Contents

Introduction

The Fourth Industrial Revolution is transforming our interaction with technology, driving increased automation, digital connectivity, and intelligent decision-making across various sectors, including the built environment [1]. As a result, residential and commercial buildings are undergoing a significant shift, with the Internet of Things (IoT) playing a central role in developing smart, efficient, and secure infrastructures. Researchers have leveraged IoT technologies to enhance real-time monitoring, automation, and control of building systems, thereby improving occupant comfort, energy efficiency, and security.

In recent years, the development of smart building (SB) control systems has surged. These systems connect monitored environmental variables—such as temperature, humidity, luminosity, and air quality—with building management systems like heating, ventilation, and air-conditioning (HVAC) and lighting systems to optimize indoor environmental conditions [2], [3]. The focus of most studies has been on optimizing energy efficiency, as buildings consume approximately 40% of global energy to maintain healthy and comfortable indoor environments for occupants, who spend over 90% of their time indoors [4]. Research demonstrates that occupancy detection can lead to energy savings of 39.4% for HVAC systems [5] and up to 75% for lighting systems [6]. Given these significant potential savings, an increasing number of buildings are being equipped with IoT sensors.

The global adoption of smart building technologies is expanding rapidly. By 2026, the number of buildings deploying these technologies is expected to reach 115 million, up from 45 million in 2022, representing over 150% growth [7]. This increase is driven by rising energy costs and the growing demand for energy efficiency among businesses and residents. Non-residential smart buildings will account for 90% of global smart building expenditures in 2026, maintaining a similar share to 2022. Consequently, it is reasonable to assume that most new buildings will be equipped with IoT sensors. However, while smart technology can optimize energy consumption, an often-overlooked issue is the waste of resources due to unused or underutilized spaces.

In the U.S., office buildings account for approximately 4 billion square feet of real estate [8]. At an average electricity cost of $2.10 per square foot, the total annual utility expense amounts to $8.4 billion. Research indicates that nearly 40% of corporate office space is paid for but remains vacant—unused conference rooms, empty desks, decorative lounge areas, and even entire unoccupied floors. This inefficiency results in an estimated $3.36 billion wasted on electricity to power, heat, and cool unoccupied spaces. This corresponds to approximately 32 billion kilowatt-hours of wasted energy and more than 22 million metric tons of CO2 emissions annually.

In summary, buildings consume vast amounts of energy, making efficiency improvements crucial. While IoT devices enhance energy management by optimizing

various systems, underutilized spaces remain a significant source of waste. Moreover, improving the utilization of existing resources can prevent unnecessary new developments, leading to long-term savings. Therefore, decisions regarding space utilization should be based on data rather than intuition to ensure efficient use of resources.

To optimize resource usage, decisions should be based on data rather than intuition. While intuition can provide valuable insights, relying solely on gut feelings can lead to inefficient use of resources. For instance, studies indicate that many decision-makers prioritize intuition even when presented with contradicting evidence [9]. However, data-driven approaches offer a more reliable foundation for optimization. A survey of over 1,000 senior executives conducted by PwC found that organizations leveraging data effectively are three times more likely to report significant improvements in decision-making compared to those relying less on data [10]. This underscores the necessity of using data to analyze space utilization rather than assumptions, ensuring that underutilized spaces are identified and addressed efficiently.

Occupancy detection systems have been employed to address this issue, with commercial solutions capable of monitoring and displaying space or desk utilization in real time. These systems are highly effective for buildings with clear functional purposes—such as offices, where people work, or meeting rooms, where discussions take place. However, in more complex, multi-functional buildings, room usage is not always straightforwardly linked to function.

AAU Innovate, established at Aalborg University in 2022 to foster innovation, collaboration, and entrepreneurship, provides a dynamic, multi-purpose environment. This study, undertaken in collaboration with AAU Innovate, explores resource usage within this unique setting. A key challenge in distinguishing it from traditional office spaces is that the actual activities taking place often cannot be determined simply by looking at a room's intended function.

The challenge of effectively managing resources in dynamic environments like AAU Innovate, where simple occupancy metrics fall short, highlights the need for advanced solutions such as digital twins. Applying this technology, however, brings forth fundamental questions critical to its successful implementation. Central among these is how a digital twin can move beyond mere presence detection to accurately map and analyze resource utilization based on specific activities and user interactions, which in turn requires clarity on the necessary data sources, analytical techniques, and data handling infrastructure. Equally important is understanding what specific capabilities define truly actionable insights in this context, ensuring the model provides value beyond basic reporting for operational decisions. Furthermore, significant considerations involve how such models can be designed for potential replication or adaptation across other campuses and similar facilities to address broader applicability and scalability needs. This links closely to the challenge of ensuring these potentially complex systems remain accessible and user-friendly, ultimately serving as viable platforms for future research endeavours and practical,

Add overview of chapters y management.

Chapter 1

Background

The background of this study comprises two main parts. First, the state of the art in smart building technologies is presented in Section 1.1, focusing on research closely related to the topic at hand. Second, a detailed description of the AAU Innovate building and its structural characteristics is provided in Section 1.2. This understanding of the building's layout and usage patterns informs the assumptions made later in this project.

1.1 State of the Art

Recent advancements in smart building technologies have explored various approaches to energy optimization, occupancy detection, and digital twin implementations. These studies highlight the increasing significance of IoT, sensor networks, and machine learning in improving building management and efficiency. This section delves into three key research areas: energy optimization, occupancy detection systems, and WiFi-based indoor positioning. While each represents a broad and active field of inquiry, the discussion below focuses on the research most directly relevant to the scope of this work.

One notable contribution comes from K. Bäcklund, P. Lundqvist, and M. Molinari [11], which integrates multiple aspects of the questions asked in the introduction. It presents a novel approach to digital twin technology for higher educational buildings, emphasizing a human-centric design that caters to both building occupants and operators. The study showcases a digital twin developed for a Swedish university campus, integrating 3D scanning, geospatial data, and IoT sensors to create a real-like navigational indoor environment. Among the various components of the digital twin, the building analysis module is used to optimize space utilization, which optimizes space utilization by analyzing occupancy data and room booking systems to identify underutilized areas and "no-show" rooms. This module provides insights into how, when, and where spaces are used, enabling building managers to improve room allocation and reduce energy inefficiencies associated with underutilized spaces. Additionally, the digital twin includes features for real-time navigation, room booking, and building operations management. It addresses challenges in indoor environmental quality and operational efficiency, demonstrating the potential of digital twins to enhance building management and sustainability.

1.1.1 Digital Twins and IoT for Energy Optimization

The integration of digital twins with IoT systems has been widely studied to improve energy efficiency in buildings. Z. Ni, C. Zhang, M. Karlsson, and S. Gong [12] introduced the Deep Energy Twin, which combines ontology-based digital twins with deep learning to forecast energy consumption. Their case study in a historic Swedish building demonstrated improved prediction accuracy, with the Temporal Convolutional Network (TCN) model excelling in point forecasting.

Cost-effective approaches have also been developed, particularly for buildings without pre-existing sensor networks, which, as discussed in the introduction, can be the case in older buildings. S. Anik, X. Gao, N. Meng, P. Agee, and A. McCoy [13] proposed BDL, a system based on Raspberry Pi and modular sensors for indoor environmental monitoring at approximately $73 per building zone. This approach offers an affordable and scalable alternative to traditional sensor installations, making IoT-based energy optimization accessible to a wider range of buildings.

Furthermore, data silos in the built environment sector remain a significant barrier to effective energy analytics and building management. D. Hugo *et al.* [14] emphasize that data from BMS, IoT sensors, and metering systems are often stored in isolated, proprietary formats, creating interoperability challenges and limiting the potential for data re-use. To address this, the author introduced the Data Clearing House—a semantic platform designed to integrate heterogeneous data from BMS and IoT sensors. This platform facilitates seamless data exchange and analysis, enabling more efficient energy management strategies. However, challenges such as data health issues (e.g., equipment faults, incorrect timestamps, and mapping discrepancies) and the need for ongoing semantic model maintenance highlight the complexity of achieving scalable, interoperable solutions in this domain.

1.1.2 Occupancy Detection Systems

Accurate occupancy detection is a crucial component of energy optimization and space management. Various studies have explored sensor-based and machine learning-driven methods to achieve high detection accuracy.

For instance, A. Floris, S. Porcu, R. Girau, and L. Atzori [15] developed an IoT-based system utilizing low-cost hardware, such as Raspberry Pi devices with cameras and environmental sensors. Their lightweight neural network achieved 99.5% classification accuracy for room occupancy detection and accurately estimated Total Volatile Organic Compounds (TVOC) levels via regression analysis.

A comprehensive review by Z. Chen, C. Jiang, and L. Xie [16] analyzed different sensor technologies—including Passive Infra-Red (PIR), CO_2, cameras, and Wi-Fi—and highlighted the effectiveness of sensor fusion and machine learning techniques, such as decision trees and neural networks, in enhancing real-time detection accuracy.

Another study, S. S. Kumar, R. Chandra, and S. Agarwal [17], introduced a rule-based system that employs decision tree algorithms with predefined thresholds for parameters like light intensity and CO_2 levels, achieving precise occupancy classification. Additionally, K. H. Andersen *et al.* [18] utilized supervised learning with indoor environmental quality sensors and Extreme Gradient Boosting (XGBoost) models, demonstrating that both global and room-specific models can effectively capture occupancy patterns, particularly in residential settings.

Recent research goes even further by attempting to estimate human activities in smart homes. Addressing challenges like data variability and sparsity, studies propose the use of digital twins, such as the VirtualSmartHome simulator, to generate realistic synthetic datasets. These datasets model daily activities, enabling human activity recognition (HAR) algorithms to achieve robust performance, with an average F1 score of 80% on real-world data. This approach demonstrates significant potential for improving HAR in smart home environments.

1.1.3 Wi-fi-based indoor Positioning and Occupancy Mapping

Wi-Fi-based systems offer a scalable and cost-effective solution for occupancy mapping and indoor positioning. I. P. Mohottige, H. H. Gharakheili, T. Moors, and V. Sivaraman [19] used Wi-Fi session logs and machine learning to estimate classroom occupancy, achieving a 13.10% symmetric mean absolute percentage error (sMAPE) accuracy. Meanwhile, V. Bellavista-Parent, J. Torres-Sospedra, and A. Perez-Navarro [20] reviewed Wi-Fi-based indoor positioning systems, noting that artificial neural networks (ANNs) dominate the field, with Deep Neural Networks (DNNs) achieving a mean error of 0.11 meters using mmWave signals. Challenges like signal interference persist, but advancements in ML and signal parameters like Channel State Information (CSI) and Signal-to-Noise Ratio (SNR) show promise.

Additionally, Y. Wang and L. Shao [21] analyzed Wi-Fi-based positioning in a university library, identifying four occupancy patterns over 30 days. The study found most users were short-term visitors, suggesting adjustments to opening hours could improve efficiency. This highlights Wi-Fi-based systems' ability to provide granular occupancy data, aiding in optimizing space utilization and energy efficiency.

1.2 Organizational Structure of AAU Innovate

AAU Innovate serves as a central hub for fostering innovation, entrepreneurship, and the commercialization of research at the university. The building accommodates two primary entities:

- **AAU Innovation** The core of the facility, dedicated to cultivating innovation and entrepreneurial activities.
- **Institute for Advanced Study in PBL (IAS PBL)** An independent institute primarily requiring office space, currently occupying a significant portion of the second floor.

Within AAU Innovation, three key teams drive its operations:

1. **Team Event & Support (E&S)** Responsible for the daily management and operations of the building, ensuring smooth functionality and an optimal working environment.
2. **Team Student Entrepreneurship (SE)** Focuses on student-driven innovation and business creation. Their flagship initiative, the Startup Program, provides AAU students with support and resources to launch their own businesses.
3.

Technology Transfer Office Plays a critical role in securing research funding and commercializing university research across various departments.

Beyond these core teams, AAU Innovate fosters a vibrant ecosystem by hosting a community of resident startups. These residents, typically students who have progressed through the Startup Program and successfully applied for dedicated "garage" spaces (individual offices within AAU Innovate), are deeply integrated into the building's innovation culture despite not being formal employees. Their consistent presence and engagement contribute significantly to the overall dynamic.

Further contextual information, including an exterior view of the building and detailed floor plans, can be found in Appendix A.

On a typical semester week, AAU Innovate experiences a significant flow of visitors, ranging from 2000 to 3000 individuals. These visits predominantly occur during the standard work week and daytime hours, although activity can extend beyond conventional closing times (e.g., 4:00 PM) due to student engagements and scheduled events.

