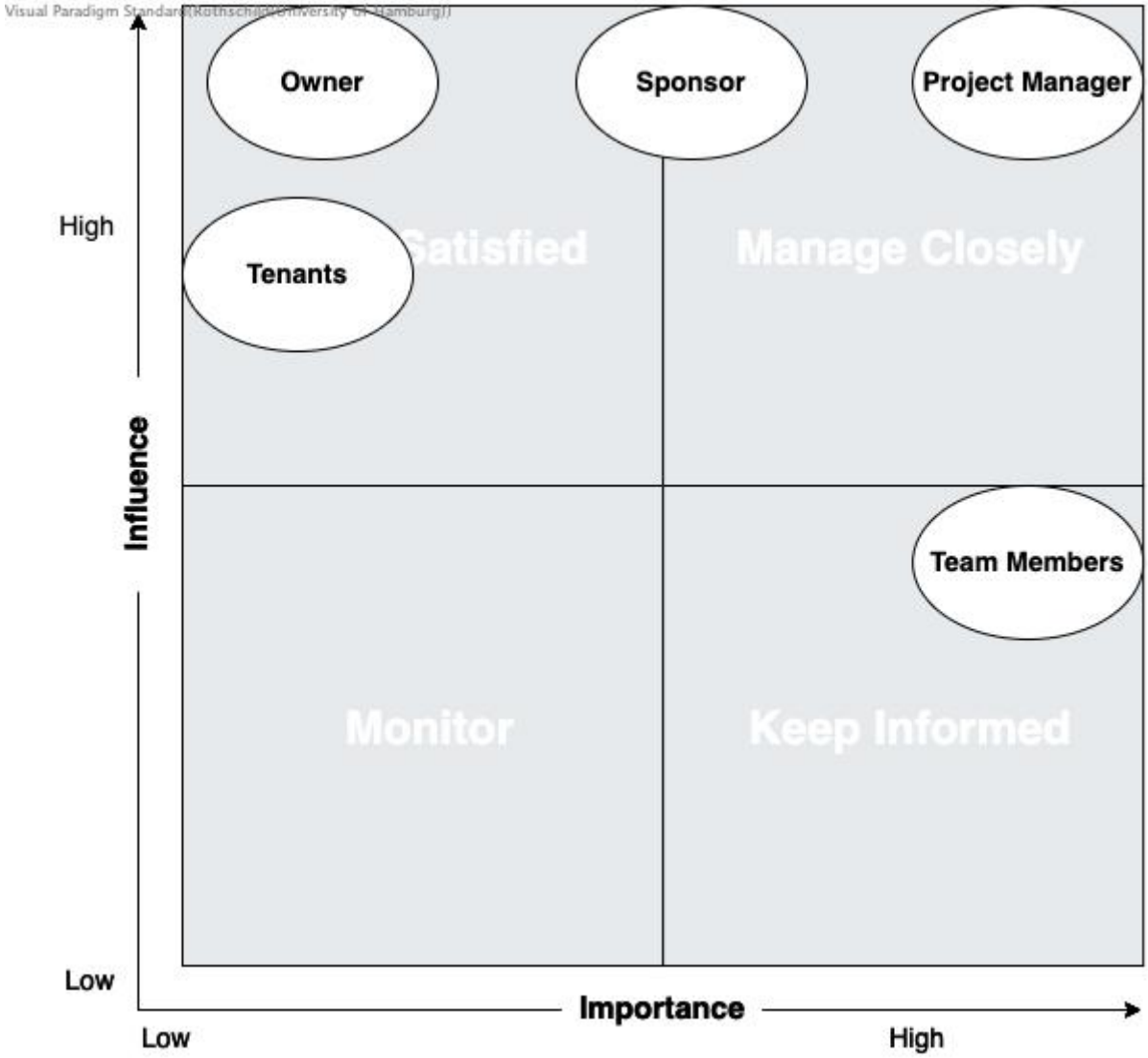
1.3. Quality Goals

- TODO

1.4. Stakeholders

Role/Name	Contact	Expectations
Customer	*<Landlord>*	_1. What is your business? What are you doing? Answer: landlord/house owner 2. What challenges you want to solve throughout the project? Answer: Automation of the processes, integrate billing system, management of the sensor system, etc … 3. Are there any existing systems that need to be integrated? Answer: excel sheets (not sure) 4. What is your product and for whom is it? Who is your target audience? Answer: the product is mainly for him, there is no need to generalize to a wider target audience 5. Is it important for you what technologies are going to be used by us? Answer: Didn’t seem that the customer cared about the technologies 6. What are the project risks for your company? Answer: He hasn’t said anything about it 7. What are the deadlines? Till what time what should be ready? Answer: He hasn’t said anything about it 8. If it is a REST app do your have the REST endpoints or should we develop them on your side? Answer: He hasn’t said anything about it 9. What is the projects budget? Answer: He hasn’t said anything about it 10. How many people are there in the project and how many you need? Answer: He hasn’t said anything about it 11. Are there any software architecture constraints that we should know about (architectural pattern or design pattern)? Answer: He hasn’t said anything about it 12. Ask about that: the building block view shows the static decomposition of the system into building blocks (modules, components, subsystems, classes, interfaces, packages, libraries, frameworks, layers, partitions, tiers, functions, macros, operations, data structures, …) as well as their dependencies (relationships, associations, …) Answer: He hasn’t said anything about it 13. Make sure that we know: * important use cases or features: how do building blocks execute them? * interactions at critical external

