Documentation

Team: g1t2

Member 1: (Christina Brückl, 01219643)
Member 2: (Leonie Katharina Geyr, 11778679)
Member 3: (Thomas Krabichler, 11906966)
Member 4: (Kevin Rimml, 12036261)
Member 5: (Ioannis Deligiannidis, 11913541)

Proseminar group: PS-703080-1, Team2

Datum: 23.06.2022

Table of contents

1. Introduction and Goals

1.1 Assignment

This software is a complete solution for immediate and long-term climate monitoring in office buildings. Arduinos equipped with different sensors are responsible for measuring temperature, air quality, humidity, noise pollution and light intensity. The devices are equipped with light-emitting

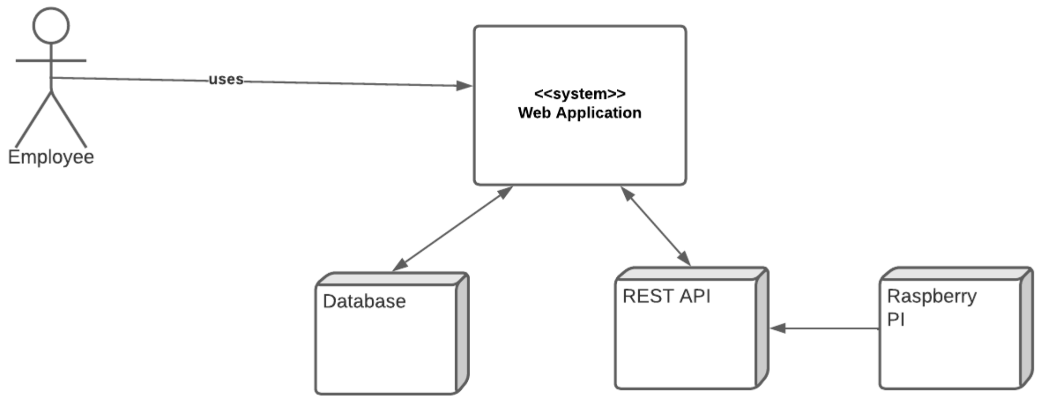
diodes that provide feedback about measurements. Thus people in the room can be notified when any of the measured data exceeds a configurable limit.
The collected data is saved and can be accessed on a web-based service. Users have insight in daily, weekly and monthly reports in their own office, hallways and recreation rooms within their department. Heads of Departments additionally are provided with for all rooms within their department. Higher Management and Facility Administration have insight on all data available.
Further features of the web service include user management, sensor configuration as well as a absence system, where users can record when they are not in office, to take into account when evaluating data according to data protection laws. Users can subscribe to mail notifications with daily/weekly/monthly room climate reports.

1.2 Quality goals
Functional system including all features tasked
Efficient and easy to use user interface
Good scope of unit testing to ensure correct behaviour

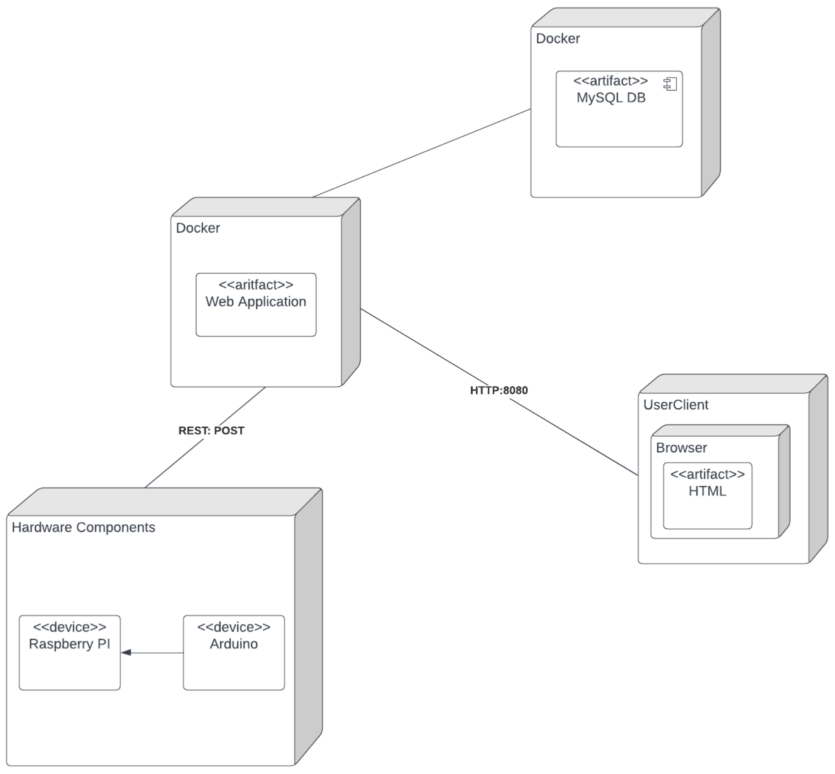
2. Constraints
2.1 Technical
Usage of provided hardware: The system must rely on the hardware given, which is an Arduino with sensors and a Raspberry Pi computer.
Platform independent operation: The web application must run on all commonly used systems and browsers. The system as a whole must be easily runnable by usage of a Docker image.
Usage of permitted technologies: Java, Spring, PrimeFaces, MySQL, Docker for the web application. C or C++ for programming of the Arduino.