The building's utility is multifaceted, encompassing a wide array of spaces and functions. It features seven large meeting or event spaces, which regularly host diverse events such as conferences, celebrations, PhD defenses, keynote speeches, community meetups, social gatherings, and numerous internal workshops. These events vary significantly in scale, accommodating anywhere from ten to over 300 attendees. Additionally, the building includes several smaller meeting rooms primarily used for internal collaborations and interdepartmental meetings within AAU.

Furthermore, AAU Innovate houses six specialized laboratories: a design lab, wood lab, metal lab, play lab, audio-video lab, and an IT lab, alongside a dedicated 3D printing room. The building also incorporates various open collaborative spaces, notably the AAU startup flex zone, a shared workspace specifically designed for early-stage startups participating in the Startup Program. Beyond departmental and employee offices, dedicated workspace is also provided to Entrepreneurs in Residence and other specific programs or startups, such as Code Lab, which currently hosts three software startups. The building also features a canteen, which attracts a significant number of people from neighbouring buildings for lunch daily.

1.3 Technologies

This section outlines the core technologies used throughout the project. Each was chosen to support the system's modularity, scalability, and ease of development. The subsections cover GraphQL (Section 1.3.1), Apollo Federation (Section 1.3.2), gqlgen for Go (Section 1.3.3), and containerization with Docker and Docker Compose (Section 1.3.4).

1.3.1 Graph Query Language

Graph Query Language (GraphQL), initially developed by Facebook and later open-sourced, presents itself as a query language for APIs combined with a server-side runtime for executing those queries against existing data sources [22]. It fundamentally shifts the control of data fetching from the server to the client. Central to GraphQL is its Schema Definition Language (SDL), which establishes a strongly typed contract outlining all the data (objects, fields, and their types) that clients

are permitted to request. Clients construct queries to specify the exact fields and relationships they need, sending these requests typically to a single endpoint (e.g., */graphql*). Operations that modify data are handled through mutations, analogous to the create, update, and delete operations in REST. Furthermore, GraphQL supports subscriptions, enabling real-time data updates pushed from the server to the client when specific events occur.

Compared to REST [23], [24], which uses multiple endpoints and fixed data structures, GraphQL offers a single flexible endpoint where clients request exactly the data they need, solving problems of over- and under-fetching. This improves efficiency and supports rapidly evolving frontend requirements. However, it also brings challenges such as more complex server setups, harder caching due to dynamic queries, and difficulties in rate limiting and monitoring. Despite this, GraphQL is well-suited for applications with complex data needs or bandwidth constraints.

1.3.2 Apollo Federation

For larger applications developed by multiple teams or built on a microservices architecture, Apollo Federation provides a powerful approach to managing GraphQL at scale [25]. Rather than maintaining a single, monolithic schema, Federation enables teams to define subgraphs—modular GraphQL schemas owned and maintained independently. These subgraphs are then composed into a supergraph using Apollo's gateway, which serves as the single entry point for client queries. This approach allows teams to work autonomously while contributing to a shared graph, supporting both scalability and domain ownership.

Each subgraph can extend types defined in other subgraphs, enabling rich cross-service relationships without tight coupling. The gateway handles query planning and routing, ensuring that each part of a query is delegated to the correct subgraph. Apollo Federation thus makes it possible to adopt GraphQL in complex systems incrementally and collaboratively, without sacrificing the unified API experience on the client side.

1.3.3 GoLang with gqlgen

In the Go ecosystem, gqlgen is a popular tool for building type-safe GraphQL servers [26]. It takes a GraphQL schema written in SDL and generates Go interfaces for resolvers, along with strongly typed Go models that mirror the schema. This approach minimizes boilerplate and helps catch errors at compile time, leveraging Go's strong static typing. By guiding developers through resolver implementation with generated interfaces, gqlgen improves developer productivity and ensures tight alignment between the GraphQL schema and backend logic. Notably, gqlgen is also compatible with Apollo Federation (Section 1.3.2), allowing developers to implement federated subgraphs in Go that can integrate into a larger supergraph via Apollo's gateway, making it a viable option for distributed GraphQL architectures.

1.3.4 Docker and Docker compose

Docker is a platform that packages applications and their dependencies into portable units called containers, much like a lunchbox holds a complete meal ready to be used anywhere, ensuring consistency across different environments and solving the "it works on my machine" issue by isolating the application [27]. Extending this concept,

Docker Compose acts like a picnic basket for these individual lunchboxes and other items, allowing developers to define and manage multiple related containers simultaneously through a single `docker-compose.yml` file. This enables the seamless setup, linking, and running of multi-container applications, such as a web server, database, and caching service, together as a unified service with a simple command like `'docker compose up'`, simplifying management and ensuring the different parts of an application work in harmony.

Chapter 2

Project Scope and Motivation

In Section 1.1, the state of the art is subdivided into areas such as energy savings, occupancy detection, and Wi-Fi-based indoor positioning. As mentioned in the introduction, empty or underused spaces in commercial buildings represent a significant waste of resources, affecting energy consumption, construction investments, and personnel allocation. To optimize resource utilization, building management must make decisions that are tailored to the building's needs and are based on data rather than intuition—a practice that research has shown leads to markedly better outcomes [10].

This project has been developed in collaboration with AAU Innovate. With the building now just over two years in operation, the first comprehensive reports and insights regarding its usage after the initial startup phase have begun to emerge. It is, therefore, an opportune moment to evaluate the current usage and steer it towards the intended goals.

A major challenge in this context is the lack of objective insights into how spaces are used. Take, for example, the usage of garages at AAU Innovate. As noted in Section 1.2, startup teams have the opportunity to utilize garages within the building. However, there are only eight garages available, in contrast to the 147 startups enrolled in the startup program in 2024 [28]. This disparity makes it crucial to ensure that the garages are allocated to those startups that truly need them and that their usage is effectively monitored. At present, the evaluation of usage relies on what will be referred to as **anecdotical evidence** — essentially, individual observations of activity. Such an approach runs the risk of misrepresenting actual usage; for instance, employees may primarily work during standard weekday hours (8:00 to 16:00), while startups, constrained by academic schedules, might only gather in the garages after 16:00, leading to the false assumption that these spaces are unused.

2.1 Conceptual Model for Space Utilization

To overcome the shortcomings of subjective observations, there is a clear need for an objective *observer* that can supply accurate data to decision-makers. In cooperation with several members of team E&S, the following conceptual model was developed (see Figure 2.1). This model is designed to answer the question: **"What did AAU Innovate facilitate?"**

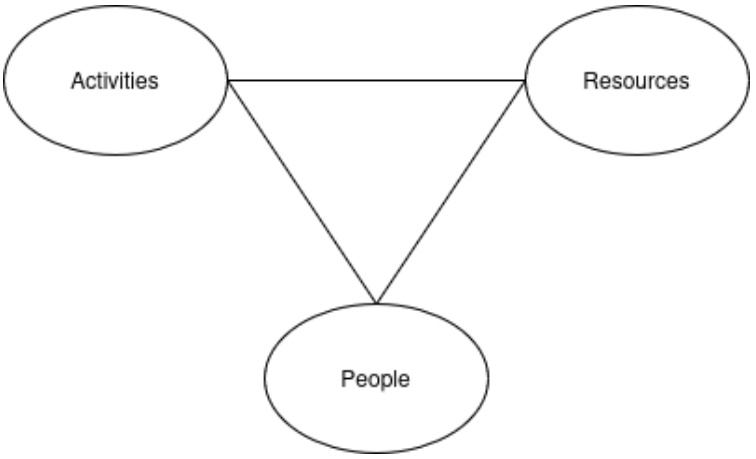


Figure 2.1: Conceptual model for space utilization

The conceptual model is structured around three primary components: activities, resources, and people. These elements form the basis for analyzing space utilization within AAU Innovate. A more detailed discussion of each component is presented in sections 2.1.1, 2.1.2, and 2.1.3.

2.1.1 Activities

Activities refer to the actions or tasks carried out by individuals within the building. These activities have been categorized as follows:

- **Events**, which include:
  - Conferences
  - Celebrations
  - PhD defenses
  - Keynotes
  - Community meetups
  - Social events
- **Workshops**
- **Desk work**
- **Dining**

2.1.2 Resources

Resources encompass the physical and non-physical assets that individuals utilize to conduct their activities efficiently. These resources are classified into the following categories:

- **Rooms**, which include:
  - Meeting rooms
  - Event rooms (e.g., auditoriums, pitch rooms, conference rooms)
  - Labs

    - Offices
    - Garages
    - Shared spaces
- **Equipment**
- **Competences**, referring to human expertise and time availability

A distinction is made between meeting rooms and event rooms based on their intended use and the facilities available. While meeting rooms are typically designed for smaller gatherings (e.g., 2–4 people or 10-person meetings), event rooms cater to larger audiences and are equipped accordingly. Although equipment and competencies could be further detailed, this level of granularity is not necessary for the current scope of this report.

2.1.3 People

Given General Data Protection Regulation (GDPR) regulations and broader privacy concerns, individuals are not represented as unique entities within the model. Instead, people are categorized based on their affiliations, ensuring privacy while still enabling meaningful data-driven insights. The classification is as follows:

- **Building residents**
    - Staff (subdivided by team/department)
    - Entrepreneurs (subdivided by their startup)
- **AAU community**
    - Staff (subdivided by department)
    - Students (subdivided by department)
- **External visitors**
    - Categorized based on their company or domain of affiliation

This approach ensures that decision-making processes focus on broader usage patterns rather than individual behaviours, thereby minimizing privacy risks while still providing valuable insights into space utilization.

2.2 Problem statement

Modern IoT devices continuously gather data without significant bias or downtime, providing a solid foundation for making data-driven decisions at AAU Innovate. Given that the building is equipped with a wide array of sensors to control various systems, a critical question arises:

> **Can already deployed systems and sensors in smart buildings be utilized to create an accurate and detailed picture of building usage?**

This project explores this question as a case study in collaboration with AAU Innovate, seeking to determine whether the current sensor infrastructure can deliver the precise and comprehensive insights necessary for effective building management.

Chapter 3

Overview of Data Systems

This chapter presents a detailed examination of the various data-generating systems operating within the AAU Innovate building. A thorough understanding of these systems, their data outputs, and associated access methods is fundamental for constructing a comprehensive digital twin and enabling nuanced analyses of building occupancy and utilization. The section is structured to provide a clear and consistent overview of each relevant data source.

Each system will be described using a uniform structure to facilitate comparison and ensure clarity. This structure begins with a General Overview outlining the system's primary purpose and the types of data it collects. Following this, the Potential Insights and Assumptions subsection explores how the data from this system can be interpreted within the context of AAU Innovate. This part will involve creating context through the use of dummy assumptions regarding the system's role and data characteristics, followed by a critical reflection on these assumptions to better understand the practical value and limitations of the data for analyzing building usage. Finally, the Data Collection and Processing subsection details how the data can be potentially integrated and accessed, including methods for retrieval and potential data flows.

The following subsections will explore the key data sources that are either currently available or hold potential for future integration. These include the FMS, which manages the spatial layout of all buildings; the Room Calendar System (Section 3.2), responsible for handling reservations of bookable spaces; and the Building Management System (Section 3.3), which oversees environmental controls and sensor networks. Additional insights may be obtained from coffee machine usage logs (Section 3.4), which can offer indirect indicators of occupant activity patterns, and the Door Counter System (Section 3.5), which tracks pedestrian flow at main entrances. The Door Access System (Section 3.6) records entries through secured doors, while the Lighting System (Section 3.7) manages facility-wide illumination —though detailed data for this system is currently unavailable. Lastly, the WiFi System (Section 3.8) provides wireless connectivity and may also serve as a source of location-based data.

3.1 Facility Management System

An Facility Management System (FMS) is a system used to store the layout of all buildings within a given organization. It is typically organized hierarchically, starting from areas, which are divided into buildings, further subdivided into floors, and

ultimately into individual rooms. The system contains detailed information for each room, and it is common for it to include floor plans for each building and floor.

AAU has such a system in place. Each campus is divided into buildings, which contain multiple floors, each with individual rooms. For each room, data such as room number, group, area, and perimeter are provided. The *group* field reflects the intended use or function of the room.

3.1.1 Potential Insights and Data Assumptions

The data within such a system is generally static or changes only infrequently, as modifications to physical building structures are typically time-consuming and resource-intensive. However, the usefulness of the data increases significantly if the following assumptions hold true:

1. Changes made to buildings are also reflected in the FMS system.
2. The room group, or function, aligns well with the actual use of the room.