1.4.1. Stakeholder Influence and Importance Matrix



1.4.2. Stakeholder Interest and Impact Table

Stakeholder	Interests	Estimated Project Impact	Estimated Priority
Owner	*Achieve targets* *Avoid liability*	*Med +* *High-*	*1*
Sponsor	*Save money through an all-in-one service instead of filling excel sheets and processing payments separately*	*Med +*	*3*
Project Manager (Lecturer)	*Keeping owner/sponsor and team members satisfied with the project in general through providing a connection between them*	*High +*	*2*
Team Members	*New product excitement* *Pass an exam in the semester end*	*Med +* *Med +*	*4*
Tenants	*Want to have a comfortable online service*	*Low-*	*5*

1.4.3. Interest-Influence Classification

Stakeholder	Estimated Project Influence	Estimated Project Importance	Assumptions and Risks
Owner	*High (10)*	*Low (2)*	_Providing all the resources, but his requirements a bit vague _
Sponsor	*High (10)*	*Medium (6)*	*Assuming we have one for this particular project, we are not sure if additional funding will be provided, when and if needed*
Project Manager (Lecturer)	*High (10)*	*High (10)*	*Likes the new project. Risks tied to explaining the requirements to team members*
Team Members	*Medium (6)*	*High (10)*	*Almost all the members are glad to work on a new project. Though one member dropped it almost at the start. Additional training required*
Tenants	*High (8)*	*Low (1)*	*Gladly use the comfortable new service. Financial risks if payment or other service fails*

2. Architecture Constraints

Technical constraints

- TODO

Nr.	Constraint	Background and/or motivation
TC1	*insert_constraint_here*	*insert_motivation_here*
TC2	*insert_constraint_here*	*insert_motivation_here*
TC3	*insert_constraint_here*	*insert_motivation_here*
TC4	*insert_constraint_here*	*insert_motivation_here*
TC5	*insert_constraint_here*	*insert_motivation_here*

