CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Network-Driven Approach to Theft Detection in Shared Computing Environments

Bachelor thesis in Computer Science and Software Engineering

William Kangas
Hassan Khan
Amanda Lindell
Jonas Nemeth
Emelie Quach
Melisa Zanjani

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Bachelor of Science Thesis in Software Engineering
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Graded by teacher: *Thommy Eriksson*
Examinator: *Patrik Jansson*
Supervisor: *Henrik Jansson Valter*
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Department of Computer Science and Engineering
Gothenburg, 2025

Abstract

In shared computing environments such as university computer labs, the risk of hardware theft is often overlooked by traditional cybersecurity measures. This project proposes and evaluates a lightweight, network-driven system for detecting potential hardware theft in real time. The solution relies on a "heartbeat" mechanism where each client device sends regular check-in signals to a central server. If a device stops reporting, the system flags it as potentially compromised.

A prototype was implemented and tested under controlled conditions, where it reliably detected critical hardware removal events. The system proved effective at generating timely alerts and minimising false positives during normal reboots. A web-based interface was developed, enabling administrators to monitor devices efficiently.

The findings highlight both the potential and the limitations of network-based monitoring in theft prevention and suggest future enhancements such as integration with maintenance schedules and machine learning. Overall, the project demonstrates that a simple, low-overhead network-based solution can offer substantial improvements in theft detection in shared-use IT environments.

Sammanfattning

I delade datorarbetsmiljöer är risken för hårdvarustölder något som ofta hamnar i skymundan för traditionella IT-säkerhetslösningar. Detta projekt har utvecklat och utvärderat ett lättviktigt, nätverksbaserat system för att i realtid upptäcka misstänkt stöld av datorutrustning. Lösningen bygger på ett så kallat "heartbeat"-system där varje enhet regelbundet skickar signaler till en central server. Om en enhet slutar rapportera så flaggas den som potentiellt utsatt för stöld.

En prototyp togs fram och testades i en kontrollerad miljö. Resultaten visade att systemet kunde identifiera borttagning av kritiska komponenter. Systemet kunde samtidigt undvika falsklarm vid normala omstarter. För att underlätta övervakning utvecklades ett användarvänligt och effektivt webbgränssnitt.

Sammantaget visar projektet att övervakningssystemet kan ge ett värdefullt skydd mot hårdvarustölder, särskilt om det kombineras med underhållsscheman och maskininlärning.

Contents

1 Introduction

This chapter introduces the motivation for the project, the specific problem addressed, and the broader relevance of the proposed solution. It begins by outlining the background of hardware theft in shared computing environments, followed by a detailed problem statement, the project's objectives, and an examination of its wider impact and ethical considerations.

1.1 Background and Motivation

Hardware theft in academic institutions, particularly in open-access computer labs, is a growing concern. Despite this risk, traditional cybersecurity measures primarily focus on mitigating digital threats such as malware and unauthorised access, while the physical security of IT assets remains an often-overlooked area [1]. Most current systems prioritise whole-device tracking (e.g., GPS or login-based systems) and network availability, but fail to protect against the theft of internal hardware components [2] [3]. As a result, these systems remain inadequate in safeguarding valuable internal components from theft.

Chalmers University of Technology exemplifies this challenge, with its open-access computer labs offering students round-the-clock access. This accessibility, while essential for academic flexibility, also increases the risk of hardware theft. These shortcomings highlight the need for monitoring solutions that can detect the theft of individual components in real-time, ensuring better protection for shared computing environments.

1.2 Problem Statement

At Chalmers University of Technology, the previously used monitoring system, *InterMapper* (a network monitoring software used for visualising network infrastructure), is unable to detect hardware removal in real-time [4]. Although it can identify when a machine disconnects from the network, it cannot distinguish between routine disconnections for system maintenance or authorised actions and unauthorised hardware removal. Unexpected disconnections, often dismissed as network instability, can actually signal an early indicator of hardware removal in shared computing environments.

Intermapper relied on "pinging" the computers to check their status. According to Chalmers network specialist Per Olsson, if no response was received within five minutes, an alert email would be sent indicating that the device was no longer active. Pinging-based monitoring proved to be unreliable; for instance, a computer could appear offline due to an accidentally unplugged Ethernet cable or disturbances during routine cleaning, resulting in false alarms. The discontinuation of Intermapper highlights the need for a more reliable and user-centred monitoring solution that reduces cognitive effort and provides IT personnel with clear, actionable feedback.

Without a system capable of differentiating between routine maintenance, temporary network issues, and unauthorised removals, institutions are also vulnerable to undetected

component theft. As highlighted by Vidaković and Vinko, hardware-based methods for electronic device protection are essential in preventing unauthorised tampering and ensuring system integrity[5]. Their findings underscore the importance of integrated physical and digital security measures, especially in environments where multiple users access shared resources.

The importance of protecting critical components, such as random-access memory (RAM), graphical processing units (GPUs), and storage drives, is often underestimated. These components are sufficiently compact to be discreetly removed, may contain valuable data, and their removal can render the device inoperable, disrupting workflows and academic activities. The failure of *InterMapper* to detect hardware removal in real-time presents a direct threat to both the physical security of IT assets and the continuity of academic operations, as it may prevent timely alerts for potential system failures during critical periods like lectures and exams [6].

This gap in monitoring capability highlights the need for a solution that can distinguish between authorised and unauthorised events in real-time, without relying on heavy infrastructure or assumptions about user behaviour. This prompted the following question, which guided the project:

Can a lightweight, network-based system reliably detect hardware theft in real-time within a shared academic computing environment?

1.3 Purpose and Objectives

The purpose of this project is to develop an automated monitoring system designed to enhance the security of shared computing environments, such as the computer labs at Chalmers University of Technology. The system will focus on detecting irregularities in real-time, with particular emphasis on identifying hardware removal and unexpected disconnections. To evaluate the success of the system, the following measurable objectives were defined:

- **To** detect when hardware components, such as random-access memory (RAM), graphical processing units (GPUs), or storage drives, are removed from the system. *The system is considered to meet this goal if it can reliably trigger alerts upon removal events during controlled tests.*
- **To** identify when a device unexpectedly disconnects from the network. *This goal is fulfilled if the system successfully detects at least 80% of tested disconnection events within the configured timeout threshold.*
- **To** provide immediate alerts to security personnel or IT administrators upon detection of potential theft. *Success is defined as alerts being generated and delivered within a defined time threshold of detection in test environments.*
- **To** minimise false alarms by accurately distinguishing between authorised actions, such as scheduled maintenance, rebooting systems, and unauthorised hardware removal (a goal for future development). *This goal is considered ongoing and is in-*

tended for future iterations, with success evaluated by reducing false positives in real-world deployments.