These assumptions were verified, leading to the following discussions and conclusions about the data value:

- **Assumption 1:** This assumption appears valid. Building modifications are generally large-scale and well-planned operations. As a result, such changes are systematically updated in the FMS, and floor plans are revised when needed. However, it is worth noting that the associated Revit model is not updated during these changes.
- **Assumption 2:** For the Innovate building, this assumption holds in almost all cases. Only a single instance was found where the actual use of a room differed from its designated function—it was used as a meeting room rather than an office. Information about other, particularly older, buildings is not directly available but is assumed to be relatively accurate. Room functions tend not to change frequently, and when they do, modifications often require structural changes, which would typically trigger updates by the same team responsible for maintaining the FMS database.

3.1.2 Data Collection and Processing

AAU is currently in the process of transitioning to a new system, which will allow for API integration. This feature is not yet available in the current system.

For the Innovate building, the author received data via email. This included an Excel spreadsheet containing detailed room data, PDF floor plans for each floor, and a 3D Revit model. These sources collectively provide a relatively static dataset. However, caution should be exercised, as updates made in the live system are not automatically reflected in this data export. While such updates are expected to be infrequent, due to the slow nature of building changes, it is recommended to periodically validate the data to ensure continued accuracy.

3.2 Room Calendar System

Room calendar systems are increasingly vital in commercial buildings featuring bookable meeting rooms, a trend noted by [29]. AAU Innovate utilizes such a system.

Microsoft 365 facilitates this through *room mailboxes*, which function similarly to standard Outlook accounts but are designated as resources [30]. This setup allows meeting rooms to be invited as resources within calendar events, automatically logging the booking in the respective room's calendar.

This system empowers users to check room availability and reserve spaces efficiently. Furthermore, door-mounted screens outside meeting rooms often display upcoming reservations and permit *Ad Hoc* bookings—instant reservations made directly on the system without needing to specify participants beforehand. Specifically, AAU Innovate employs LiveConnect from MediaConnect for functions including room booking and visualizations on these screens. They also utilize AskCody, another meeting room visualization tool. Regardless of the front-end tool, all underlying booking data resides within the AAU Exchange system. While this describes the setup at AAU Innovate, similar resource calendars and visual display systems are standard practice in shared office environments and conference centers.

At AAU Innovate, room calendars are configured for various spaces, such as meeting rooms, event areas, offices, and garages. However, this study disregards the office and garage calendars, as they are currently unused and serve no practical function for analyzing room utilization.

3.2.1 Booking Procedures

The method for booking rooms at AAU Innovate varies by room size. Smaller and medium-sized rooms are typically booked using the standard Outlook *Room Finder* function, where the room's email address is included as a resource invitee in the meeting request.

However, AAU Innovate employs two dedicated staff members to manage bookings for larger meeting rooms (those with a capacity exceeding 12 people). To reserve these spaces, users must submit a detailed request via a Microsoft Form. This form submission triggers a Power Automate flow, which sends an email containing the request details to the designated staff. These employees then manually create the booking in the room calendar, pasting the information from the form into the body of the calendar event.

3.2.2 Potential Insights and Data Assumptions

Room calendar data, stored within Microsoft Outlook, offers valuable insights into room usage patterns, the purpose of meetings, and attendee information. This data aligns with the key aspects of the theoretical model (Figure 2.1), provided the following assumptions hold true:

1. Every use of a bookable space results in a corresponding calendar booking.
2. All individuals attending the meeting (whether physically present or joining remotely) are formally invited via the calendar event.
3. Meeting start and end times recorded in the calendar are accurate and promptly updated to reflect any changes, such as delays, extensions, or cancellations.

These assumptions were verified, leading to the following discussions and conclusions about the data value:

- **Assumption 1:** Observations and interviews indicate that most meetings are indeed booked. Instances where individuals briefly use an empty room without booking it ('just jumping in') are rare. More commonly, users create an Ad Hoc booking directly at the room screen, primarily to avoid interruptions if someone else checks the calendar and sees the room as available. The prevalence of Ad Hoc meetings is also reflected in the booking data.
- **Assumption 2:** Analysis of the data shows numerous bookings with only a single participant listed. While single-person use is possible, it may not align with the intended purpose of a meeting room. This pattern likely occurs when individuals book rooms on behalf of others or invite attendees through alternative channels (e.g., separate emails or different calendar systems) instead of directly via the room booking event. It may also happen for meetings involving external partners not within the AAU system.
- **Assumption 3:** Booked meetings are generally reliable when it comes to reflecting cancellations. However, the scheduled start and end times are less precise. Users sometimes reserve longer durations than strictly necessary to allow for setup or guest arrival buffers. Additionally, meeting overruns are rarely updated in the calendar, with participants typically checking the room's calendar directly to see if it's available for the additional time needed.

3.2.3 Data Collection and Processing

The room booking data is accessible via the Microsoft Graph REST API, specifically using the *events* endpoint [31]. This API provides structured access to room booking details, facilitating seamless integration into the digital twin model. Using this API, it's possible to retrieve events from all relevant room calendars. Since calendar events list invited participants, the application accessing the API can determine the number of attendees and their affiliations, leveraging the personnel information available within the AAU Outlook directory (e.g., department data).

For the large rooms booked via the form process, the detailed information submitted through the form can be accessed within the body text of the corresponding calendar event. This is necessary because inviting potentially hundreds of participants (e.g., for a large event) directly to the calendar entry is impractical; attendees typically receive notifications through other means.

This relatively straightforward data access setup is visualized in Figure 3.1.

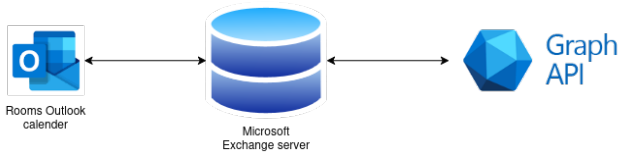


Figure 3.1: A model representing the Outlook room calendar data access

3.3 Building Management System

At Aalborg University, all modern buildings — including AAU Innovate — are managed using the Schneider Electric EcoStruxure Building Operation system for comprehensive building management. This integrated Building Management System

(BMS) platform serves as the central hub for monitoring, controlling, and optimizing key building systems such as heating, ventilation, and air-conditioning (HVAC), lighting, power, and security [32]. By unifying these systems under a single interface, building managers can improve energy efficiency, enhance occupant comfort, and streamline facility operations for smarter, more sustainable buildings. It should be noted that lighting at AAU is managed through a separate system, as detailed in Section 3.7.

The EcoStruxure Building Operation system provides an intuitive interface, typically starting with a campus overview where users can select specific buildings equipped with the system. This report focuses on the AAU Innovate buildings. After selecting a building, users are presented with an overview of various systems, including ventilation, heating, hot water, outside lighting, and sun covers. Each system can be explored individually, or users can switch to a floorplan view to navigate the building spatially. Within the floorplan, individual rooms can be selected to view near real-time data on system and sensor statuses. This detailed room view displays information such as window status (open/closed), heating output percentage, ventilation openness, CO_2 levels, current temperature, acceptable temperature ranges, and the temperature of incoming air. Each room is also equipped with a Passive Infra-Red (PIR) sensor to detect occupancy. A visual example of this room overview interface is shown in Figure 3.2.

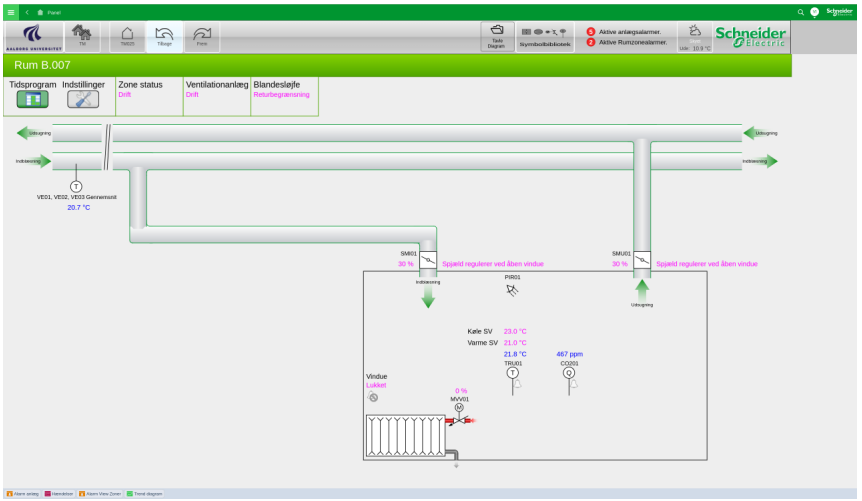


Figure 3.2: A screenshot of the room overview interface of the Schneider EcoStruxure Building Operation software.

In each room, a wall-mounted physical device houses the various sensors and provides a local user interface for adjusting the desired temperature. Open areas within the buildings are similarly equipped with sensors but typically lack the user interface screen.

3.3.1 Potential Insights and Data Assumptions

The data collected by this system holds significant potential for understanding room occupancy and estimating the number of people present. By analyzing the data from the PIR sensor in conjunction with CO2 and temperature measurements and the ventilation state, valuable insights into room usage patterns can be derived. However, the accuracy and reliability of these insights depend on the following key assumptions:

1. The PIR sensor is optimally positioned to detect all movement within the room.
2. Individuals present in a room exhibit sufficient movement to be detected by the PIR sensor.
3. Doors between rooms remain closed unless someone is actively passing through.

These assumptions were verified, leading to the following discussions and conclusions about the data value:

- **Assumption 1:** During testing, the PIR sensor consistently detected movement within its field of view. Typically, this sensor is positioned on the wall next to a door, approximately 1.5 meters above the floor. While this placement generally proves effective in simple, rectangular rooms, the performance of the PIR sensor can be affected by both the layout of the room and the sensor's specific placement. For instance, office *C.108* on the first floor presents a potentially problematic room configuration (refer to Appendix A for the detailed building layout). Consequently, these factors concerning room layout and sensor placement should be taken into account when analyzing the data collected by the PIR sensor.
- **Assumption 2:** A known limitation of PIR sensors is their inability to detect stationary individuals. To mitigate this, a sufficiently long data aggregation interval can be used. For instance, if the interval is set such that even individuals who move every 10 to 15 minutes are counted as 'continuously present', this issue can be partially addressed. However, this approach introduces a trade-off between data granularity and accuracy.
- **Assumption 3:** At AAU Innovate, it is generally observed that doors remain closed, except for a few specific rooms. Furthermore, the doors are equipped with auto-closing mechanisms. To ensure doors remain open when needed, a common practice is to use a physical object to prop them open. This practice aligns with the requirements of the access control system, further reinforcing the likelihood of doors being closed.

3.3.2 Data Collection and Processing

As explained in S. P. Melgaard *et al.* [33], a separate BMS database was established to enable convenient access to system data. Data from the various sensors deployed throughout the building is first collected by local terminals (Automation Servers) within the BMS. From there, the data is forwarded in chunks, rather than at fixed intervals, to a centralized PostgreSQL database, which uses TimescaleDB, a time-series optimized extension for PostgreSQL.

TimescaleDB is an open-source time-series database built on top of PostgreSQL, designed to efficiently store and query large volumes of time-stamped data. It enhances PostgreSQL with features such as automatic data partitioning, continuous aggregation, and optimized time-series queries, making it particularly well-suited for handling sensor data from building management systems.

The transfer of sensor data from the Automation Servers to the AAU BMS database is event-driven, triggered when a certain amount of data has been collected

rather than on a strict schedule. Due to the large number of sensors and the volume of data, it typically takes between 10–15 minutes for data from all logged sensors to become available in the database.

In addition to timeseries data, metadata — including sensor locations, types, and other descriptive information — is also managed through the Schneider BMS platform. Whenever a new sensor or data point is added to the Schneider system, the corresponding metadata is automatically transferred into a dedicated METADATA table within the AAU BMS database.

The system provides access via two separate API endpoints:

- Metadata Endpoint: Returns relevant information about each sensor, including the externalID, source path, and unit. This endpoint does not require any input arguments.
- Trend Data Endpoint: Accepts an externalID, a start time, and an end time as input, and returns a list containing the externalID, timestamp, and recorded value for the specified sensor and period.

An overview of the system setup is shown in Figure 3.3.

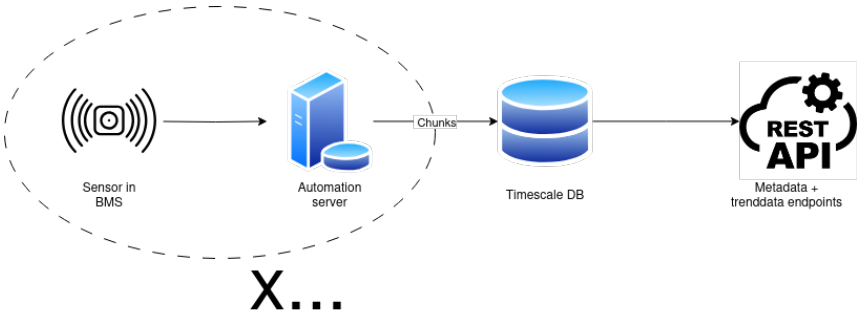


Figure 3.3: An overview of the setup of the BMS data capture system.