Organizational constraints

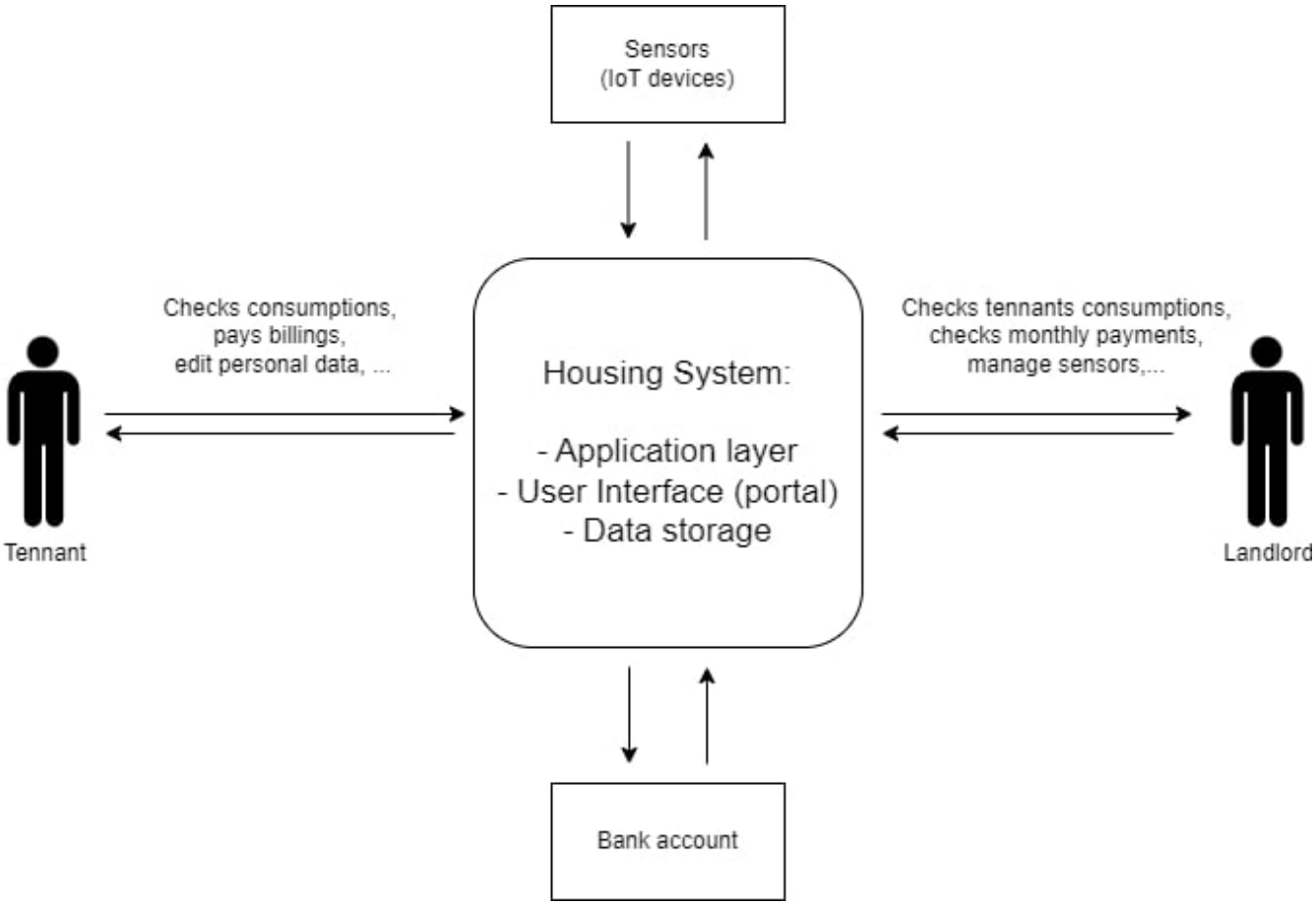
- TOFINISH

Nr.	Constraint	Background and/or motivation
OC1	User data location	All the personal data about users have to be stored in Germany or at least in a EU country, for privacy reasons.
OC2	Use of AWS	All the datas have to stored in cloud like AWS.
OC3	*insert_constraint_here*	*insert_motivation_here*
OC4	*insert_constraint_here*	*insert_motivation_here*
OC5	*insert_constraint_here*	*insert_motivation_here*

3. Context and Scope

3.1. Business Context

This section describes the environment and context of the Housing system at a high level: Who uses the system and on which other system does the it depend.



<Diagram or Table>

<optionally: Explanation of external domain interfaces>

3.2. Technical Context

<Diagram or Table>

<optionally: Explanation of technical interfaces>

<Mapping Input/Output to Channels>

4. Solution Strategy

5. Building Block View

5.1. Whitebox Overall System

*<**Overview Diagram**>*

Motivation

<text explanation>

Contained Building Blocks

<Description of contained building block (black boxes)>

Important Interfaces

<Description of important interfaces>

5.1.1. <Name black box 1>

<Purpose/Responsibility>

<Interface(s)>

<(Optional) Quality/Performance Characteristics>

<(Optional) Directory/File Location>

<(Optional) Fulfilled Requirements>

<(optional) Open Issues/Problems/Risks>

5.1.2. <Name black box 2>

<black box template>

5.1.3. <Name black box n>

<black box template>

5.1.4. <Name interface 1>

…

5.1.5. <Name interface m>

5.2. Level 2

5.2.1. White Box <*building block 1*>

<white box template>

5.2.2. White Box *<building block 2>*

<white box template>

...