These objectives aim to create a proactive monitoring solution that not only detects theft but also helps prevent it before it disrupts academic operations, ensuring more reliable protection for shared IT infrastructure.

1.4 Relevance and Broader Impact

While this project developed a prototype meant for Chalmers, the system addresses a common challenge in shared IT environments: detecting hardware theft without costly infrastructure or surveillance. The prototypes lightweight and platform-independent design makes it applicable in other environments as well - such as schools, offices, and libraries. The broader societal and environmental implications - such as reducing equipment loss and extending hardware lifespan - are further explored in Section 7.5.

1.5 Ethical and Social Aspects

The proposed monitoring system enhances security in shared computing environments by detecting potential hardware theft at an early stage. This capability can reduce financial losses, ensure the availability of IT resources, and improve trust in institutional security measures. However, such monitoring systems also introduce ethical considerations related to privacy, transparency, and responsible data usage.

A key challenge is balancing theft prevention with privacy protection, ensuring that monitoring does not lead to unnecessary surveillance. While the system is designed to limit the collection of non-essential data, any form of monitoring inevitably raises concerns regarding data handling, storage, and protection. Compliance with privacy regulations and ethical standards is critical to ensure responsible implementation. These aspects will be further examined in the discussion.

The remainder of the report is structured as follows: Chapter 2 presents related work; Chapter 3 describes the methodology used throughout the project; Chapter 4 outlines the technical design decisions made; Chapter 5 details the graphical user interface and its development process; Chapter 6 evaluates the system in relation to the project's objectives; Chapter 7 provides a discussion of the results, limitations, and future improvements; and Chapter 8 concludes the report.

2 Related Work

In recent years, there has been a wave of research taking on different approaches to monitoring, managing, and securing networked computing environments. Network uptime monitoring and physical theft detection are some areas where the focus has been. Every system comes with its own set of benefits and downsides, however few of the research models have been about early-stage physical hardware removal detection, and especially not in educational environments or other semi-public settings. The goal of this chapter is to present an overview of related systems, grouped by methodology, and over the gaps in research. Understanding the strengths and gaps in previous research provides a baseline for the development of a new system, particularly through comparison of existing solutions and their relevance to the targeted problem space.

2.1 Network Monitoring Tools

Network monitoring tools are commonly found in IT environments to track the status and availability of connected devices. These tools often aim to ensure service uptime, visualise network topologies for better system overview, and detect issues such as high latency or system failure to enable rapid response.

One example previously used at Chalmers University of Technology is **InterMapper**. This software scans the local network, logs connected devices, and monitors their status [4]. InterMapper will not alert for physical changes, as while it can effectively indicate whether a device is currently online, it lacks the ability to distinguish between normal disconnections and abnormal events, such as unauthorised hardware removal.

One system that goes a step further than InterMapper is **IBM QRadar SIEM**, a Security Information and Event Management (SIEM) tool used to monitor on-site IT systems such as servers and workstations. The system collects data from devices, analyses it to detect patterns, and allows the user to define custom rules that trigger alerts [7]. The primary purpose of QRadar is to identify security events, not to monitor physical device presence.

Setting up QRadar is a complex process that requires significant infrastructure, which has limited its use within large enterprises or governmental environments. These environments typically have a greater need for comprehensive security analytics across multiple subsystems [8].

OP5 Monitoring is another tool used to monitor device health, system performance, and network services. It features visual dashboards, alert mechanisms, and integrations with various third-party platforms to enhance usability. However, like InterMapper and QRadar, OP5 does not track device presence at the physical level. Instead, OP5 focuses on providing insight into the system status by monitoring CPU usage, uptime, and response times [9].

All of these tools are highly capable within their intended contexts, as evidenced by their widespread use. However, they were designed primarily for system health and cybersecurity monitoring, rather than for detecting hardware theft or monitoring disconnections at

device level.

2.2 MAC Address Tracking and Post-Theft Identification

Another category of related systems include methods that identify devices based on their unique hardware addresses. One such method was proposed by Mandaville *et al.* (2012) [10], who presented a post-theft detection system based on logging MAC and IP addresses. The system was developed for use on university campuses, where laptops and mobile devices might be stolen but still have the potential to reconnect to the same network. If the MAC address of a stolen device reappears, the system can alert administrators and use data from wireless access points to help locate the device.

This approach not only addresses MAC address tracking and post-theft identification but also opens up possibilities for further research and mechanisms for post-incident recovery, particularly in environments where devices tend to return to the same network as is often the case on university campuses [11]. However, this system is not designed for any preventative measures or real-time alerting, and its scope is limited to tracking entire network-connected devices. The system cannot detect the removal of internal components that lack individual network identities, such as memory modules or storage drives. The effectiveness of the system depends entirely on if the stolen devices reappear on the same monitored infrastructure. Therefore, this system is primarily useful as part of a larger, multi-layered security framework.

This type of MAC address tracking relies on centralised logging, and network access control (NAC) tools. While such systems provide a record of which devices were connected at a given time, they are more reactive in nature and lack the capability to detect sudden absences or unauthorised removal as the event occurs.

2.3 Physical Tagging and Tamper Detection

Another method for preventing hardware component theft involves using external tags or internal sensors to monitor the physical location or condition of the equipment. One such system was proposed by Cruz *et al.* [12], who introduced a **Near Field Communication (NFC)**-based tracking solution. In their approach, each device was tagged with a unique NFC identifier, and stationary sensors placed in rooms were used to detect the presence of these tags. If a tag left its designated area, the system would send out an alert. This approach is well-suited for environments with tightly controlled layouts, such as laboratories or secure offices.

This method does, however, come with certain limitations. NFC tagging systems require the installation and calibration of a physical infrastructure consisting of sensors and readers throughout all monitored areas. These systems also demand ongoing maintenance overtime to ensure that sensors remain active and properly aligned. Additionally, such systems are prone to false positives, temporary tag removal or insufficient room-level granularity that can trigger unintended alerts.

Another approach is the **Anti-Tamper Radio (ATR)** system, introduced by Staat *et al.* [13]. The ATR approach uses embedded antennas and radio frequency analysis to detect physical tampering. By monitoring electromagnetic patterns inside the chassis of a computer or server, a baseline can be established, and any deviations from this signal may indicate that the device has been opened or altered. ATR systems are highly sensitive and capable of detecting even subtle changes in the physical configuration. This type of solution is particularly suitable for environments where tamper detection is mission-critical, such as industrial control systems, or military-grade equipment.