3.4 Coffee Data

AAU Innovate, like many other workspaces, provides coffee machines for its residents. Specifically, there are four Wittenborg 9100 ES 2FB machines located throughout the building: one in the kitchen, one in the first-floor hallway, and one each in the hallways on the second and third floors. These machines are capable of exporting usage data either via a USB key or over a serial connection [34]. This data logs every beverage dispensed, effectively maintaining a usage record.

It is important to note that non-residents cannot access the first floor and above, meaning only residents have access to these machines, as mentioned in Section 1.2. Consequently, the data gathered provides, at least in theory, insights into resident behaviour.

3.4.1 Potential Insights and Assumptions

While coffee machine usage data may not seem significant at first glance, it has potential applications. For example, it could offer insights into floor usage patterns or assist in automating maintenance and cleaning schedules. The information would provide the most valuable insights if the following assumptions hold:

1. Individuals obtain hot beverages from the machine closest to their current workspace.
2. Each person consumes a consistent number of beverages daily.
3. Beverage consumption is distributed evenly throughout the working day.
4. Only residents use the coffee machines.

If these assumptions hold, the data can provide a broad and generalized representation of resident movement within the building, making it a valuable dataset. However, these assumptions were verified through discussions with residents and direct observation.

- **Assumption 1:** This assumption generally holds. An exception was observed where individuals working on the ground floor sometimes move up a floor to get their drinks.
- **Assumption 2:** This assumption does not hold well. Significant individual variations, daily fluctuations, and potential seasonal effects influence the data.
- **Assumption 3:** This assumption is inaccurate. Consumption patterns indicate that some periods of the day, particularly in the afternoon, may not be well represented by this data.
- **Assumption 4:** This assumption holds fairly well. Only employees and students in the startup program have access to floors 1 and above. While guests can also use the machines, their contribution is minimal compared to residents.

Overall, while coffee machine data can reflect facility usage by residents and capture usage variations, interpreting it requires caution. Due to the impracticality of frequent data collection and the strong potential for variation, a one-month collection period was chosen to balance out individual fluctuations and provide a general activity overview. Despite this, data interpretation should be done carefully, considering the limitations highlighted by assumption validation.

3.4.2 Data Collection and Processing

The coffee machines store usage data locally, and the most practical method for extracting it is through a USB key. Unlike an automated system, this manual approach limits data collection to weekly or monthly intervals, as more frequent extraction is impractical.

To streamline processing, a computer program has been developed. Once the data is transferred from the USB key, the program automatically uploads it to a database and generates a CSV file for backup. The CSV format ensures easy access for non-technical personnel. This workflow is illustrated in Figure 3.4.

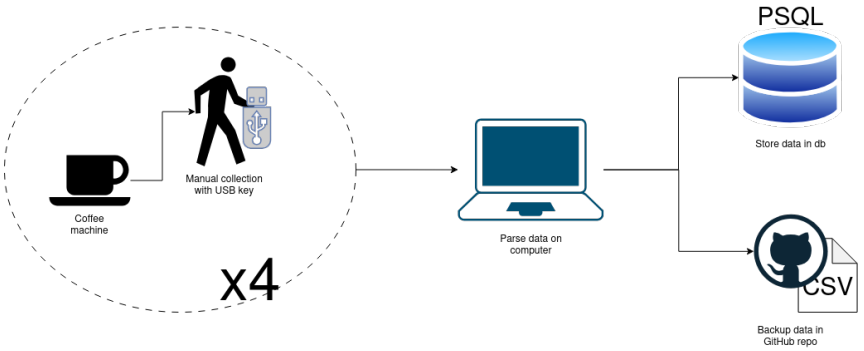


Figure 3.4: A model representing the coffee data extraction process

It is essential to highlight that the automated data extraction, utilizing the USB key and processing software, was developed as part of my student job. Therefore, it should not be considered an accomplishment of this project but rather a pre-existing system.

3.5 Door counter system

To monitor building occupancy, AAU Innovate employs a people-counting system installed at the primary entrances. Specifically, counters are located at entrances A, B, and C, whose positions are detailed in the ground floor plan (Figure A.3).

The system utilizes the MultiTeknik People Counter. While this counter is generally marketed towards retail businesses for tracking customer traffic, typically uploading data via USB to a local computer Multiteknik [35], the implementation at AAU Innovate deviates from this standard. Instead of a USB connection, the AAU Innovate setup transmits data over an unspecified protocol to a Thingsboard server managed by Multiteknik. ThingsBoard itself is an open-source IoT platform designed for collecting, processing, visualizing, and managing data from connected devices, supporting the development of scalable IoT applications. A conceptual diagram of the basic people counter hardware setup, adapted from Multiteknik [35], is shown in Figure 3.5.

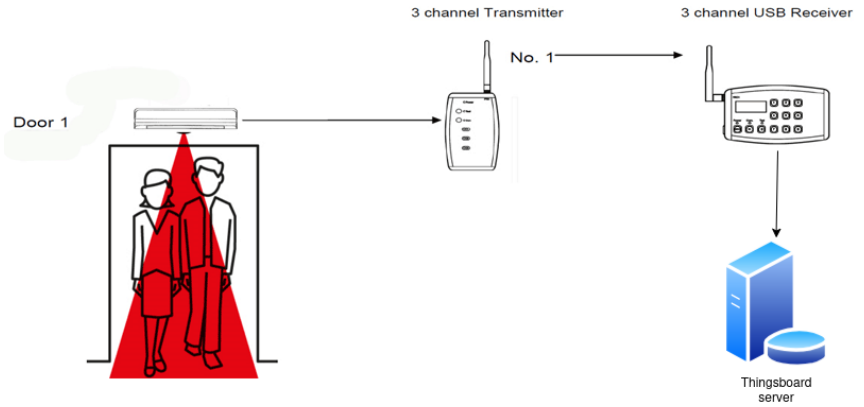


Figure 3.5: A model representing the people counter setup, inspired by Multiteknik [35]

Data captured by the sensors is stored and visualized on the Multiteknik-managed Thingsboard server. The underlying mechanism involves an infrared (IR) beam sensor positioned approximately one meter high just inside each main door, with a corresponding reflector placed opposite. Each interruption of this IR beam registers

as a single "passage". Periodically (specified as hourly), each of the three sensor units (transmitters) sends a value representing the total number of detected passages divided by two to a central receiver. This receiver assigns a unique label (c1, c2, or c3) corresponding to each sensor and forwards this labelled data to the Thingsboard server.

3.5.1 Potential Insights and Data Assumptions

The passage data collected by the sensor system offers valuable insights into the building's usage patterns, particularly the number of people entering and exiting each day. However, the validity of these insights depends on several key assumptions:

1. Visitors' entry and exit occur within the same hourly measurement window.
2. Individuals pass through the sensor detection zone one at a time.
3. All entries and exits occur solely through the three monitored main doors (A, B, C).

These assumptions were verified, leading to the following discussions and conclusions about the data value:

- **Assumption 1:** This assumption is often inaccurate. Visitors frequently stay for multiple hours, and their movements rarely align with hourly reporting intervals. The sensors also lack directionality and cannot distinguish between entries and exits. As a result, a reported value (e.g., 40 passages) may reflect any number of entry/exit combinations. The system attempts to estimate this by halving the count (passages/2), but this method introduces ambiguity and limits its accuracy on an hourly basis. A more reliable approach is to analyze data on a daily level, assuming most individuals leave before the following morning (e.g., using a 04:00–03:59 'day' definition).
- **Assumption 2:** This assumption generally holds. Narrow doorways and social norms around personal space usually ensure that people pass through one at a time. Occasional inaccuracies may arise when individuals pass closely together, but these occurrences are considered infrequent and not a significant source of error.
- **Assumption 3:** This assumption is known to be incomplete. Several alternative routes exist, including a fourth door to a terrace and emergency exits connected to four staircases (Appendix A). The terrace door is often used during working hours (e.g., 08:00–15:30), and emergency exits sometimes serve as shortcuts to parking areas. Movements through these unmonitored routes are not captured by the system and introduce a measurable error in estimating total building occupancy. While the main doors account for the majority of traffic, these alternative paths should be considered when interpreting sensor data.

3.5.2 Data Collection and Processing

The data stored on the Thingsboard server can be accessed programmatically via its built-in REST API. The standard procedure involves:

1. Authenticating using valid credentials (username and password) to obtain an access token.
2. Using the obtained token in subsequent API requests to fetch the desired telemetry data (i.e., the hourly passage counts labelled c1, c2, and c3).

This data retrieval process, forming part of the overall system implementation at AAU Innovate, is depicted conceptually in Figure 3.6.

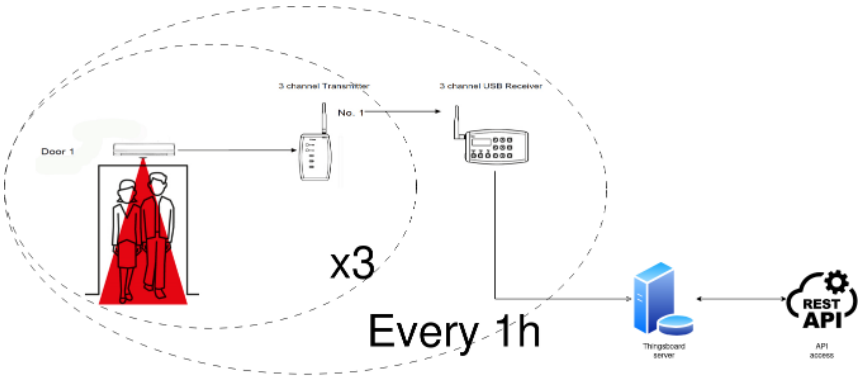


Figure 3.6: A model representing the data collection setup for the people counter system at AAU Innovate, utilizing the Thingsboard API.

3.6 Door access system

Aalborg University utilizes a digital door access system. Scanners are installed at designated doors. Furthermore, every student, employee, and guest (via guest cards) possesses a personal card equipped with an NFC chip. This chip contains the card number. To gain entry, an individual scans their card by holding it in close proximity to the scanner. The scanner subsequently communicates with a central server, signalled by a flickering orange indicator light, which verifies access permissions defined for each door or door group. If access is granted, the indicator light turns green. Conversely, if entry is denied, the light turns red. These visual indicators are accompanied by corresponding audible signals. Successful authentication potentially results in the door unlocking.

Picture c
brand

3.6.1 Potential Insights and Assumptions

Given that each card is personal, the logged data could offer valuable insights into the 'people' aspect (see Section 2.1) concerning individuals within the building. The value of this information is contingent upon the validity of the following assumptions:

1. All doors consistently require a card scan prior to opening.
2. Every individual always scans their card before entry, including when entering immediately after another person.
3. Comprehensive logs are maintained for every card-scanner interaction.
4. Each scanner possesses a unique ID and an associated position description, ensuring clarity regarding its placement within the building.

These assumptions were verified, leading to the following discussions and conclusions about the data value:

- **Assumption 1:** This assumption does not hold universally. Main entry doors are typically open by default during normal office hours (8:00–15:30). Certain doors can also be set to a 'kept open' state, where scanning a card unlocks the door and a second scan relocks it. This functionality is inactive outside of normal office hours.
- **Assumption 2:** This assumption does not always hold. Individuals may hold doors open for others or allow entry without an additional scan. The main access gates to the first floor remain open for approximately 5–10 seconds, making it relatively common for people to enter immediately after someone else. Circumventing the gate by jumping over it is reportedly rare.
- **Assumption 3:** This assumption holds. When equipment theft occurs at AAU, access logs dating back up to six months can be requested for investigation.
- **Assumption 4:** This assumption likely holds. Each scanner must possess a unique ID to enable permission control for specific doors or rooms. It is presumed that scanners are organized into logical groups. Whether detailed location descriptions are associated with scanner IDs is unknown to the author, but it is probable to support maintenance and repair jobs.

3.6.2 Data Collection and Processing

This access data is managed by AAU Campus Security; currently, the author has been unable to obtain access. Access would likely first require establishing an official collaboration agreement with AAU Innovate and other relevant parties. Such an agreement could potentially secure the necessary resources for this project. Pending data access, a hypothetical schematic representing the author's assumptions about the system's operation is provided in Figure 3.7. The specific methods for potential (automated) log file access remain speculative and are likely system-specific.