5.2.3. White Box *<building block m>*

<white box template>

5.3. Level 3

5.3.1. White Box <_building block x.1_>

<white box template>

5.3.2. White Box <_building block x.2_>

<white box template>

5.3.3. White Box <_building block y.1_>

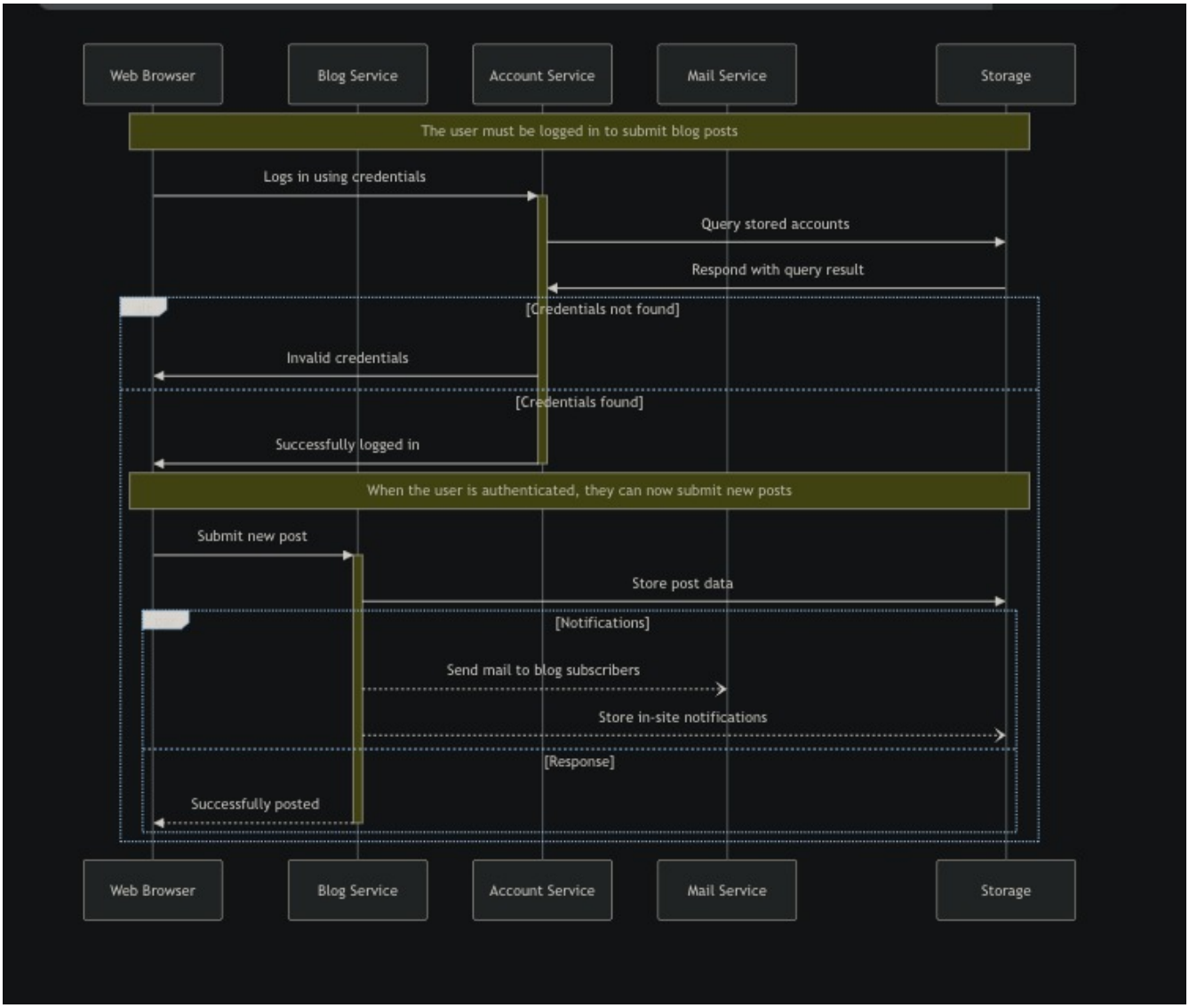
<white box template>

6. Runtime View

User interaction with *project_name* and error handling is pretty basic and simple.