While ATR has demonstrated high detection accuracy, it requires hardware integration and may not be cost-effective or practical in general-purpose settings. ATR solutions also lacks user friendliness, as it does not include a visual user interface and is often integrated into a broader embedded monitoring platform.

Systems that implement physical tracking and tampering detection are most effective when deployed in controlled or high-security environments. Such systems are less suitable for more flexible or mixed-use spaces where installation and maintenance resources may be limited.

2.4 Motion-Based Theft Detection Systems

Motion-based theft detection approaches the problem by monitoring the physical movement of devices. These systems have been used in several projects, but are primarily focused on securing mobile or personal device. Muthukannan *et al.* (2017) [14] proposed a system that used motion-sensing cameras to detect unauthorised activity in restricted areas. The system was designed to trigger alerts when movement was detected in a monitored zone, such as an office or secure room. This does, however, require that continuous surveillance is accepted within the environment, and that the necessary infrastructure is already in place.

However, even motion-sensing systems raise significant concerns related to privacy, and their deployment remains one of the major challenges. Continuous monitoring through cameras is not acceptable in all environments, such as educational settings. In addition, camera-based systems have several operational requirements, including calibration, secure data storage, and sufficient lightning conditions. These constraints reduce the applicability of such systems in open-access or larger-scale computing labs.

Another innovation within the same field was presented in an IEEE article by Shen *et al.* [15], who explored behaviour-based theft detection in mobile devices. The proposed system leverage built-in motion sensors to determine whether the device is being handled by its authorised user. If any deviation from normal movement patterns is detected, the system triggers an alert. The findings show that behavioural data, rather than direct tracking of physical presence, can be used to detect anomalies.

Motion-based systems can be effective tools for small-scale or individualised security applications. However, these systems require hardware sensors, user profiling, or continuous data collection to operate reliably.

2.5 Hybrid Network Methods

Some systems attempt to address the problem using hybrid methods that combine multiple monitoring techniques, typically involving both passive and active approaches. One such tool is **Nmap**, an open-source network scanner used to identify hosts and services on a network [16]. Nmap analyses responses to packets it sends out to devices, thereby determining their status. Nmap is commonly used for network inventory, vulnerability scanning, and service detection.

While the system is effective for discovering network-connected devices, active scanning can introduce additional load on the network [16]. Firewalls may also detect Nmap scans and flag them as suspicious activity, particularly in managed enterprise environments. Furthermore Nmap must be scheduled to run at regular intervals in order to detect changes, which may introduce a delay between when a device disconnects and when the event is detected.

One solution to the system load problem is to use passive Address Resolution Protocol (ARP) monitoring. As defined in the original ARP specification [17], devices broadcast requests to resolve IP addresses into MAC addresses when needed. By listening to this traffic, a monitoring system can infer which devices are active. ARP monitoring is a less intrusive, passive method; however it lacks speed and precision of active scanning, particularly in environments with infrequent ARP broadcasts.

Whether active or passive, network scanning tools of this kind are primarily designed for infrastructure diagnostics rather than theft detection [18]. Their use in real-time alerting is limited by the need for scheduled scans or background traffic.

2.6 Comparisons of Solutions

Table 1 provides an overview of the systems and methods discussed in this chapter, comparing their core characteristics and limitations in the context of detecting unauthorised device removal. Each approach serves a specific purpose; however, none are explicitly designed to address theft in open or academic computing environments without relying on additional hardware or complex network analysis.

Table 1: Comparison of related theft detection systems. The proposed system (highlighted) is included for reference and comparison.

System/Approach	Real-time	Theft Detection	Hardware Needed	UI	Scalabilit	Primary Limitation
InterMapper	No	No	No	Yes	Moderate	No alerts for suspicious events
IBM QRadar SIEM	Yes	No	No	Yes	High	Complex, focused on cyber threats
Mandaville et al. (2012)	No	Post-theft only	No	No	Moderate	Requires device to reconnect
Cruz et al. (NFC)	Yes	Yes	Yes	Yes	Low	Requires NFC infrastructure
ATR (Staat et al.)	Yes	Yes	Yes (custom)	No	Low	Expensive, no admin UI
Nmap/ARP methods	Partial	Partial	No	Varies	Moderate	Low accuracy, network overhead
Motion Camera (Muthukannan)	Yes	Yes	Yes (cameras)	No	Low	Privacy concerns, infra-heavy
The Proposed System	Yes	Yes	No	Yes	High	Prototype; further testing needed

2.7 Chapter Summary

This chapter provides an overview of the areas addressed by previous research and highlights where existing systems fall short. The systems reviewed cover aspects such as device presence, security monitoring, and theft detection, though they employed varying technical approaches. These solutions are effective in their respective domains - such as enterprise IT, secure laboratories, or mobile devices - by integrating hardware, extensive infrastructure, or assumptions about user behaviour. However, there is a very limited body of research focused on detecting unauthorised physical removal of hardware in shared-use computing environments, such as university labs or public terminals.

3 Methodology

This chapter outlines the methodological approach used in the development of the monitoring system. It includes the project's overall structure, development process, task management strategies, and how the system's functionality was validated. The methodology was designed to ensure adaptability, transparency, and alignment with the project's objectives.

3.1 Project Approach

The project followed an iterative and agile development methodology, incorporating elements of Scrum to ensure continuous progress and adaptability. The primary objective was to develop a Minimum Viable Product (MVP or prototype) for a network-based hardware monitoring system and compare it to existing solutions in theft prevention. The MVP aimed to deliver the core functionality needed for the system to be considered viable, while leaving additional features for later phases.

The methodology was divided into three main phases. The first phase, *Research and Planning*, involved reviewing existing solutions, defining system requirements, and planning the MVP. This phase provided the foundation for the project by gathering insights and establishing clear objectives.

The second phase, *Development and Implementation*, focused on designing and building the prototype. During this phase, the backend, database, and monitoring mechanisms were implemented to create a functional MVP.

The final phase, *Evaluation and Comparison*, was dedicated to testing the system's functionality and comparing it to existing network-based theft prevention methods, such as InterMapper and OP5. This phase was essential for assessing the effectiveness of the solution and identifying areas for improvement.

3.2 Development Process

To manage tasks efficiently and ensure transparency throughout the project, a **Kanban board** within GitHub Projects was created. This allowed the team to track progress, assign responsibilities, and monitor the completion of tasks. Tasks were categorised into the following stages:

- **Backlog**: Tasks that are identified but not yet ready for the current sprint.
- **To Do**: Tasks that are ready for the current sprint and need to be worked on.
- **In Progress**: Assigned tasks that are actively being developed.
- **Ready for Review**: Completed tasks that are pending peer review.

- **Done**: Fully tested and merged tasks.