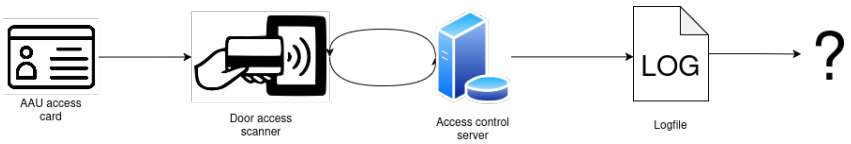


Figure 3.7: A model representing the believed setup of the card reader system.

3.7 Lighting system

Implement when I get more info from company

3.8 WiFi System

Aalborg University maintains a comprehensive WiFi infrastructure, managed by IT support (ITs), which extends to AAU Innovate. This infrastructure comprises three primary networks designed for different user groups.

- First, the `AAU 1-DAY` network serves guests without university affiliations. Access is granted via a daily password, retrievable through the dedicated `AAU 1-DAY` application, which displays credentials for the current day and the subsequent three days. This design facilitates temporary access for visitors.

- Second, the `Eduroam` network is the primary access point for students and staff. Users configure unique credentials for each device they wish to connect. As these credentials are generated while logged into an AAU account, devices connected to `Eduroam` can, in principle, be associated with their respective owners.
- Third, another network exists, reportedly designated primarily for employees. However, the specific details and configuration of this network require further confirmation.

Get more info about this network, at leat the name

This segmented network architecture presents an opportunity for user categorization. Devices connected to `AAU 1-DAY` can generally be assumed to belong to guests, while `Eduroam` connections signify students or staff. A potential challenge in occupancy estimation arises from users carrying multiple devices (e.g., smartphone and laptop). This issue is more pronounced on the guest network, as the device-specific credentialing of `Eduroam` mitigates double-counting for affiliated users to some extent.

Leveraging this WiFi infrastructure for occupancy estimation and indoor positioning offers significant potential, building upon techniques discussed in the state of the art (Section 1.1.3). For instance, even devices not actively connected can contribute to occupancy counts. Research by M. Vega-Barbas, M. Álvarez-Campana, D. Rivera, M. Sanz, and J. Berrocal [36] demonstrates methods using passive WiFi sensing to analyze "probe request" messages broadcast by smartphones. By developing unique fingerprints for devices, this technique overcomes MAC address randomization challenges and estimates crowd sizes with high accuracy (reported near 95%), making it applicable for estimating the total number of individuals present, regardless of network connection status.

Furthermore, data from connected devices can be analyzed. Work by S. Liu, Y. Zhao, F. Xue, B. Chen, and X. Chen [37] introduced DeepCount, a system utilizing deep learning (specifically Convolutional Neural Networks (CNNs) and Long Short-Term Memory networks (LSTMs)) to analyze Channel State Information (CSI) from commercial WiFi devices. This approach infers crowd counts in multi-person indoor environments, achieving notable accuracy (up to 90% with refinement mechanisms) without relying solely on connection logs. Such techniques could potentially be applied to the AAU Innovate environment.

Accurate indoor positioning is another valuable application derivable from the WiFi infrastructure. As surveyed by J. Dai, M. Wang, B. Wu, J. Shen, and X. Wang [38], numerous WiFi-assisted positioning schemes exist, leveraging signal characteristics to determine device location within buildings. Integrating such positioning capabilities with occupancy data, potentially supplemented by alternative methods like mobile connection analysis [39], and knowledge of the AAU Innovate floor plan, could provide a highly detailed spatiotemporal understanding of building usage.

3.8.1 Potential Insights and Assumptions

Effective utilization of the WiFi system data could yield granular, near-real-time estimates of occupant counts within specific rooms or zones. This enables the mapping of movement patterns and density heatmaps throughout the building. A significant advantage of the AAU setup is the potential to categorize occupants based on their network affiliation (`Eduroam` vs. `AAU 1-DAY`). By correlating `Eduroam` usage

with AAU account types (student, staff, faculty), it may be possible to approximate the distribution of different user groups defined in Section 2.1.3 across the building.

This detailed occupancy information holds considerable value for optimizing internal processes, managing resources (e.g., HVAC, lighting), evaluating space utilization, and informing strategic decisions. For large events, accurate visitor counts derived from WiFi data could provide crucial feedback. However, the collection and analysis of such data, particularly in real-time and at an individual level, must carefully address data privacy regulations (e.g., GDPR) and ethical considerations, likely requiring data aggregation and anonymization.

3.8.2 Data Collection and Processing

Given that the precise technical specifications and data logging policies of the AAU WiFi system are not currently known to the author, a detailed description of the data collection and processing pipeline cannot be provided at this stage. Accessing and utilizing this data would require collaboration with ITs and adherence to university data governance protocols.

Chapter 4

Data integration

As mentioned by D. Hugo *et al.* [14] and discussed in Section 1.1.1, data silos remain a significant barrier to research in the built environment. Isolated systems and proprietary data formats make integration difficult, especially when data lacks spatial context. For example, sensors might be identifiable by ID, but determining their physical location, such as which room they are in, is a major challenge when using these systems independently.

This leads to the first challenge of this project: to meaningfully assemble and integrate all deployed systems and sensors. Achieving this not only enhances future research opportunities but also increases the contextual value of each individual data source.

To address this, GraphQL is used as the API technology. As described in Section 1.3.1, GraphQL excels at managing complex data structures, eliminating the need to query multiple, disparate systems. Since the goal is to avoid requiring access to many different systems and APIs, GraphQL is especially well-suited for this use case.

Moreover, using Apollo Federation (see Section 1.3.2) enhances this setup. Each subsystem, outlined in Chapter 3, is implemented as an individual GraphQL graph. This allows for both scalability and maintainability. The end result is an integrated supergraph, where a single endpoint provides access to all available data at AAU Innovate. This approach also supports easy extensibility: adding a new system simply involves creating a new small graph and integrating it into the supergraph, offering high adaptability.

Another advantage of using GraphQL is that it is self-documenting. A common limitation of REST APIs is the reliance on comprehensive documentation for each endpoint and its parameters. In contrast, GraphQL uses a schema that can be explored in tools like the GraphQL sandbox, allowing users to discover available data without external documentation. This feature greatly benefits future research.

This section first discusses the implementation of each individual subgraph, corresponding to the systems outlined in Chapter 3 and described in detail in Section 4.1. For each, the setup and its corresponding GraphQL schema are explained. Following that, Section 4.2 describes the supergraph, showcasing the complete graph model and detailing how Apollo Federation is applied in practice to integrate all subgraphs into a unified API.

4.1 Individual Data Systems

Each individual data system is implemented as a standalone Go server using *gqlgen*, as described in Section 1.3.3. The *gqlgen* tool generates a `server.go` file that can be executed to launch the server. This exposes both the GraphQL Playground at the root domain and the GraphQL query endpoint at the */query* path. The GraphQL schema is defined in `schema.graphqls`, which *gqlgen* uses to generate the corresponding Go types. The logic for handling queries and mutations is implemented in `schema.resolvers.go`.

Additional configuration can be specified in the `gqlgen.yml` file. Notably, federation support can be enabled here, which generates additional files, including `entity.resolvers.go`, where entity resolvers are implemented to support cross-service references. Specific implementations and schema overviews for each system are provided in the following subsections.

4.1.1 Facility Management System

As described in Section 3.1, the current FMS system lacks an API implementation. As a temporary solution to serve this essential information, the system operates using *static* data. This data is periodically received via Excel sheets sent by email and exported from the FMS system.

To simplify server-side processing, the Excel data is converted into a CSV file and placed in the server's file system. On startup, the server reads the CSV file and loads its contents into memory. Since the dataset is relatively small, this approach poses no memory concerns. However, if the dataset were to grow significantly, alternative approaches should be considered, such as reading the CSV file on each request or, preferably, importing the data into a self-hosted database for efficient querying. If performance becomes an issue, caching strategies can be employed to optimize data retrieval.

This setup results in the GraphQL schema shown in Listing 4.1. The schema defines three core types: `Building`, `Floor`, and `Room`. Since this is the first data source in the supergraph and acts as the central connector for other systems, no additional federation types are needed. Only a key definition is required for each of the current types to support entity resolution within the federated architecture.

```
type Room @key(fields: "id") {
    id: ID!
    roomNumber: String!
    type: String!
    area: Float!
    circumference: Float!
}

type Floor @key(fields: "id") {
    id: ID!
    name: String!
    rooms: [Room!]!
    floorplanUrl: String!
}

type Building @key(fields: "id") {
    id: ID!
    address: String!
    city: String!
    property: String!
    floors: [Floor!]!
}

type Query {
    buildings(ids: [ID!]): [Building!]!
    floors(ids: [ID!]): [Floor!]!
    rooms(ids: [ID!]): [Room!]!
}
```

Listing 4.1: The GraphQL schema for the Facility Management System.

4.1.2 Room calendar system

describe 'hack', when a more stable solution is found with the shutdown at midnight

4.1.3 Building Management System

As described in Section 3.3.2, the BMS system exports its data into a database consisting of two main tables: *metadata* and *timeseries data*. This process is illustrated in Figure 3.3. The Timescale DB can be accessed via a REST API that exposes two endpoints—one for *metadata* and one for *timeseries data*.

Authentication is handled through HTTP Basic Authentication, with the username and password included in the request header. Requests are made to *https://bms-api.build.aau.dk/api/v1/* followed by the relevant endpoint. It is important to note that this server is only accessible from within the AAU network or through a Virtual Private Network (VPN) connection.

When the server starts, it sends a request to the *metadata* endpoint and caches the response, as this data rarely changes. This improves performance, since subsequent incoming requests only need to query the *trend data* endpoint and not repeatedly request metadata. The data structure is represented in the GraphQL seen in Listing 4.2.

```
type Value {
    timestamp: Time!
    value: Float!
}

type Sensor @key(fields: "externalID") {
    externalID: String!
    sourcePath: String!
    unit: String!
    values(startTime: Time!, endTime: Time): [Value!]!
}

type Query {
    sensors(ids: [String!]): [Sensor!]!
}
```

Listing 4.2: The GraphQL schema for the Building Management System.

Although the BMS system already includes spatial metadata such as area, building, floor, and room, this information is not currently exported into the database. Therefore, each sensor is manually connected to a corresponding `Room` in the FMS schema, which is defined in Listing 4.1.

This integration is implemented by extending the schema from Listing 4.2 with additional code, shown below in Listing 4.3.

```
extend type Room @key(fields: "id") {
    id: ID! @external
    name: String! @external
    sensors: [Sensor!]! @requires(fields: "name")
}
```

Listing 4.3: The extra code in the GraphQL schema to integrate it into the Federation.

This code declares the `Room` type as external and identifies it using the `id` field as a key. It then extends the `Room` type by adding a new field, `sensors`, which lists all associated sensors. The resolver for `sensors` requires the `name` field (also marked as external) to function correctly.

4.1.4 Coffee Data

At AAU Innovate, beverage consumption is also monitored through a semi-automated data collection process, as described in Section 3.4. As shown in Figure 3.4, data is still manually exported to a USB key, after which a Python script processes the data and stores it in both CSV files and a database. The CSV export functionality was developed during my student job at AAU Innovate, while the integration of a PostgreSQL database was implemented specifically for this project.

Since data is collected monthly, the dataset grows over time. Relying solely on CSV files and reading them into memory becomes inefficient and problematic as the volume increases. Additionally, using only CSV files would require manually updating the server with new files each month and either restarting the server or implementing logic to detect changes. This approach was considered inadequate, which led to the implementation of a PostgreSQL database.

The database schema is defined using SQL, as shown in Listing 4.4. It consists of a `machines` table containing general information about each coffee machine. There

are also two tables for beverage data: `beverage_counts` and `beverage_details`. The `beverage_counts` table stores total beverage counts, while `beverage_details` records the number of each specific beverage consumed.

The `beverage_counts` table was introduced because aggregated data is queried much more frequently than detailed beverage data. Without it, calculating totals would require repeated aggregation of the `beverage_details` table. Although it would be theoretically possible to use separate tables for each drink type, this would complicate the schema, slow down queries due to the need for frequent joins, and offer only marginal storage benefits, as each entry would still require a joinable identifier.

```
CREATE TABLE machines (
    machine_id VARCHAR(20) PRIMARY KEY,
    machine_name VARCHAR(100) NOT NULL,
    floor_id VARCHAR(30) NOT NULL
);

CREATE TABLE beverage_counts (
    id SERIAL PRIMARY KEY,
    machine_id VARCHAR(20) REFERENCES machines(machine_id),
    total_beverages INTEGER NOT NULL,
    timestamp TIMESTAMP NOT NULL,
    UNIQUE(machine_id, timestamp)
);

CREATE TABLE beverage_details (
    id SERIAL PRIMARY KEY,
    machine_id VARCHAR(20) REFERENCES machines(machine_id),
    beverage_name VARCHAR(50) NOT NULL,
    count INTEGER NOT NULL,
    timestamp TIMESTAMP NOT NULL
);
```

Listing 4.4: The SQL setup for the coffee data database

Based on this structure, a corresponding GraphQL schema was created for the coffee system, shown in Listing 4.5. The schema closely mirrors the database structure. Data can be queried through the `machines` query, which accepts an optional `machineIDs` argument—an array of IDs used to filter the results. If this argument is omitted or left empty, data for all machines is returned.