6.1. <Runtime Scenario 1 (Login procedure)>

The following is the use case of the login procedure.



- *<insert runtime diagram or textual description of the scenario>*
- *<insert description of the notable aspects of the interactions between the building block instances depicted in this diagram.>*
    - TODO: add other runtime scenarios.

6.2. <Runtime Scenario 2>

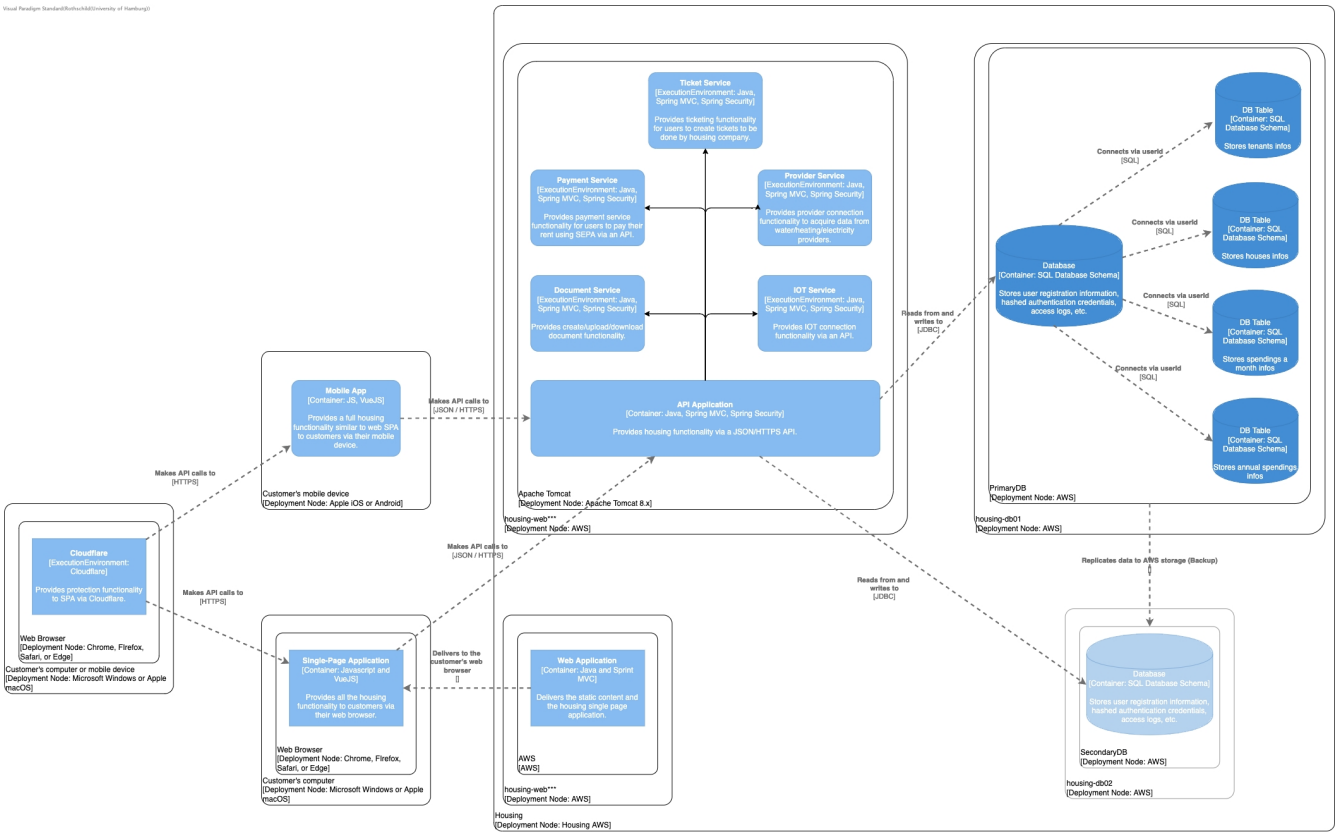
6.3. ...

6.4. <Runtime Scenario n>

7. Deployment View

7.1. Infrastructure Level 1

Main deployment diagram for the Housing System.