This structure ensured a smooth workflow and clear progression from backlog to completion, promoting transparency and efficiency within the team.

3.2.1 Development Milestones

The project was divided into five major milestones to track progress efficiently:

- **Database & Backend Setup** (March 1, 2025) – Initial setup of the database schema and backend API.
- **Network Scanning & Device Tracking** (March 10, 2025) – Implementing network scanning, device tracking, and log storage.
- **Alerting System & Notifications** (March 25, 2025) – Developing the notification system for security alerts.
- **UI for Monitoring & Visualisation** (March 28, 2025) – Creating an interface for administrators to monitor system status.
- **Finalised MVP** (April 1, 2025) – Completing the prototype and integrating all core functionalities.

By following this structured approach ensured a *logical progression* through the development phases while maintaining adaptability.

3.2.2 Task Workflow

Tasks were categorised based on their type (e.g., UI, Backend, Database, Bug) and managed through GitHub Issues. A structured workflow ensured that development was transparent, adaptable, and aligned with project deadlines. A new issue is created, assigned to a developer, moved through the stages (In Progress, Ready for Review), and closed once successfully verified and merged.

The methodological framework described above provided a structured yet flexible foundation for developing and evaluating the monitoring system. With a clear workflow in place guiding the implementation, the next step involved a series of critical design decisions. These decisions shaped the system's architecture, technology stack, and core functionalities. The following chapter outlines and justifies these technical choices in detail.

4 Technical Design Decision Process

This chapter outlines key technical decisions made during the system's development, including the limitations of traditional network monitoring tools and the rationale for adopting a lightweight, hybrid monitoring architecture.

4.1 Challenges of Traditional Monitoring Approaches

During the early design phase, we evaluated both active and passive network monitoring methods, such as Nmap and ARP-based scanning. While commonly used for network diagnostics, these tools present limitations when adapted for real-time theft detection.

Active scanning tools like Nmap can cause significant network overhead if run frequently, while passive ARP monitoring tends to be delayed and imprecise. Based on these limitations, we opted not to use either approach for continuous monitoring.

4.2 Adoption of a Hybrid Monitoring Strategy

To ensure accuracy and lower network load, we implemented a hybrid monitoring approach that. One component of this strategy is a heartbeat mechanism — a lightweight process where each device sends regular check-in messages to a central server. This hybrid approach allows the system to detect potential hardware removal in near real time without impacting network performance.

Additionally, the system includes a Discovery page in the administrative interface. This view displays devices that have sent a check-in signal but are not yet registered with location parameters, allowing administrators to manually register them. Registered devices are no longer shown on the Discovery page and are instead moved to the active monitoring view, where administrators can configure notification settings, location data, and device labels.

4.3 Design of the Heartbeat mechanism

Instead of attempting to ping each machine from a central location, we opted for an architecture where the clients themselves would report their status to a central server. This approach would also allow us to host our server application outside of the local network, improving both accessibility and scalability.

4.3.1 Windows Task Scheduler

In exploring solutions, the Windows Task Scheduler was found to provide a practical and user-friendly way to achieve the goals of the heartbeat mechanism. It comes with an

intuitive graphical interface that enables the users to set up scheduled tasks and export them for reuse or deployment across multiple machines.

An essential consideration when scheduling tasks is the selection of the user account under the Security options. Windows Task Scheduler documentation indicates that, for security reasons, tasks created by one user cannot be viewed or managed by non-administrator users. By selecting the built-in SYSTEM account, tasks are automatically executed independently of user login sessions[19]. Selecting the built-in SYSTEM account ensures that tasks begin running immediately upon computer startup, even if no regular user is logged in, and remain hidden and protected from tampering by standard users, significantly reducing the risk of accidental or intentional interference.

As shown in Figure 1, the interface allows administrators to choose whether the task should run regardless of user login status, a critical setting for ensuring background operation.

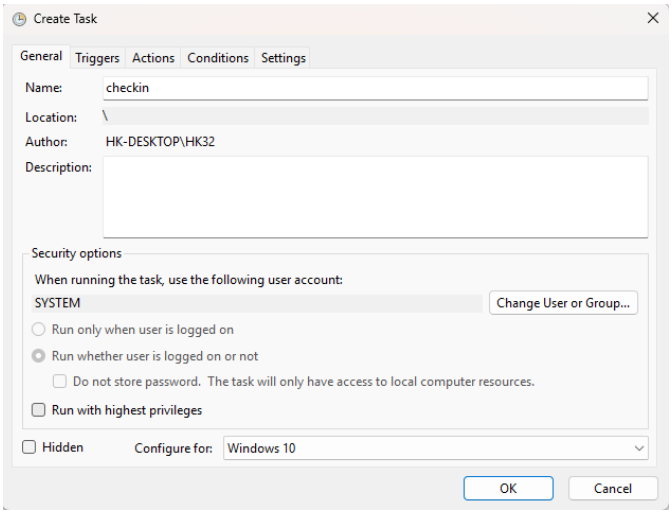


Figure 1: Windows Task Scheduler interface for creating a task displaying option for user account and to run the task whether the user is logged on or not.

The Triggers tab allows configurations for when and how a task executes. Tasks can run on a one-time basis or on recurring schedules such as daily, weekly, or monthly. As shown in Figure 2, advanced settings allow repetition at fixed intervals, with five minutes being the shortest available duration. This interval was suitable for the initial prototype development and demonstration purposes.

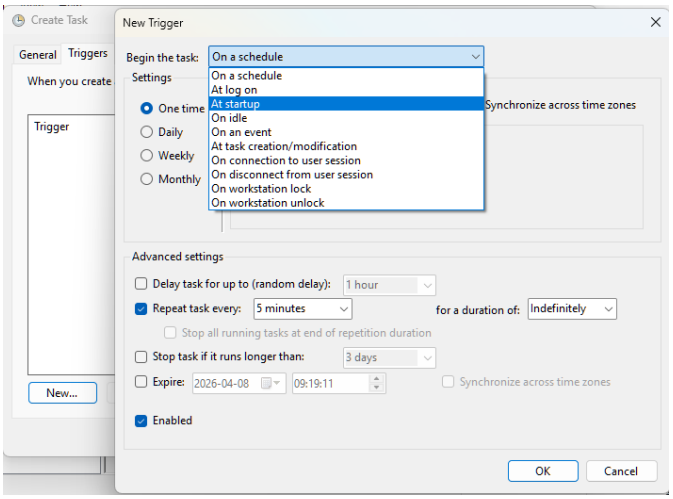


Figure 2: Windows Task Scheduler interface showing trigger options, including startup execution and repeated intervals of five minutes.