For each machine, beverage data can be requested via the `beverageCounts` and `beverageDetails` fields, both of which require a `startTime` and accept an optional `endTime`. If `endTime` is not provided, it defaults to the current time.

```
type Machine @key(fields: "id") {
    id: ID!
    name: String!
    beverageCounts(startTime: Time, endTime: Time): [BeverageCount!]!
    beverageDetails(startTime: Time, endTime: Time): [BeverageDetail!]!
}

type BeverageCount {
    id: ID!
    totalBeverages: Int!
    timestamp: Time!
}

type BeverageDetail {
    id: ID!
    beverageName: String!
    count: Int!
    timestamp: Time!
}

type Query {
    machines(machineIDs: [ID!]): [Machine!]!
}
```

Listing 4.5: The GraphQL schema for the coffee data system.

Finally, the coffee data was integrated into the federated system, as shown in Listing 4.6. Since coffee machines are typically located in hallways rather than specific rooms, they are associated with a `Floor` entity rather than a `Room`. The `machines` field is defined as an extension on the external `Floor` type from the FMS system graph.

```
extend type Floor @key(fields: "id") {
    id: ID! @external
    machines: [Machine!]!
}
```

Listing 4.6: The GraphQL code to implement the coffee data into the federation.

4.1.5 Door Counter System

The door counter system is initially described in Section 3.5. It was set up by an external company, and their configuration is illustrated in Figure 3.6. The collected data is stored in ThingsBoard, an open-source IoT platform [40], which is hosted by Multiteknik, the company behind the system, at *http://iot.multiteknik.dk:8080* .

ThingsBoard offers a REST API that can be leveraged to access the data. To authenticate, a `POST` request must be sent to the */api/auth/login* endpoint with a payload containing the `username` and `password`.[1] Upon successful login, an access

[1]The author is aware that this request is transmitted over HTTP, which exposes the request payload to interception during network transit. A request was made to Multiteknik to upgrade to HTTPS while outlining the potential security risks; however, this request was denied. Until a new system is in place, the credentials used are both unique and rotated regularly. In the future, AAU Innovate plans to replace this system with one that offers more detailed data, such as direction of movement, which could allow real-time estimations of building occupancy instead of only daily counts.

token is returned, along with a renewal token. The system authenticates once on server start and uses the renewal token to refresh the access token before it expires.

The telemetry data itself is accessed via a `GET` request to the */api/plugins/telemetry/DEVICE/[DEVICE_ID]/values/timeseries* endpoint. The request must include query parameters such as `keys`, `startTs`, `endTs`, and `limit`. The API returns a JSON object structured as shown in Listing 4.7. Notably, the `ts` field represents the timestamp in Unix milliseconds.

```
{
  "c1": [
    {
      "ts": number,
      "value": "string"
    },
    {
      "ts": number,
      "value": "string"
    }
    // ... more objects
  ],
  "c2": [
    {
      "ts": number,
      "value": "string"
    }
    // ... more objects
  ],
  "c3": [
    {
      "ts": number,
      "value": "string"
    }
    // ... more objects
}
```

Listing 4.7: The structure of the response of the door counter system REST API.

This data access is encapsulated in the GraphQL schema shown in Listing 4.8. The schema defines two types: `Entrance` and `TelemetryData`. Data can be requested using the `getEntrances` query, optionally filtered by specific entrance IDs. Each `Entrance` can return telemetry data for a given time range using the `telemetryData` field.

```graphql
type Entrance @key(fields: "id") {
    id: ID!
    name: String!
    telemetryData(startTime: Time!, endTime: Time): [TelemetryData!]!
}

type TelemetryData {
    timestamp: String!
    value: Int
}

type Query {
    getEntrances(ids: [ID!]): [Entrance!]!
}
```

Listing 4.8: The GraphQL schema of the door counter system.

To integrate the door counter system into the federation, the `Building` type is extended with an `entrances` field, as shown in Listing 4.9. While entrances could also be associated with a specific floor, this would often default to the ground floor, leaving other floors with empty entrance data. This could create confusion, for example, interpreting staircases or elevators as entrances. Therefore, associating entrances at the building level provides a simpler and more intuitive structure.

```graphql
extend type Building @key(fields: "id") {
    id: ID! @external
    entrances: [Entrance!]!
}
```

Listing 4.9: The GraphQL code to integrate the door counter system into the federation.

4.2 The Supergraph

All the individual data sources described in Section 4.1 are combined using Apollo Federation (see Section 1.3.2). This results in the formation of a unified supergraph, which aggregates all previously defined subgraphs. A schematic overview of this supergraph is shown in Figure 4.1. From the diagram, it is evident that the FMS system serves as the backbone of the entire architecture. This makes logical sense, as it anchors each system through its spatial positioning in the building. Consequently, the system forms a truly integrated API, now enriched with spatial context.

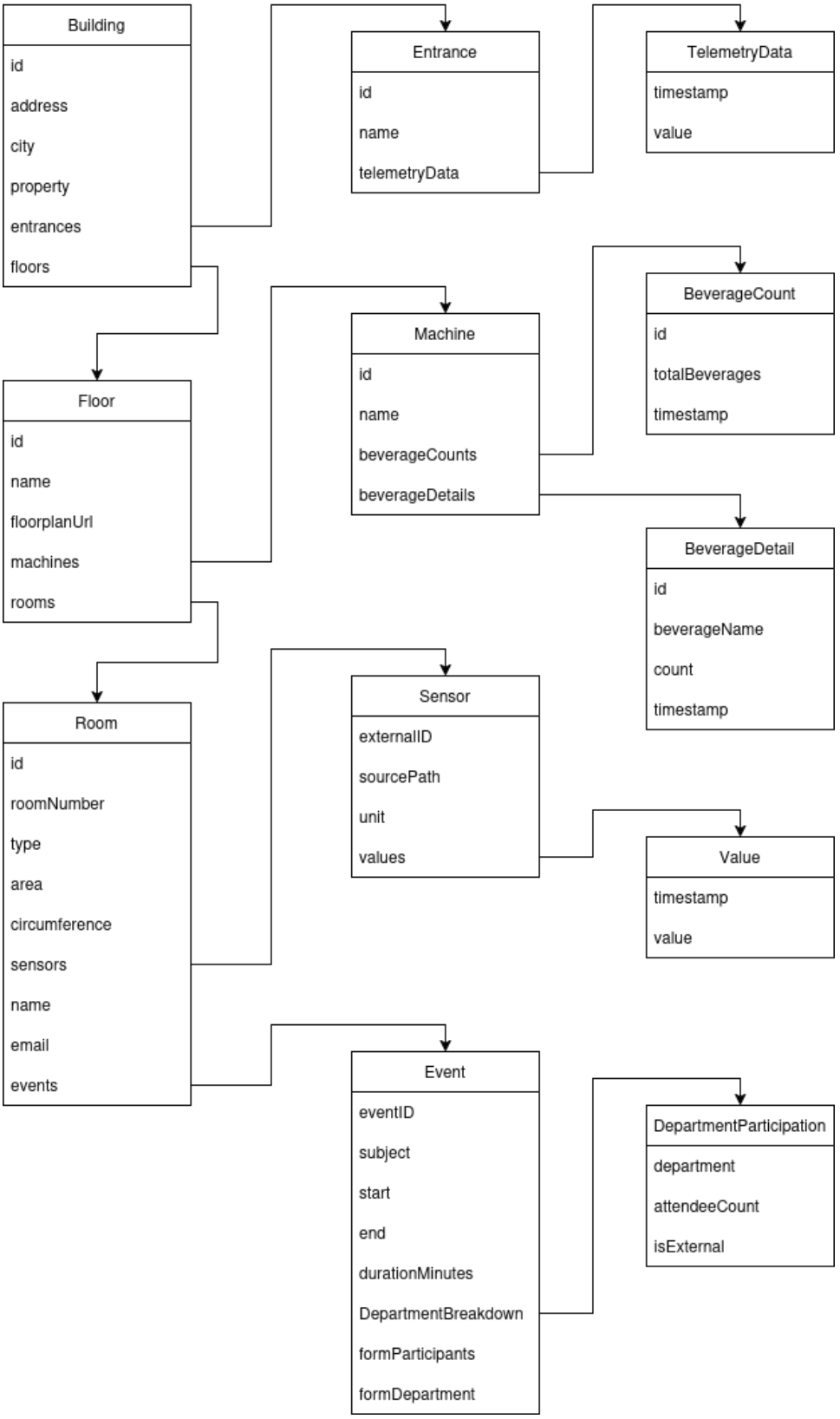


Figure 4.1: The integrated API schema of the GraphQL Federation.

To implement this in practice, a NodeJS gateway server is created using the `apollo-server` and `@apollo/gateway` packages. These packages are not available in GoLang, which is why NodeJS is used for this component.

The source code required to create this server is straightforward and is shown in full in Listing 4.10. Each subgraph is defined with a name and a URL inside the `IntrospectAndCompose` configuration of the `ApolloGateway` instance. The federated server is then instantiated using the `ApolloServer` class.

```
const { ApolloServer } = require("apollo-server");
const { ApolloGateway, IntrospectAndCompose } = require("@apollo/gateway");

const gateway = new ApolloGateway({
  supergraphSdl: new IntrospectAndCompose({
    subgraphs: [
      { name: "doorcounters", url: "http://doorcounters:4001/query" },
      { name: "bms", url: "http://bms:4002/query" },
      { name: "fms", url: "http://fms:4003/query" },
      { name: "outlook", url: "http://outlook:4004/query" },
      { name: "coffee", url: "http://coffee:4005/query" },
    ],
  }),
});

const server = new ApolloServer({
  gateway,
  subscriptions: false,
});

server.listen().then(({ url }) => {
  console.log(`🚀 Server ready at ${url}`);
});
```

Listing 4.10: The source code to create the gateway for the Apollo Federation.

To deploy the entire system efficiently, Docker and Docker Compose are employed. The benefits and setup of this approach are discussed in detail in Section 1.3.4. Each subgraph service runs in its own container. Since the GoLang-based subgraphs follow a similar hosting structure, a single reusable Dockerfile is used across them. The gateway uses its own Dockerfile tailored to NodeJS. Docker Compose is used to orchestrate all these containers. A full overview of this deployment setup is provided in Appendix B.

References

[1] K. Lawal and H. N. Rafsanjani, "Trends, benefits, risks, and challenges of IoT implementation in residential and commercial buildings," *Energy and Built Environment*, vol. 3, no. 3, pp. 251–266, 2022.

[2] J. Aguilar, A. Garcès-Jimènez, N. Gallego-Salvador, J. A. G. De Mesa, J. M. Gomez-Pulido, and À. J. Garc\ia-Tejedor, "Autonomic management architecture for multi-HVAC systems in smart buildings," *IEEE Access*, vol. 7, pp. 123402–123415, 2019.

[3] B. Dong, V. Prakash, F. Feng, and Z. O'Neill, "A review of smart building sensing system for better indoor environment control," *Energy and Buildings*, vol. 199, pp. 29–46, 2019.

[4] S. D'Oca, T. Hong, and J. Langevin, "The human dimensions of energy use in buildings: A review," *Renewable and Sustainable Energy Reviews*, vol. 81, pp. 731–742, 2018.

[5] F. Wang *et al.*, "Predictive control of indoor environment using occupant number detected by video data and CO2 concentration," *Energy and Buildings*, vol. 145, pp. 155–162, 2017.

[6] T. Leephakpreeda, "Adaptive occupancy-based lighting control via grey prediction," *Building and environment*, vol. 40, no. 7, pp. 881–886, 2005.

[7] N. Maynard and D. Grant, "Smart Buildings & the Battle for Sustainability," Basingstoke, Hampshire, UK, Mar. 2022. [Online]. Available: http://www.juniperresearch.com/

[8] Density, "Unused Space in Buildings Generates 22M Tons of CO\textsubscript{2}." [Online]. Available: https://www.density.io/resources/how-companies-can-take-climate-action-with-occupancy-data

[9] R. K. Garrett, "Should We Worry That Half of Americans Trust Their Gut to Tell Them What's True?." [Online]. Available: https://theconversation.com/should-we-worry-that-half-of-americans-trust-their-gut-to-tell-them-whats-true-84259

[10] T. Stobierski, "Data-Driven Decision-Making." [Online]. Available: https://online.hbs.edu/blog/post/data-driven-decision-making

[11] K. Bäcklund, P. Lundqvist, and M. Molinari, "Showcasing a digital twin for higher educational buildings: developing the concept toward human centricity," *Frontiers in Built Environment*, vol. 10, p. 1347451, 2024.