2.2 Organisational
Timeline: Finished product by 19.05.2022 for acceptance tests, final results including documentation by 23.06.2022
Development tools: Eclipse or IntelliJ
Version control: GitLab
Testing: JUnit Framework

3. Context and Scope
3.1 Business Context



Employee: Employee can use the system to view all sensor measurements such as: Temperature, noise level, light intensity etc. for his workplace. Furthermore, he can enter and also view his absences.
Database: Stores all measurements of the sensor data transmitted by the Raspberry PI and the entire data of the employees.
REST API: For communication between the end system and the Raspberry PI.
Raspberry PI: Receives the sensor data and communicates with the system using a REST API, receiving measurements from the Arduino.
3.2 Technical Context

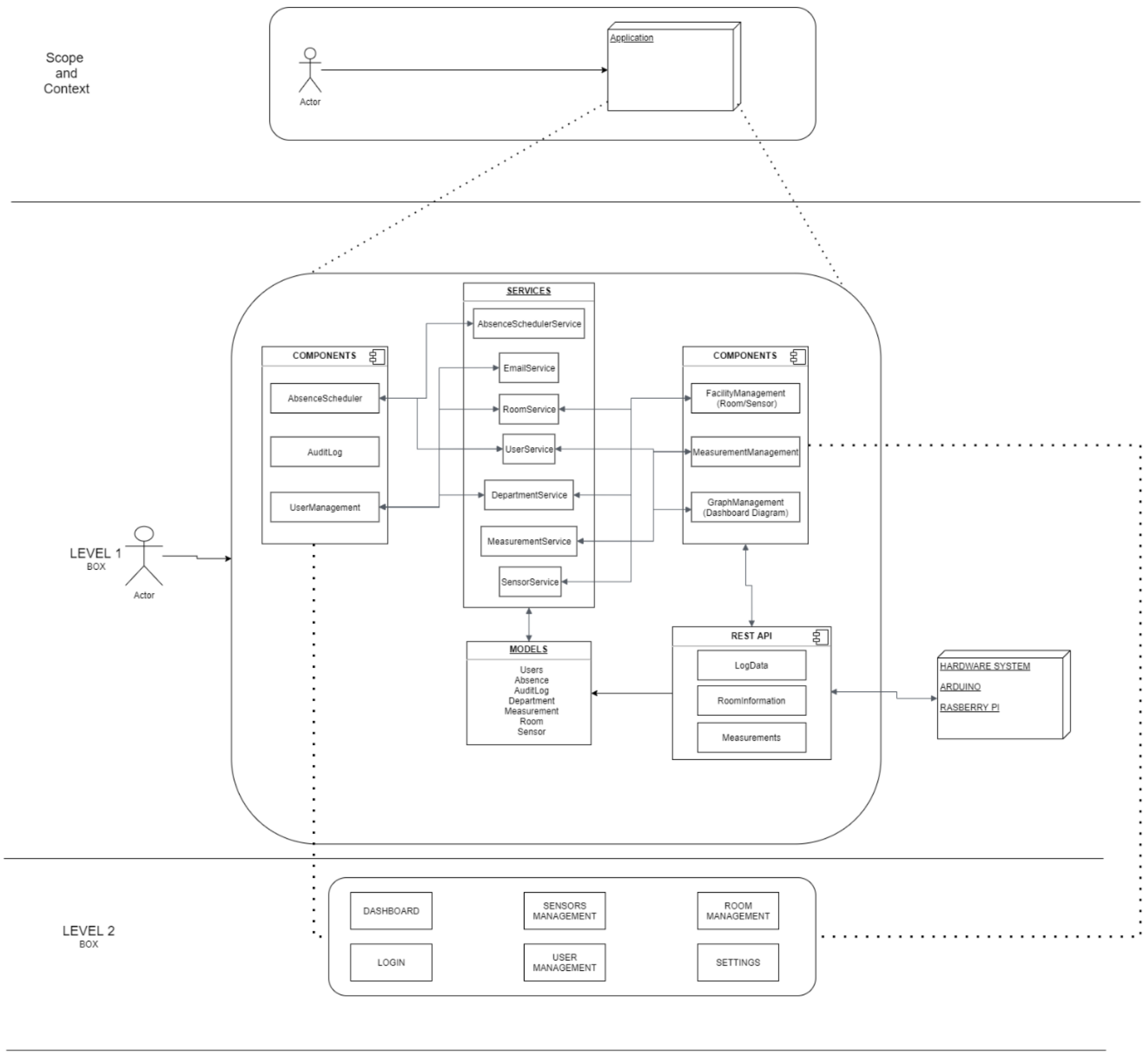


4. Solution Strategy

Goal/Requirement	Architectural Approach