Tasks may also run based on specific events such as system startup or user login. For this implementation, two triggers were configured:

- A scheduled execution repeating every five minutes
- Execution at system startup

An additional available trigger for user login was noted as potentially useful for separately logging user login events in the future.

When the task is triggered, the action to be performed can be set to starting a program which in this case would be the actual script or executable that sends the heartbeat.

4.3.2 Heartbeat Script and Executable

To implement the heartbeat mechanism, a PowerShell script was developed to send an HTTP POST request to the backend server. The script retrieves the device's MAC address and includes it in the request body, allowing the server to identify the reporting device. This method was chosen for its simplicity, low overhead, and native compatibility with Windows environments.

The implementation is illustrated in Figure 3.

```
# Retrieve the MAC address of the active network adapter
$macAddress = (Get-NetAdapter | Where-Object { $_.Status -match "Up" } |
    Select-Object -ExpandProperty MacAddress)

# Define the target URL for the check-in request
$url = "http://localhost:8080/checkin"

# Construct the JSON body with the MAC address
$body = @{ macAddress = $macAddress } | ConvertTo-Json -Depth 10

# Send the HTTP POST request to the backend server
Invoke-RestMethod -Uri $url -Method Post -Body $body -ContentType "
   application/json"
```

Figure 3: PowerShell script used to send heartbeat check-ins

Once the PowerShell script was created, it was added under the Program/Script field in Task Scheduler and configured to run every five minutes. This setup functioned as intended: the task was triggered at system startup and continued executing at the defined interval. While effective as a prototype, this approach had limitations in terms of flexibility. For instance, modifying the check-in interval or adding new data fields (e.g., hostname) required manually editing the script or replacing it entirely. It also lacked real-time configurability and could not easily adapt to server-side instructions. To support a more dynamic and configurable solution, a stand-alone executable was deemed necessary to replace the PowerShell component.

To build this executable, several programming languages were evaluated. Java was initially considered due to its familiarity and widespread use. However, Java requires the Java Virtual Machine (JVM) to be installed on the target system, and it does not generate native executables by default, making it less suitable for lightweight or low-maintenance deployment. In contrast, compiled languages such as C, C++, and Golang (Go) offer the ability to produce native binaries without runtime dependencies.

Go was ultimately selected due to its rich standard library, concise syntax, and cross-compilation capabilities. Go provides built-in support for HTTP requests, system calls, sleep functions, and access to system identifiers like MAC addresses and hostnames—making it ideal for building a minimal, efficient check-in service[20].

The refined design included functionality for dynamic interval control, allowing the server to dictate the check-in frequency. Hostname information was also added to the payload to improve device identification. While the final application consisted of approximately 150 lines of Go code, its core logic is summarised in Figure 4.

```
var url = "http://localhost:8080/checkin"
var interval = 15

function main() {
    var mac = os.GetMacAddress()
    var host = os.GetHostname()

    while (true) {
        var response = http.post(url, {"macAddress": mac, "hostname":
            host})

        // Update interval if specified in the server response
        if (response.interval != interval) {
            interval = response.interval
        }

        sleep(interval * 60 * 1000)  // Sleep in milliseconds
    }

}
```

Figure 4: Pseudocode for the Go-based heartbeat executable

4.3.3 Fail-safe Systems

To ensure the reliability and fault tolerance of the heartbeat mechanism, several safeguards were integrated into both the task scheduling configuration and the Go-based client executable.

1. What happens if the executable crashes unexpectedly?

To mitigate unexpected termination due to memory issues, network failures, or bugs, two mechanisms were configured in Task Scheduler:

- *Scheduled Recurrence*: One of the task's triggers is a recurring schedule that executes every five minutes. If the process terminates unexpectedly, it is relaunched automatically at the next interval.
- *Instance Control*: The setting "Do not start a new instance if the task is already running" prevents overlapping executions, ensuring that only one instance of the executable is active at any given time.

Together, these measures create a self-healing environment where any crash results in a restart within a maximum downtime window of five minutes. This approach eliminates the risk of silent failures, sometimes referred to as "underground mode," where a client silently stops reporting.

2. What if the network is down or the server is temporarily unreachable?

The Go-based client is designed to fail gracefully, meaning that it continues operating without crashing or requiring manual intervention. If the server cannot be reached, the

application simply waits and retries at the next scheduled interval. Its stateless design ensures that it automatically resumes normal operation once connectivity is restored. This approach also minimises memory usage and avoids risks of data corruption.

3. Could a user manually kill the task or the executable?

To mitigate tampering, the task runs under the SYSTEM account and is therefore hidden from standard users in Task Scheduler and Task Manager. The executable can also be placed in protected directories such as `C:\Windows\System32` , with restricted permissions.

Even if a user with elevated permissions manages to terminate the process, the recurring schedule ensures the task is relaunched within five minutes. Moreover, interference would be counterproductive, as devices that fail to report are flagged and may prompt administrative follow-up.

4.3.4 Consideration of Windows Services

During the evaluation of different methods for running persistent background processes, Windows Services emerged as a potential alternative to Task Scheduler. Services in Windows are designed to run continuously in the background, can start automatically at boot—before a user logs in—and are often used by core system components such as networking and Bluetooth.

At a conceptual level, this approach appeared to offer deeper OS-level integration and greater robustness. However, upon closer examination, several drawbacks were identified that made services unsuitable for the scope and scale of this project:

- *Setup Complexity*: Configuring a service to start automatically involves a more complicated setup process compared to Task Scheduler. This would have introduced additional overhead during deployment.
- *Development Overhead*: Developing a Windows Service in Go requires the use of specialised packages (e.g., `golang.org/x/sys/windows/svc`) and mandates that the application respond to control signals such as start, stop, and pause. This diverges significantly from a simple looping executable and complicates testing.
- *Deployment Friction*: Services must be installed using administrative tools such as `sc.exe` or via custom PowerShell scripts. This adds unnecessary friction when deploying across large numbers of machines.
- *Debugging Limitations*: Services typically lack interactive output and must be monitored using Event Viewer or custom logging mechanisms. Task Scheduler, by contrast, allows tasks to be tested interactively and provides optional logging of execution history.
- *No Added Security Benefit*: While services initially seemed to offer improved invisibility or protection, the existing setup—where the executable runs as SYSTEM via Task Scheduler—already fulfils these requirements. As such, no tangible security improvement was found.

In summary, while Windows Services are powerful tools for certain use cases, their added complexity was not justified in this context. The Task Scheduler-based solution met all the functional and operational requirements, including invisibility, reliability, automatic startup, and ease of deployment and maintenance. For these reasons, the Task Scheduler was retained as the primary mechanism for executing the heartbeat client.