<Overview Diagram>

Motivation

<explanation in text form>

Quality and/or Performance Features

<explanation in text form>

Mapping of Building Blocks to Infrastructure

<description of the mapping>

7.2. Infrastructure Level 2

- TODO: add other level of infrastructure if needed.

7.2.1. *<Infrastructure Element 1>*

<diagram + explanation>

7.2.2. <*Infrastructure Element 2*>

<*diagram + explanation*>

...

7.2.3. <*Infrastructure Element n*>

<*diagram + explanation*>

8. Cross-cutting Concepts

8.1. <*Concept 1*>

<*explanation*>

8.2. <*Concept 2*>

<*explanation*>

…

8.3. <*Concept n*>

<*explanation*>

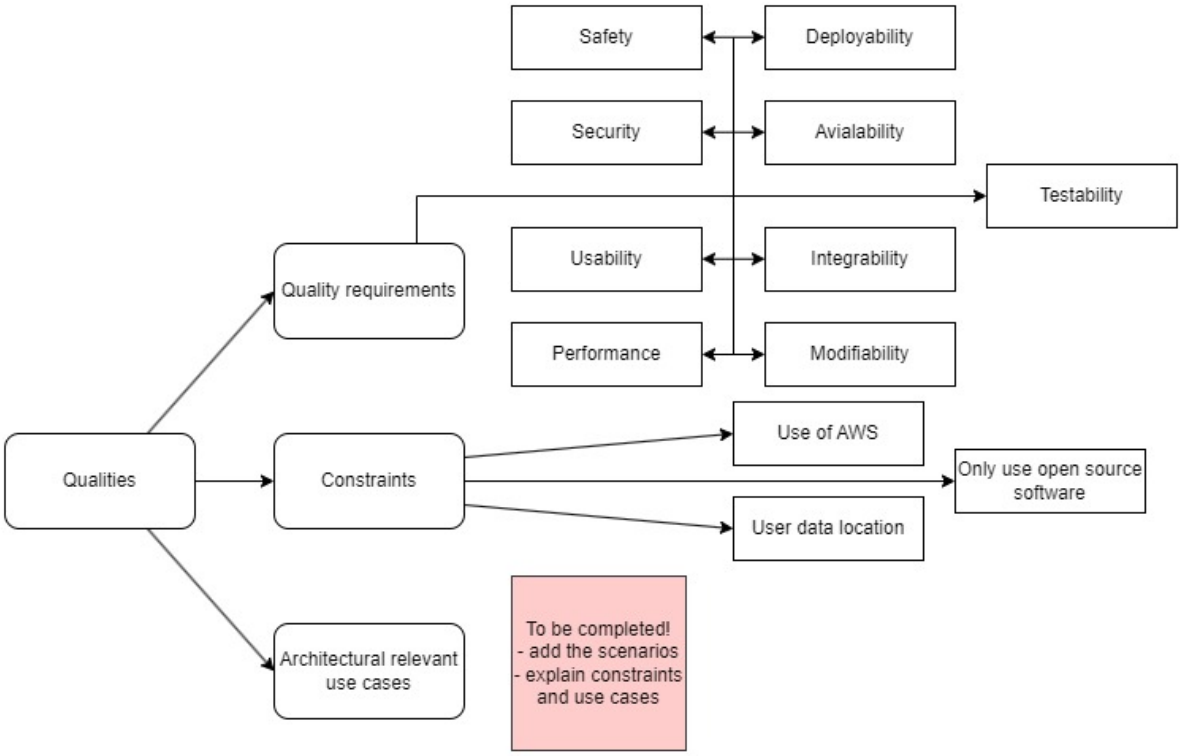
9. Architecture Decisions

9.1. Quality Requirements

Quality Attribute	Changeability	Flexibility
Availability	*low*	*low*
Deploy-ability	*low*	*high*
Integrability	*medium*	*medium*
Modifiability	*medium*	*high*
Performance	*low*	*medium*
Safety	*low*	*low*
Security	*low*	*low*
Testability	*high*	*high*
Usability	*high*	*high*

10. Quality Requirements

10.1. Quality Tree



10.2. Quality Scenarios

11. Risks and Technical Debts

12. Glossary

Term	Definition
<Term-1>	*<definition-1>*
<Term-2>	*<definition-2>*