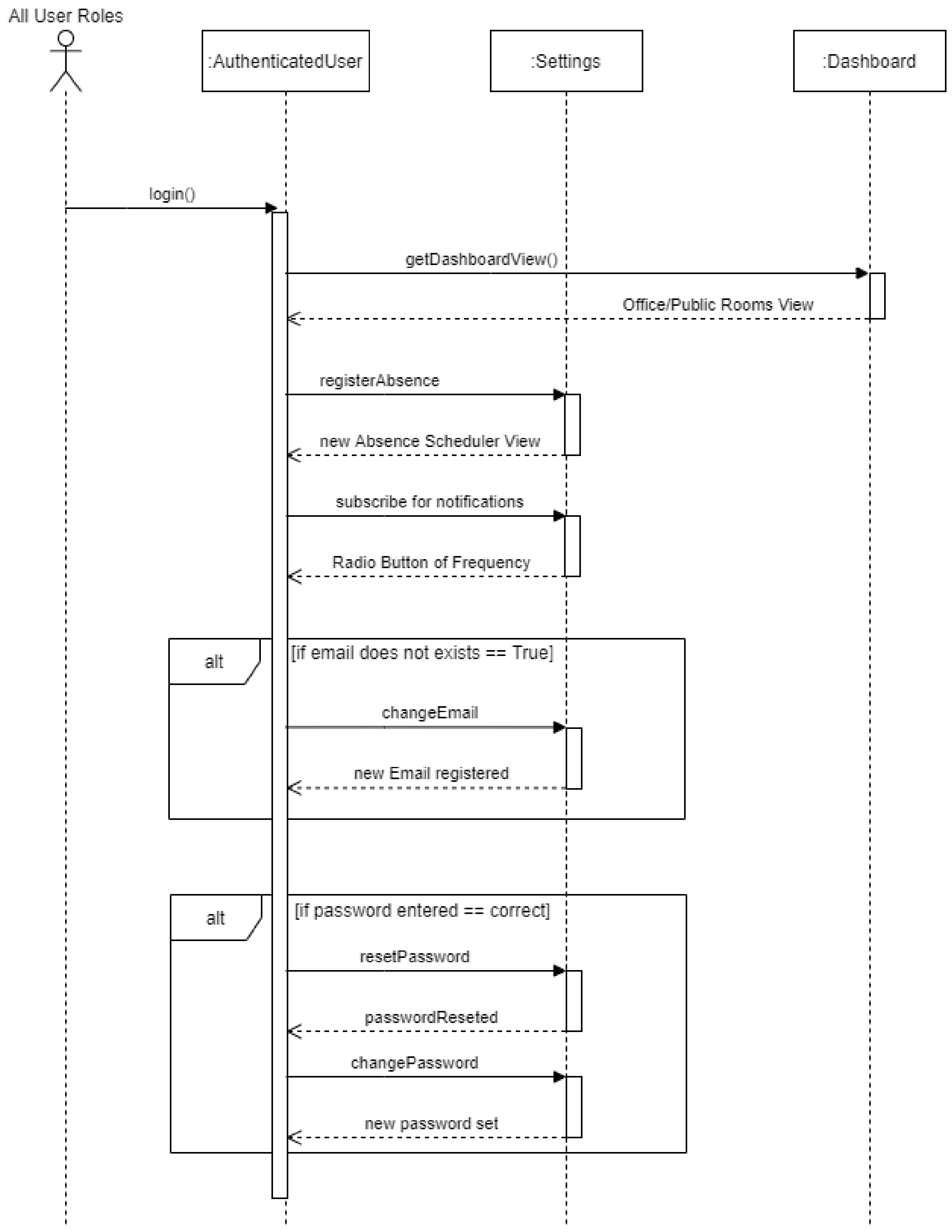
Automatic Transmission of Data (from Sensor to Arduino)	Bluetooth Technology is used for transmission of the measured values
Automatic Transmission of Data (from Arduino to Backend)	data is transferred collectively in given intervals via REST Interface
Persistent storage of data	Data is processed in the Backend (Policy Checks) and stored persistent in Database
Privacy Policy (Like stated in the assignment)	Different Roles and Access Authorization for Users as well as checks if policy is satisfied before data is stored in DB
Tracking of Absences	Absences can be entered into the System by the user, where it is stored in the database
Reports via Email	The stored data is summed up (monthly, weekly, daily) and sent to the users who subscribed for email reports
Visualization of collected Data	Users can access the data for given rooms on their "Dashboard" and see visualizations of the values over time in graphs