4.4 Device Status Detection

The system determines device status by evaluating check-in intervals. If a device fails to send a check-in within a short window (e.g., one minute), it is flagged as potentially at risk. If no check-in is received within a longer threshold (e.g., five minutes), the device is marked as offline and an alert is generated for administrative attention.

This model enables both proactive monitoring and timely notification. It provides a balance between early warning for transient issues and confirmation of prolonged offline states.

4.5 Technology Stack

The backend was implemented in Java using Spring Boot framework which manages HTTP requests, check-in interval logic, and notification processing. This choice was motivated by several factors. First, Spring Boot simplifies the process of building and deploying stand-alone applications, it can be run directly without the need for external application servers or complex configuration, which reduces overhead and simplifies both development and deployment. Second, it offers strong support for building RESTful APIs, and includes robust tools for dependency injection, scheduling, and error handling, which matched the project's need for periodic device check-ins and alert logic.

All data is stored in an embedded H2 SQL database selected for its lightweight nature and ease of integration with Java-based applications. H2 enables fast local prototyping without requiring external setup, while still supporting complex queries. Device activity is tracked in the `Checkins` table, while longer outages are logged in the `OfflineEvents` table.

When a device starts up, its client requests the current interval configuration from the backend. This interval determines the frequency of subsequent check-ins. The system also includes logic to compare timestamps of incoming check-ins and identify missing intervals. Offline events and status changes are reflected in both the database and the user-facing notification system.

A Discovery page is provided to assist administrators in registering new devices that have checked in but are not yet explicitly monitored because location information is missing. These can be promoted to the registered list through manual review. The combination of real-time check-in logic, persistent state tracking, and alert generation ensures reliable device monitoring across the network.

4.6 Summary of Design Rationale

Throughout the design and implementation process, multiple approaches were evaluated for running background processes reliably on Windows machines. Two viable options were considered in detail: Windows Services and Windows Task Scheduler. Windows Services offer deeper OS-level integration, automatic startup before user login, and robustness for long-running system applications. However, they also introduce increased complexity in development, installation, and debugging. Creating a Windows Service in Go requires the use of specialised packages and mandates support for lifecycle control signals, which added considerable development overhead.

In contrast, Windows Task Scheduler provides a simpler and more accessible alternative. It allows for scheduled and event-driven execution, supports privilege escalation via the SYSTEM account, and includes a graphical interface for easy task creation and testing. Most importantly, it aligns well with the goals of this project: to enable periodic, secure check-ins from client devices in a distributed lab environment with minimal deployment friction.

Given the project's time constraints, operational scope, and operational requirements, Task Scheduler emerged as the most pragmatic and effective solution. While it may not offer the same level of integration as Windows Services, it fulfilled all functional needs and provided sufficient control, reliability, and maintainability. This decision reflects a broader principle in system design: the optimal solution is not necessarily the most technically sophisticated, but the one that best fits the specific constraints and context of the problem at hand.

5 Graphical User Interface Design Processes

To create a user-friendly and effective monitoring system, a graphical user interface (GUI) was designed with a strong focus on usability. Prior to this project, no dedicated GUI for theft monitoring was available at Chalmers; the previously used Intermapper system relied on ping-based status checks and email notifications, lacking any real-time interface or administrative visualisation. Usability was therefore prioritised to deliver a high-quality product, increase the productivity of system administrators and IT personnel when managing hardware theft incidents, and minimise the associated institutional costs. These objectives underline the importance of not only designing a functional system, but also aligning it with real user needs and institutional limitations. Successful user experience cannot be imposed; it must be designed for the experience, taking into account user workflows, expectations, and context [21].

To further guide the prioritisation of user needs, Maslow's hierarchy of human needs was used as a conceptual lens on foundational requirements before introducing more advanced features. In line with the idea that the user tends to prioritise products that meet basic needs before engaging with higher-level functionalities [22]. Since there was no existing GUI to use as a reference point, Maslow's hierarchy served as a helpful framework for establishing a design process focused on addressing users' needs. The system design first focused on establishing security and safety, corresponding to the second level of Maslow's hierarchy (see Figure 5). Features such as device tracking, network scanning, and disconnection alerts address to meet user's core need for hardware theft protection and situational control in the computer laboratories. The dashboard aggregates essential system data, offering administrators with immediate visibility and reinforcing a sense of control and safety that the computer rooms are secure.

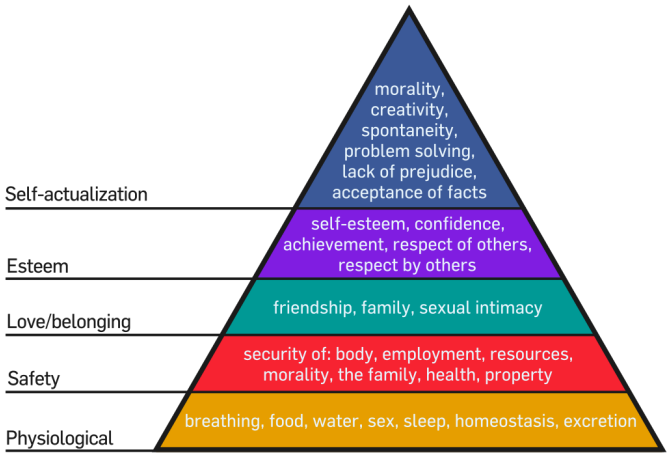


Figure 5: Maslow's hierarchy of Needs. Adaptation of Maslow's hierarchy of human needs to the monitoring system context. The design prioritise fundamental needs such as safety and security before addressing higher-order needs like efficiency, confidence, and professional mastery. **Source Image:** CC BY-SA 3.0 via Wikimedia Commons.

5.1 Supporting Confidence and Operational Efficiency

After laying the groundwork for basic safety and monitoring functionalities such as device tracking, disconnection alerts and real time visibility across computer labs. The system design progressed toward addressing user confidence and operational efficiency, factors loosely coupled with the esteem level in Maslow's hierarchy of needs (see Figure 5). Although Maslow's model originates from psychology, it served here as a conceptual framework to illustrate how users first require functional security before benefiting from features that support autonomy and effectiveness.