[12] Z. Ni, C. Zhang, M. Karlsson, and S. Gong, "Leveraging Deep Learning and Digital Twins to Improve Energy Performance of Buildings," in *2023 IEEE 3rd International Conference on Industrial Electronics for Sustain-*

able Energy Systems (IESES), IEEE, Jul. 2023, pp. 1–6. doi: 10.1109/ieses53571.2023.10253721.

[13] S. Anik, X. Gao, N. Meng, P. Agee, and A. McCoy, "A Cost-Effective, Scalable, and Portable IoT Data Infrastructure for Indoor Environment Sensing." [Online]. Available: https://arxiv.org/abs/2110.14042

[14] D. Hugo *et al.*, "A smart building semantic platform to enable data re-use in energy analytics applications: the Data Clearing House." [Online]. Available: https://arxiv.org/abs/2311.11630

[15] A. Floris, S. Porcu, R. Girau, and L. Atzori, "An iot-based smart building solution for indoor environment management and occupants prediction," *Energies*, vol. 14, no. 10, p. 2959, 2021.

[16] Z. Chen, C. Jiang, and L. Xie, "Building occupancy estimation and detection: A review," *Energy and Buildings*, vol. 169, pp. 260–270, 2018, doi: https://doi.org/10.1016/j.enbuild.2018.03.084.

[17] S. S. Kumar, R. Chandra, and S. Agarwal, "A Real-Time Approach for Smart Building Operations Prediction Using Rule-Based Complex Event Processing and SPARQL Query." [Online]. Available: https://arxiv.org/abs/2309.10822

[18] K. H. Andersen *et al.*, "Exploring occupant detection model generalizability for residential buildings using supervised learning with IEQ sensors," *Building and Environment*, vol. 254, Apr. 2024, doi: 10.1016/j.buildenv.2024.111319.

[19] I. P. Mohottige, H. H. Gharakheili, T. Moors, and V. Sivaraman, "Modeling Classroom Occupancy Using Data of WiFi Infrastructure in a University Campus," *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9981–9996, May 2022, doi: 10.1109/jsen.2022.3165138.

[20] V. Bellavista-Parent, J. Torres-Sospedra, and A. Perez-Navarro, "New trends in indoor positioning based on WiFi and machine learning: A systematic review," in *2021 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, Nov. 2021, pp. 1–8. doi: 10.1109/ipin51156.2021.9662521.

[21] Y. Wang and L. Shao, "Understanding occupancy and user behaviour through Wi-Fi-based indoor positioning," *Building Research & Information*, vol. 46, no. 7, pp. 725–737, 2018.

[22] GraphQL Foundation, "GraphQL: A query language for your API." Accessed: May 02, 2025. [Online]. Available: https://graphql.org/

[23] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," 2000. Accessed: May 02, 2025. [Online]. Available: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[24] G. Bello, "GraphQL vs. REST." Accessed: May 02, 2025. [Online]. Available: https://blog.postman.com/graphql-vs-rest/

[25] "Federation." Accessed: May 02, 2025. [Online]. Available: https://www.apollographql.com/docs/graphos/schema-design/federated-schemas/federation

[26] The gqlgen team, "gqlgen." Accessed: May 02, 2025. [Online]. Available: https://gqlgen.com/

[27] S. Khayalian, "Docker vs. Docker Compose: Simple and Fun Explanation." Accessed: May 02, 2025. [Online]. Available: https://medium.com/@ShantKhayalian/docker-vs-docker-compose-simple-and-fun-explanation-4811582127f7

[28] A. Innovation, "AAU Innovation - Data." Accessed: Oct. 05, 2023. [Online]. Available: https://www.touch.innovate.aau.dk/aau-innovation-data

[29] FM:Systems, "Top Office Space Trends for Hybrid Work Success." Accessed: Apr. 02, 2025. [Online]. Available: https://fmsystems.com/blog/top-office-space-trends-hybrid-work-success/

[30] Microsoft Learn, "Create and manage room mailboxes in Exchange Server." Accessed: Jul. 03, 2025. [Online]. Available: https://learn.microsoft.com/en-us/exchange/recipients/room-mailboxes?view=exchserver-2019

[31] Microsoft Corporation, "List events - Microsoft Graph v1.0." Accessed: Apr. 02, 2025. [Online]. Available: https://learn.microsoft.com/en-us/graph/api/group-list-events?view=graph-rest-1.0&tabs=http

[32] Schneider Electric, "EcoStruxure Building Operation." Accessed: Apr. 03, 2025. [Online]. Available: https://www.se.com/uk/en/product-range/62111-ecostruxure-building-operation/?parent-subcategory-id=26426255&filter=business-2-building-automation-and-control

[33] S. P. Melgaard *et al.*, "Fault Detection in AHU: A Walkthrough for Implementation in a Danish Educational Building." 2024.

[34] N&W GLOBAL VENDING GROUP, "OPERATOR MANUAL Wittenborg 9100 ES 2FB." Via Roma 24 - 24030 Valbrembo (BG) Italy, 2017. Accessed: Oct. 27, 2023. [Online]. Available: https://f.hubspotusercontent20.net/hubfs/6668578/Manuals/Wittenborg9100-3_Manual.pdf

[35] Multiteknik, "Kundetæller – kom i gang på under en time!." Accessed: Apr. 03, 2025. [Online]. Available: https://www.multiteknik.dk/produkter/kundetaeller-til-butikker/

[36] M. Vega-Barbas, M. Álvarez-Campana, D. Rivera, M. Sanz, and J. Berrocal, "AFOROS: A Low-Cost Wi-Fi-Based Monitoring System for Estimating Occupancy of Public Spaces," *Sensors*, vol. 21, no. 11, p. 3863, Jun. 2021, doi: 10.3390/s21113863.

[37] S. Liu, Y. Zhao, F. Xue, B. Chen, and X. Chen, "DeepCount: Crowd counting with WiFi via deep learning," *arXiv preprint arXiv:1903.05316*, 2019.

[38] J. Dai, M. Wang, B. Wu, J. Shen, and X. Wang, "A Survey of Latest Wi-Fi Assisted Indoor Positioning on Different Principles," *Sensors*, vol. 23, no. 18, 2023, doi: 10.3390/s23187961.

[39] A. Le Floch, R. Kacimi, P. Druart, Y. Lefebvre, and A.-L. Beylot, "A comprehensive framework for 5G indoor localization," *Computer Communications*, vol. 228, p. 107968, 2024, doi: https://doi.org/10.1016/j.comcom.2024.107968.

[40] "ThingsBoard." Accessed: May 02, 2025. [Online]. Available: https://thingsboard.io/

Appendix A

AAU Innovate's Layout

This section details the layout of the AAU Innovate building, which serves as the case study for this report. The building comprises five levels, ranging from the basement to the third floor. The detailed floor plans for each level are illustrated in the following figures: Figure A.2, A.3, A.4, A.5 and A.6 To provide context for the floor plans, Figure A.1 shows the exterior view of the AAU Innovate building.

Figure A.1: The AAU Innovate building viewed from the outside.

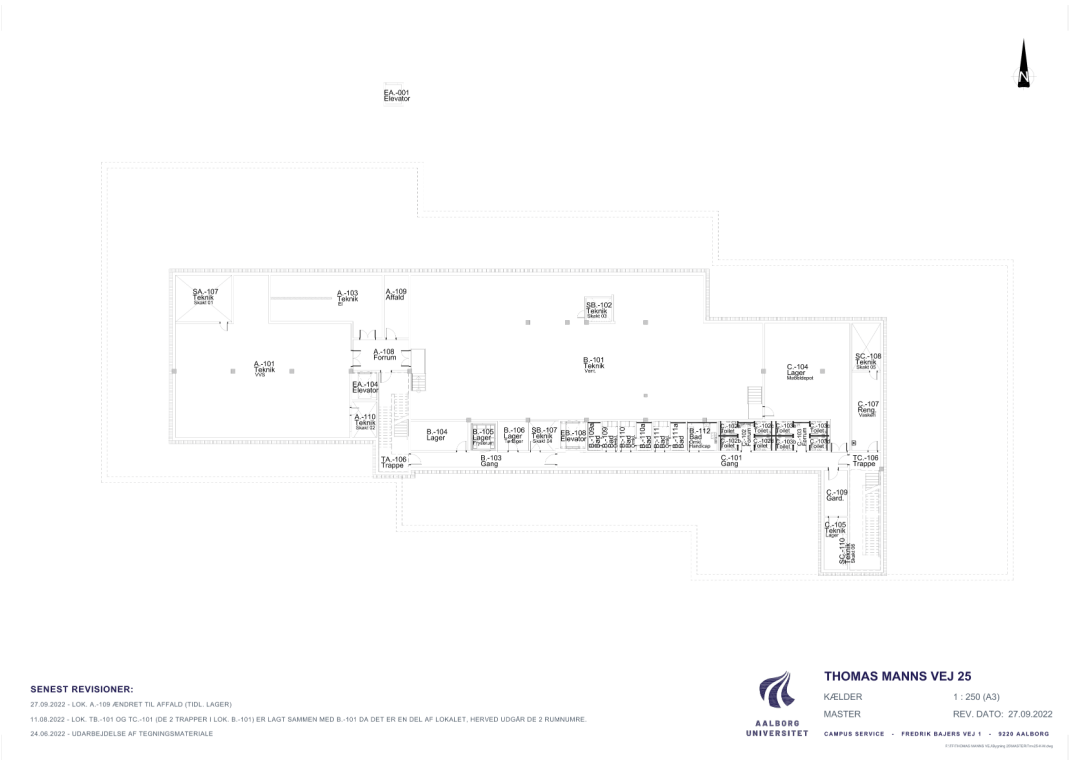


Figure A.2: The floorplan of the basement floor of the AAU Innovate building.

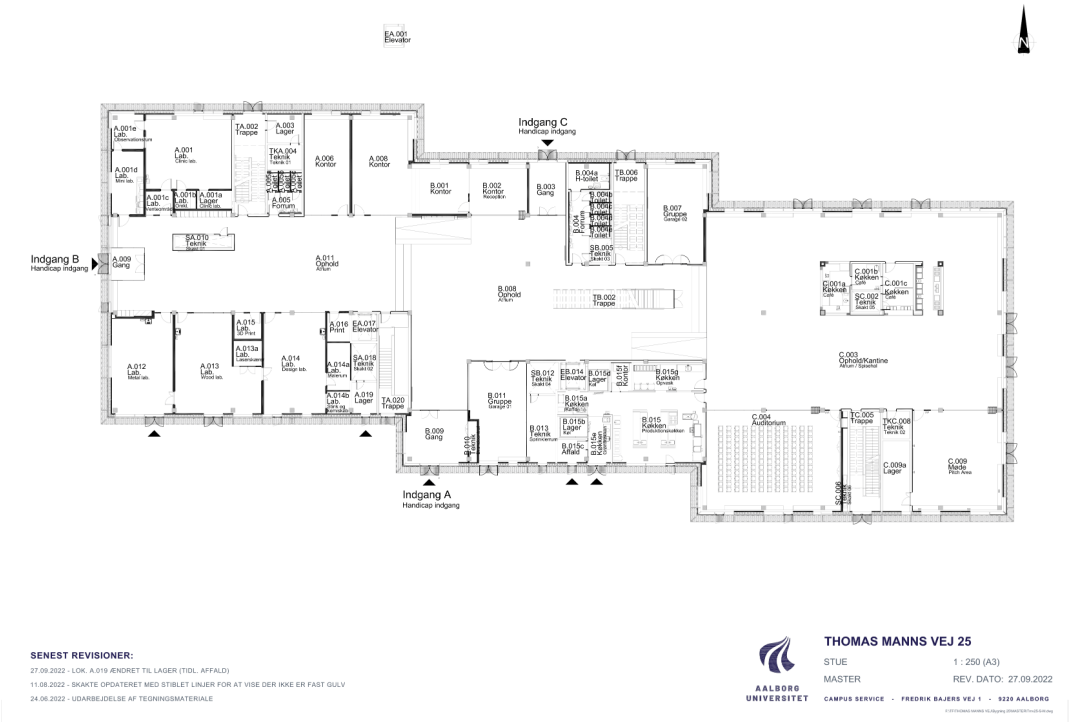
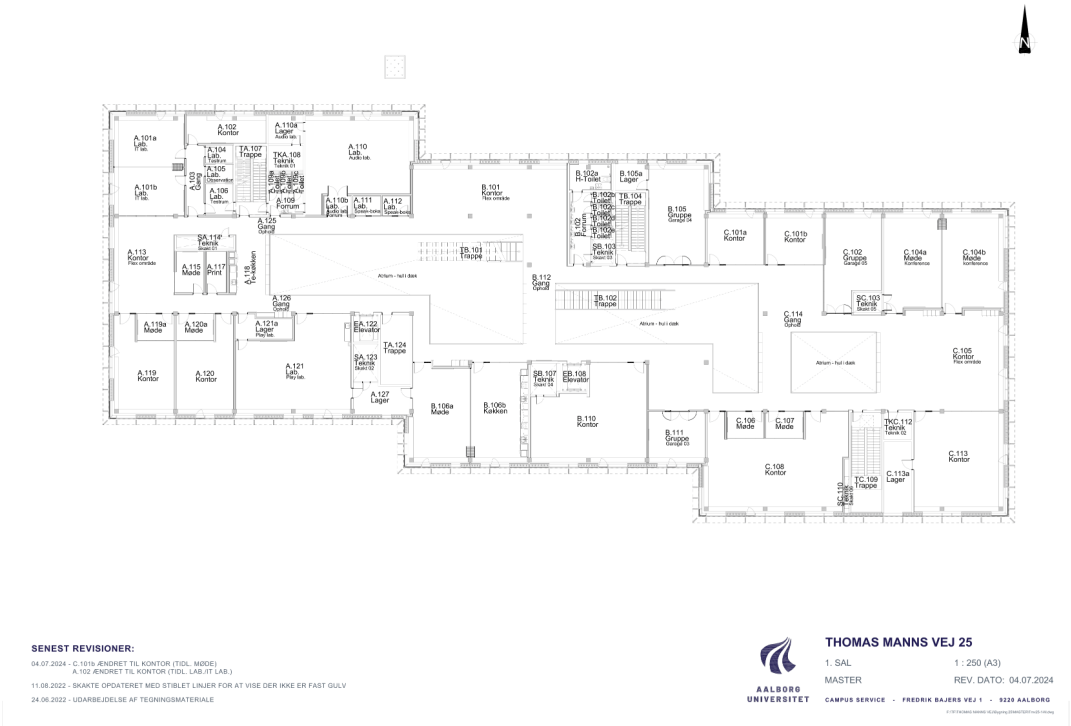