5. Building Block View

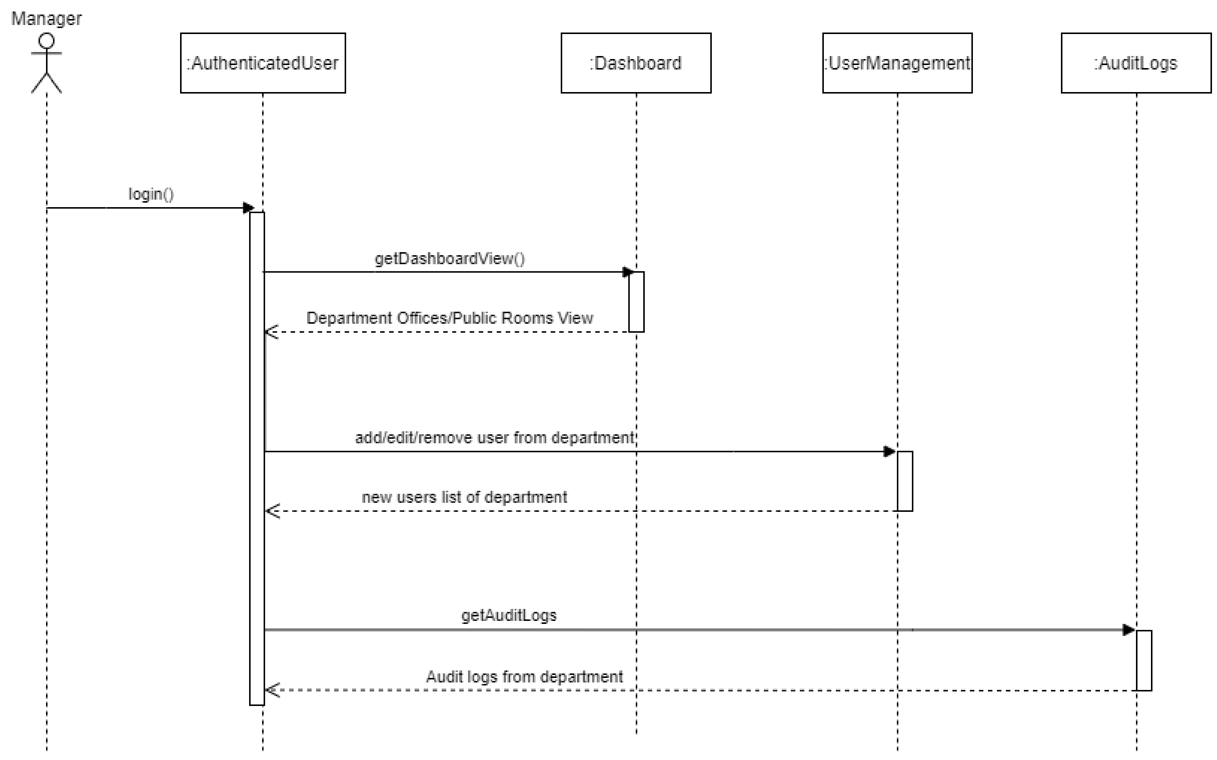


6. Runtime View

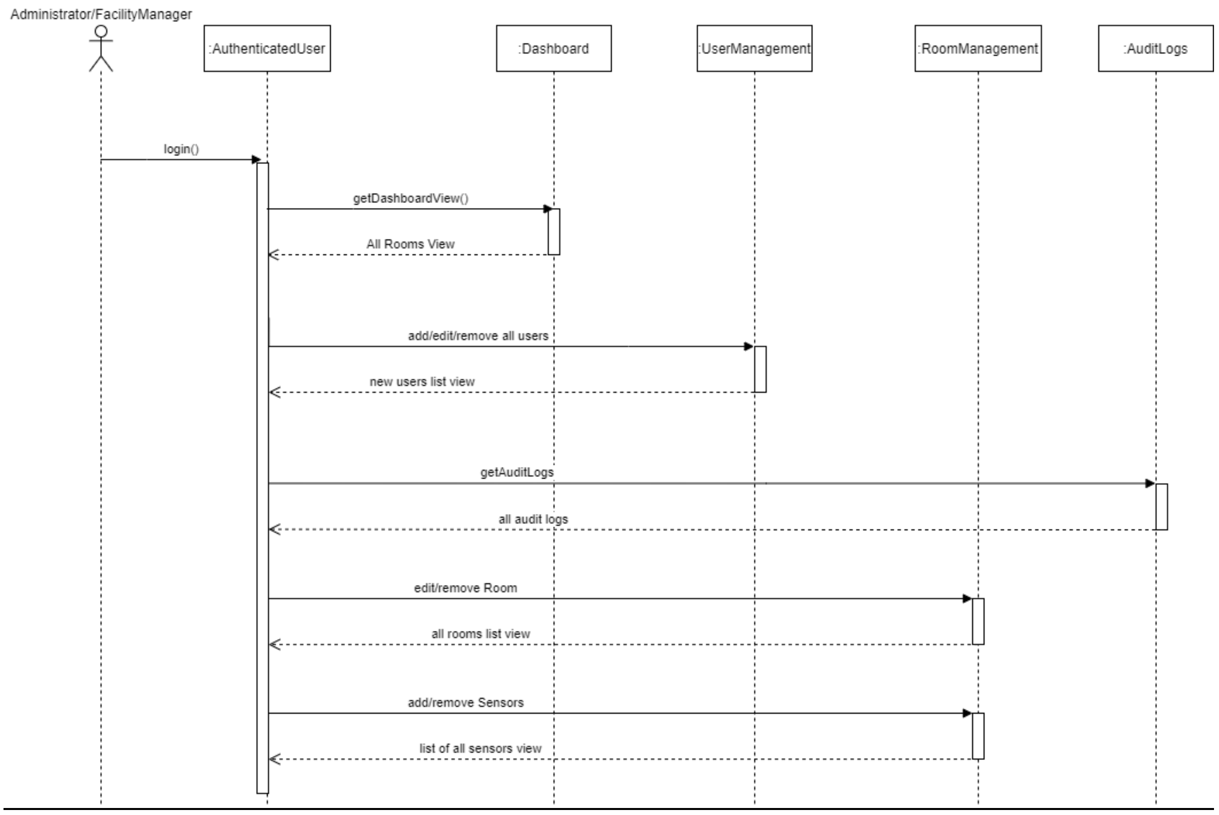
RUNTIME VIEW OF ALL ROLES



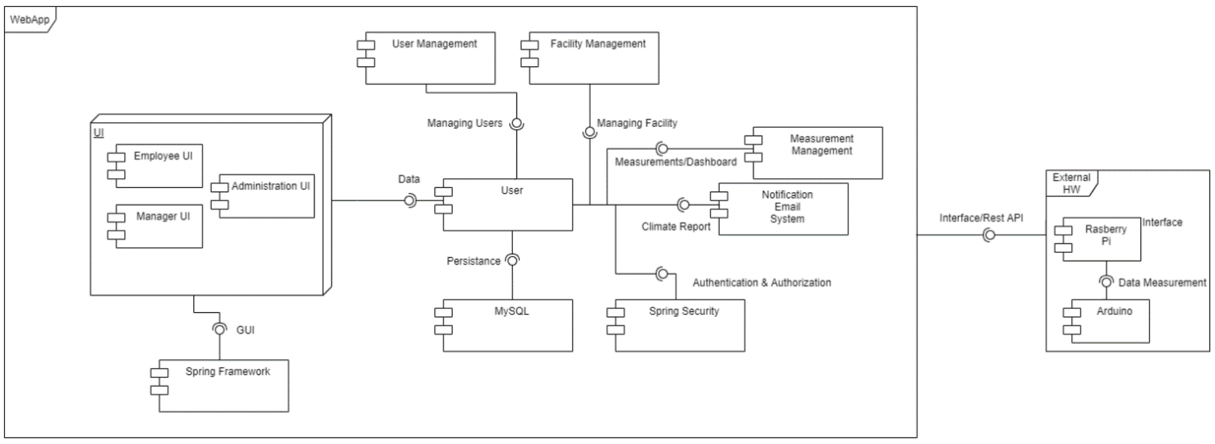
RUNTIME VIEW OF ALL ROLES INCLUSIVE MANAGER


Manager
:AuthenticatedUser
:Dashboard
:UserManagement
:AuditLogs
login()
getDashboardView()
Department Offices/Public Rooms View
add/edit/remove user from department
new users list of department
getAuditLogs
Audit logs from department

RUNTIME VIEW OF ALL ROLES INCLUSIVE ADMINISTRATOR/FACILITY MANAGER



7. Deployment View



8. Concepts

Domain Concept

User Experiences Concept (UX)

The web application surface should give the user an easy-to-handle experience. To guarantee this, necessary information is on not many subpages to make an easy navigation through those. Icons should make navigation easier to understand and black background makes the experience for the user pleasing, as the normal white screen can hurt after 6h+ on the computer. One graph which includes all six measurements of one room (average) should make a handling of information easier for the user.

Safety and security concept

To guarantee safety and security the login mechanism and all subsets are regulated to the user. So if he is within a specific user group only the view of this group is possible. Hardcoding http addresses is