To that end, the GUI was structured to minimise interaction barriers through a clear visual hierarchy, consistent iconography and well-defined navigation, for example, separate views for Logs & Events, Registered Devices and Settings (see Figures 8, 7, 11). These design choices are intended to strengthen users sense of control and reliability under time-sensitive conditions. To complement this section, the following figures present a complete overview of the graphical interface of the system. The goal is to illustrate how the individual components support user needs in various operational situations:

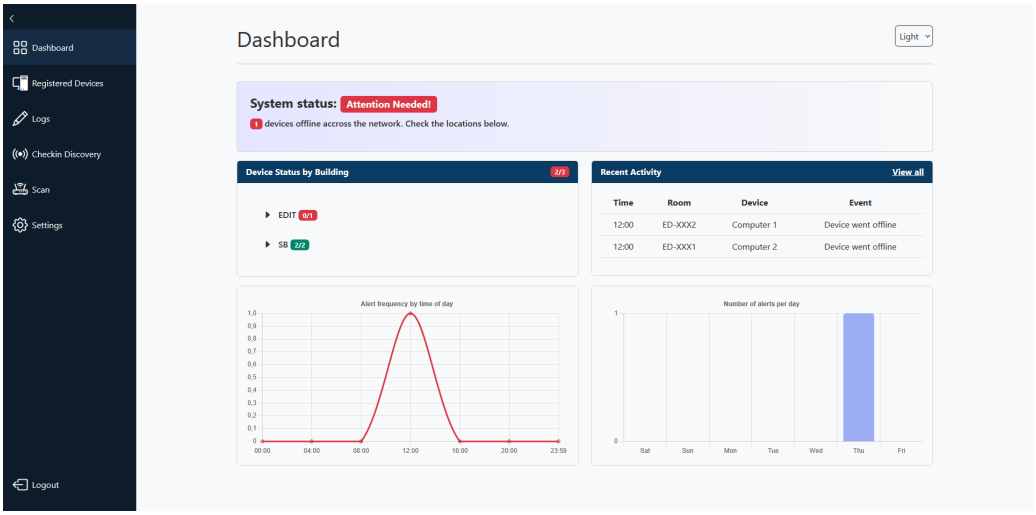


Figure 6: The dashboard page displays the overall system status, device status by building, a table with recent activity and two graphs: The first one showing alert frequency by time of day and the second one the number of alerts per week. Together, the elements provides a clear overview of device activity.

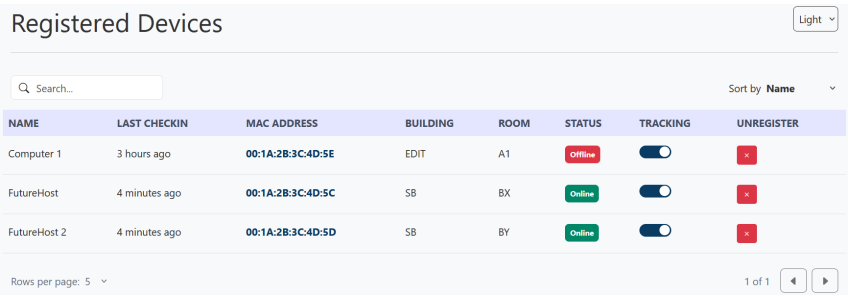


Figure 7: The registered devices page lists all registered devices, each assigned to a specific building and room. The system will send an alert when a device marked as tracked goes offline. When the unregistered button for a device is pressed, the device is moved back to the Check-in discovery page.

Figure 8: The logs page displays a record of when devices goes offline and are restored. Notes can be added for tracking and documentation purposes.

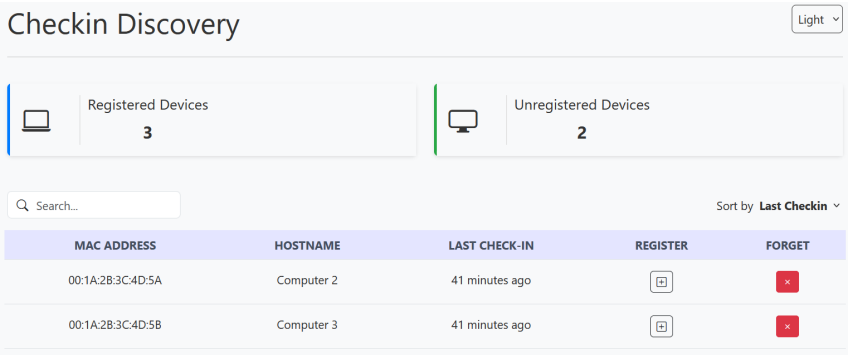


Figure 9: Checkin discovery page showing the number of registered and unregistered devices. Discovered devices are displayed in a table, and when the register button for a device is clicked, the user is prompted to assign the device to a building and room. The device is then moved to the registered devices page.

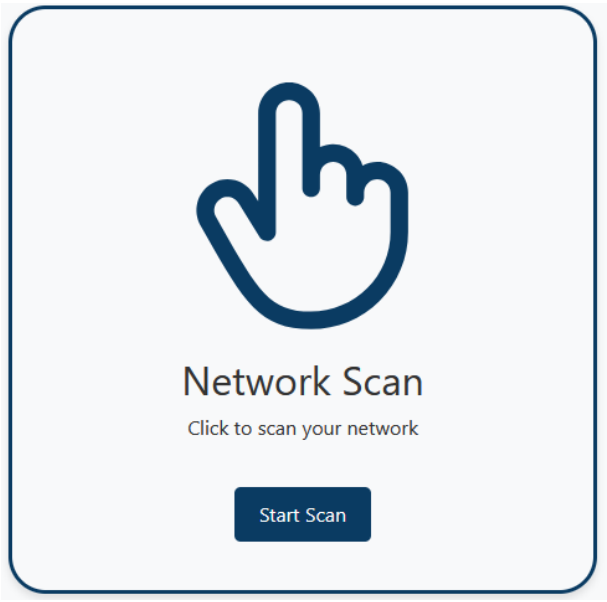


Figure 10: The scanning page displays an animation when the ”start scan” button is clicked and redirect the user to the check-in discovery page.

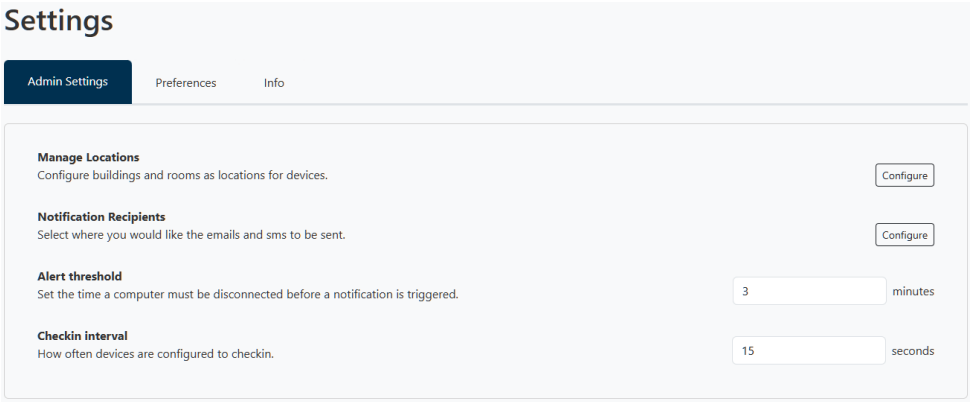


Figure 11: A settings page with three tabs: (1) Admin settings where administrators can add locations, configure recipients for notification, set alert threshold and adjust the Check-in interval (2) Preference tab where the appearance of the application can be changed. (3) Info tab providing additional information about the application.