Figure A.3: The floorplan of the ground floor of the AAU Innovate building.



SENEST REVISIONER:

04.07.2024 - C.101b ÆNDRET TIL KONTOR (TIDL. MØDE)
A.102 ÆNDRET TIL KONTOR (TIDL. LAB./IT LAB.)

11.08.2022 - SKAKTE OPDATERET MED STIBLET LINJER FOR AT VISE DER IKKE ER FAST GULV

24.06.2022 - UDARBEJDELSE AF TEGNINGSMATERIALE

AALBORG UNIVERSITET

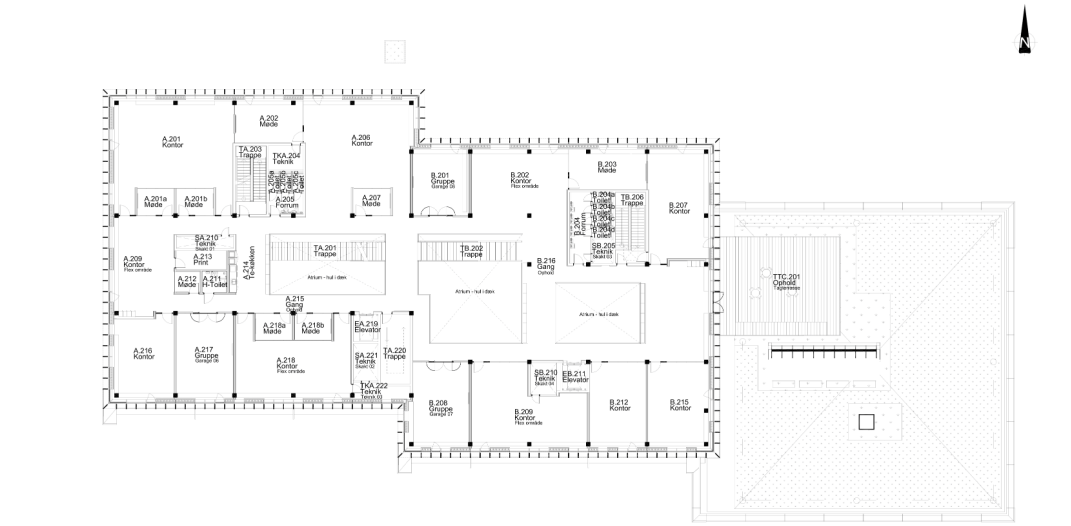
THOMAS MANNS VEJ 25

1. SAL | 1 : 250 (A3)

MASTER | REV. DATO: 04.07.2024

CAMPUS SERVICE - FREDRIK BAJERS VEJ 1 - 9220 AALBORG

Figure A.4: The floorplan of the first floor of the AAU Innovate building.



SENEST REVISIONER:

19.12.2024 - RUMNUMMERET TIL TRAPPE TB.202 (TIDL. TB.102) ER OPDATERET

04.07.2024 - OMBYGNING VED B.209, HVOR DER ER OPSAT GLASVÆG OG DØR MV.

11.08.2022 - SKAKTE OPDATERET MED STIBLET LINJER FOR AT VISE DER IKKE ER FAST GULV

24.06.2022 - UDARBEJDELSE AF TEGNINGSMATERIALE

AALBORG UNIVERSITET

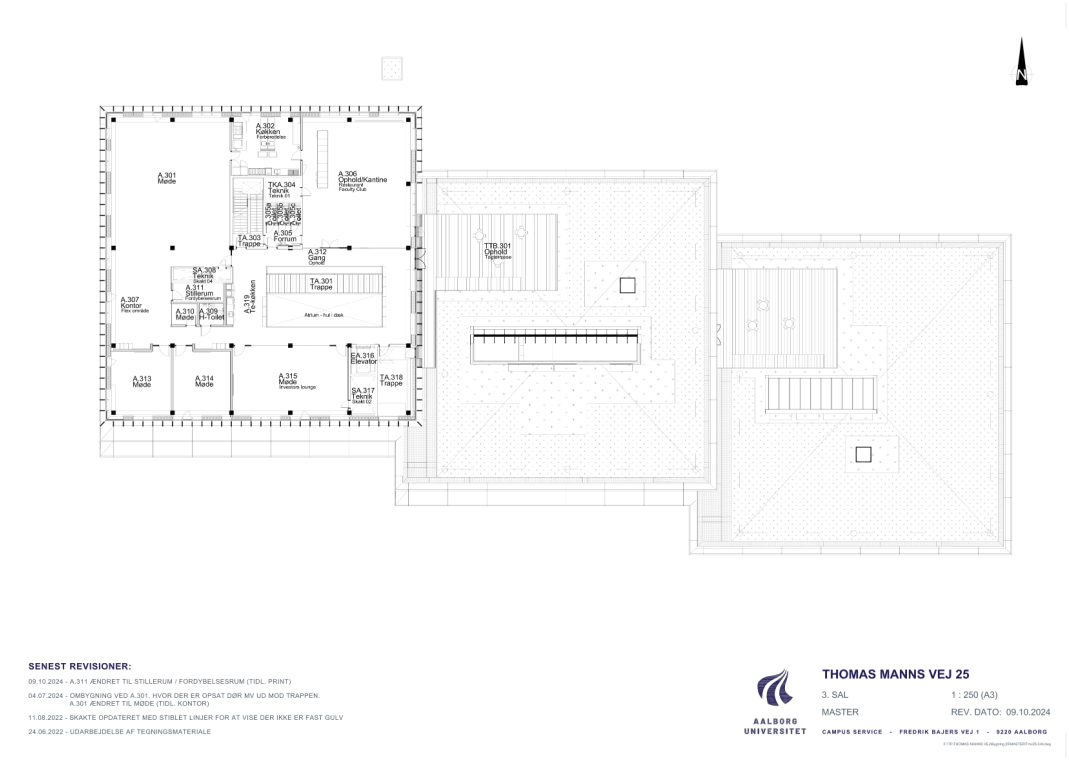
THOMAS MANNS VEJ 25

2. SAL | 1 : 250 (A3)

MASTER | REV. DATO: 19.12.2024

CAMPUS SERVICE - FREDRIK BAJERS VEJ 1 - 9220 AALBORG

Figure A.5: The floorplan of the second floor of the AAU Innovate building.



SENEST REVISIONER:

09.10.2024 - A.311 ÆNDRET TIL STILLERUM / FORDYBELSESRUM (TIDL. PRINT)

04.07.2024 - OMBYGNING VED A.301, HVOR DER ER OPSAT DØR MV UD MOD TRAPPEN.
A.301 ÆNDRET TIL MØDE (TIDL. KONTOR)

11.08.2022 - SKAKTE OPDATERET MED STIBLET LINJER FOR AT VISE DER IKKE ER FAST GULV

24.06.2022 - UDARBEJDELSE AF TEGNINGSMATERIALE

AALBORG UNIVERSITET

THOMAS MANNS VEJ 25

3. SAL 1 : 250 (A3)

MASTER REV. DATO: 09.10.2024

CAMPUS SERVICE - FREDRIK BAJERS VEJ 1 - 9220 AALBORG

Figure A.6: The floorplan of the third floor of the AAU Innovate building.

Appendix B

Hosting the application

The application is hosted using Docker containers, orchestrated through Docker Compose to manage inter-container networking and simplify service configuration. This setup is illustrated in Listing B.1.

```
# Stage 1: Builder
FROM golang:1.24-alpine AS builder

# Define an argument for the application source directory relative to the
context
ARG APP_SRC=.

WORKDIR /app

# Copy only necessary files first for layer caching
COPY ${APP_SRC}/go.mod ${APP_SRC}/go.sum ./
RUN go mod download

# Copy the specific application source code using the build argument
COPY ${APP_SRC} ./

# Build the application binary, ensure static linking
RUN CGO_ENABLED=0 GOOS=linux go build -ldflags="-w -s" -o /app-binary .

# Stage 2: Runner
FROM alpine:latest

# Create a non-root user for security
RUN adduser -D -u 10001 appuser
WORKDIR /app

# Copy the compiled binary from the builder stage
COPY --from=builder /app-binary .
# Change ownership if using a non-root user
RUN chown appuser:appuser /app/app-binary

# Switch to the non-root user
USER appuser
```

Listing B.1: The Dockerfile used by all the GoLang individual graphs

Each Go-based microservice is built using a multi-stage Dockerfile. The first stage uses the `golang` image to compile the source code into a statically linked binary, optimizing for size and performance. In the second stage, the compiled binary is

transferred to a lightweight `alpine` image. For enhanced security, a non-root user is created and assigned ownership of the binary. This practice reduces the attack surface in production environments.

The gateway, a Node.js application, uses a straightforward setup based on the official Node.js image. It installs dependencies via `npm ci` for reproducible builds, and then copies over the core application file. This configuration is shown in Listing B.2.

```
# Use a specific, minimal Node.js image
FROM node:22-alpine

# Set the working directory inside the container
WORKDIR /app

# Copy package.json and package-lock.json (or yarn.lock, pnpm-lock.yaml)
first
# This allows Docker to cache this layer, speeding up builds if only code
changes
COPY package*.json ./

# Install dependencies
RUN npm ci --production

# Copy the rest of the application code
COPY ./server.js ./server.js
```

Listing B.2: The Dockerfile for the NodeJS gateway.

To coordinate all services, a `docker-compose.yaml` file is used. This file defines how containers should be built and configured, including port mappings, environment variables, and command overrides. It also specifies volumes and context paths for individual services. A portion of this configuration is shown in Listing B.3. Running `docker compose up` brings up the entire system, whether locally or on a remote server, thanks to Docker's platform-independent nature.

```yaml
services:
  gateway:
    build:
      context: ./service-gateway
    ports:
      - "4000:4000"
    environment:
      - APP_LISTEN_PORT=4000
    command: ["node", "server.js"]

  doorcounters:
    build:
      context: .
      args:
        APP_SRC: ./service-doorcounters
    ports:
      - "4001:4001"
    environment:
      - APP_LISTEN_PORT=4001
    env_file:
      - ./service-doorcounters/.env
    command: ["./app-binary"]

...All other subgraph systems

  coffee:
    build:
      context: .
      args:
        APP_SRC: ./service-coffee
    ports:
      - "4005:4005"
    environment:
      - APP_LISTEN_PORT=4005
    env_file:
      - ./service-coffee/.env
    command: ["./app-binary"]
```

Listing B.3: The Docker Compose file defines the interaction between all the containers.