therefore not possible. Only Administrators and Manager can change the affiliation of a user to a group. Deleting of users is possible, but needs a confirmation of the admin, so there for accidental deletion is not possible.

Architecture and design patterns

For the architecture pattern, the three-layered pattern was chosen, as in the previous project the team had knowledge about the pattern and the pattern was appropriate for this project.
Some Design patterns:
Observer Pattern, e.g., users can subscribe to get email notifications. Here the push method is used.
Factory Pattern, e.g., measurements data being saved by their class and for each sensor there exists new measurements.

"Under-the-hood" concepts

Error handling: error within the web app is handled via pop-up notifications instant, Error handling within IoT Connections is handled with constant check of current data time, notification via E-Mail to reach the necessary user groups is implemented.
Transaction handling: content loops check and reinstate connections to secure necessary workflow within IoT, an automated timer secures data transfer via Rest API every hour
Communication handling: concurrent connection with the database, insurer a data flow and saving of data, power outage, after restart sends an automated notification to the necessary user group

Development concepts

Concurrent Testing ensures a running system and checks immediately for bugs to fix. Teams were formed to guarantee a consistent code and workflow for the developers in their used environment. Migration to dev branch for constant updating and check for the other teams. After team leaders approve, migration to main branch.

Operational concepts
Administration: Set Up of the project should/has limited administration needed. Admin is used for adding IoT to Network and/or webapp. As Limitations a currently set up in the pi and are customized to the working law of Austria, regular changing is not needed and in smaller offices should therefore be checked with the leader first. Set-up of IoT requires minimalistic knowledge of coding. Administration of web app requires no knowledge of coding.

Disaster-Recovery: Persistent data saving of data within privacy protocols.

Monitoring/Logging: Data, which meet the requirement, get instant saved in the persistent database (MySQL), for the Administration logging of user behavior (required in the project), no invasive information like what changed (privacy), IoT: Arduino Pi Logs data in file set-up concurrently to saving in a non-persistent list. This guarantees in case of a power outage, data gets logged (incl. timestamp), after power is reinstated, log data gets immediately transferred via rest API.

9. Architecture Decisions

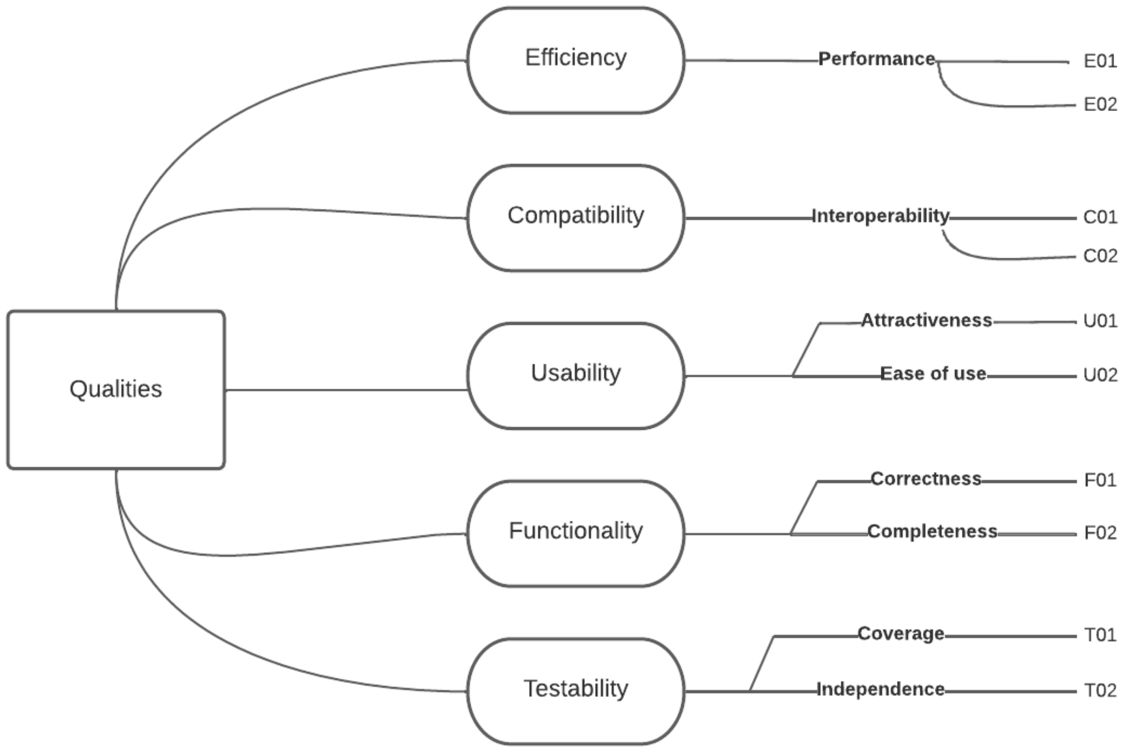
Title: Raspberry Pi connecting to backend of web app
Context: for constant data flow the pi has to be connected to the backend (rest api)
Decision: a connection per se is not made, the pi sends hourly the data through the rest api, backend only accepts that info, if sensor data is integrated in the web app, so outsider can't send and get accepted
Status: accepted
Consequence: sensor id has to be in the web app, or otherwise, data can't get processed

Title: limits set for measurements
Context: every measurement needs limits and Arduino has to act accordingly(LED)
Decision: currently limits are hardcoded in the pi, as the are regulated by law in Austria, so editing is possible but should not be done regularly, pi gives info to web app about the limits for display
Status: accepted
Consequence: changing limits requires limited structure knowledge of pi file system and coding

Title: Graphs in webapp and measurement saving
Context: best display of graphs for readability and usability
Decision: one graph shows one room with all measurements and the average of all sensors in that one room of all data for each measurement
Status: accepted
Consequence: users with more rooms to view, have easier understanding and fewer graphs, database has more space left and performance stays good

10. Quality

10.1 Quality Tree



10.2 Quality Scenarios

ID	Scenario
E01	Every page that is viewable by any user, gets rendered instantly. Also, several charts load the content immediately.
E02	The complete system is easy and fast to setup.
C01	With the usage of REST APIs and dockerization, the system is independent from each component. Therefore, the exchange of Raspberry PIs and Arduinos is intuitive.
C02	As the frontend and backend part are separate from each other, a future change of the frontend framework is also possible.
T01	By using SonarQube during the development process, a 75% test coverage is ensured.
T02	Most of the system components are completely independent and therefore grant a good testability.
U01	The charts are designed in a very user-friendly way, so that the users can view and understand the needed information instantly.
U02	The webapp is implemented fairly straightforward and therefore, all the needed information can be found straight away.
F01	By implementing every edge case found and testing, the correctness of the system is granted.
F02	All of the features are implemented in a high quality, ready to use for any company.