5.2 Advanced Features: Supporting User Autonomy and Mastery

Apart from covering basic security and operational needs, the design also incorporates features targeting higher-level user needs. According to Maslow's hierarchy, self-actualisation reflects the user's desire for autonomy, mastery, and personal optimisation [23]. To support these higher-level goals, the monitoring system addresses several advanced customisation options. One example is the Settings page (Figure 11), which allows experienced IT personnel to define what qualifies as an ”abnormal disconnection”, adjusting the system's sensitivity to fit specific operational contexts. Users can also decide which devices

to selectively track, configure disconnection, and check-in intervals, providing users with a greater sense of control over system behaviour.

Though not essential for basic system operation, these features help expert users foster a greater sense of ownership, flexibility, and competence, aligning with the top tier of Maslow's hierarchy.

The overall design approach was based on a structured prioritisation of needs, first determining that the core functionalities were in place before integrating the higher-order customisation options. Reliable device tracking, real-time disconnection alerts, and clear visibility of the dashboard established a strong foundation of safety and trust, which are critical prerequisites for greater user engagement and satisfaction [22].

Only after securing these central needs did the system introduce advanced features such as adjustable thresholds, dark mode and flexible tracking settings found in the pages Check in Discovery and Registered devices (see Figures 9, 7). This phrased development reflects a human-centred design philosophy, emphasising that users must feel secure, competent, and confident in fundamental interaction before engaging meaningfully with more complex system customisations [23].

5.3 Iterative Interface Development and User-Centred Refinements:

The iterations of the GUI part in the monitoring system followed an user-centred design (UCD) process grounded in Human Computer Interaction (HCI) methodology, ISO 9241-11 standards in usability and user centred design[24], and practical user feedback. Each stage of the design process combined functional requirements from HCI principles, cognitive considerations, and empirical insights gathered through interviews.

5.3.1 Functional Requirements and Data Gathering

Initial functional requirements were derived from practical scenarios in which the design team assumed the role of IT administrators at Chalmers, following a narrative-based walk-through approach. Rather than relying on abstract use cases, the team used storytelling and role play to map out what users would actually require in high-pressure situations, such as responding to theft, unexpected disconnection, or device malfunction.

Early-stage scenario testing and data analysis were also applied, with personas developed to represent real user types [21]. The personas included novice administrators, lab managers and experienced IT security staff. These personas informed both the functional requirements (e.g., real-time visibility and alerting) and data requirements (e.g., MAC address, last check-in time, room, and building location) [21].

5.3.2 Cognitive Process Considerations

Cognitive factors such as problem-solving, planning, decision-making and learning directly influenced the layout of the application [21]. For example, features like the disconnection alert system support decision-making under pressure by highlighting anomalies with clear colour cues and offering a guided response pathway, as illustrated in the Logs and Events page (Figure 8). Similarly, the Settings page (Figure 11) was designed to support learning and user autonomy by exposing system behaviour in a structured and explainable way, allowing experienced users to adjust thresholds and notification parameters without risking accidental misconfiguration.

5.4 Iterative Design and Feedback Loops

The GUI was developed through three iterative stages. The first two iterations were prototyped in Figma, a browser-based interface design tool, to establish and refine the interface structure, while the final version was implemented in HTML, based on feedback and usability evaluations.

5.4.1 Iteration 1: Establishing Foundational Structure

The first iteration focused on establishing the core layout of the interface (See Figure A.2 in Appendix A), including a left-hand navigation panel and key system pages such as Dashboard, Logs, Online Devices and Settings. The primary goals were to improve learnability and recognition by using recognisable icons, maintaining consistent placement of interface elements, and ensuring immediate visibility of system status.

5.4.2 Iteration 2: Improvement and Refinement

The second iteration was guided by internal heuristic evaluations and early user feedback (visualised in Appendix A.3). Several accessibility improvements were introduced, including refined colour contrast, improved icon consistency, and clearer navigational labels.

Feedback from David Gidlöf Sjöquist, a teaching assistant in the *Design and Implementation of Graphical Interfaces (DAT216)* course at Chalmers, was particularly valuable. He emphasised the need for administrators to quickly grasp system overview, which led to the addition of summary cards on the dashboard and filtering tools for device status. He also pointed out the need for customisable alert tiers and suggested restructuring the Logs page to make incident analysis more effective. His input directly improved user trust, clearer dashboard, and greater situational awareness.

5.4.3 Iteration 3: HTML Implementation

The third iteration was implemented in HTML and reviewed by Arne Linde, Senior Lecturer and Head of the Computer and Network Systems Division at the Department of Computer and Information Technology at Chalmers.

He provided expert insights that critically shaped the final version of the system prototype. The most significant contributions included:

- Efficient overview of more than 1000 devices
- Alert routing based on operational hours
- Configurable alert thresholds for each computer lab, adapted to the specific condition of individual rooms

One of Linde's primary concerns was that early versions of the prototype presented device-level information in a way that became difficult to manage at scale. To address this, the final system prototype included a redesigned dashboard that grouped computer statuses by building and room, rather than displaying individual devices. This revised layout is shown in Figure 6.

These changes aligned well with the HCI principle "*match between system and the real world*"[21], as they reflected how IT administrators typically think about infrastructure, in terms of physical spaces rather than specific machines.

Additionally, Linde pointed out the need for customisable alert thresholds, allowing administrators to define what constitutes abnormal behaviour in specific contexts. For example, alerts could be triggered only if multiple devices go offline simultaneously in the same room (Figure 11).

As a result, the Setting page was extended to support user-defined sensitivity levels and adjustable check-in intervals.

The third iteration also demonstrated a stronger alignment with the ISO 9241-11 standard for usability [24]. Evaluations focused on the following dimensions:

- **Effectiveness:** Ensuring that the system reliably supports detection and response to hardware loss or downtime.
- **Efficiency:** Achieved through views such as *Registered Devices and Logs& Events*, which incorporate search and filtering tools to support faster decision-making (Figures 7 and 8)
- **Learnability and Memorability:** Facilitated by consistent layout structures, including left-hand navigation and tabbed views, enabling inexperienced or returning IT personnel to quickly orient themselves.

- **Context of use:** Addressed through feedback from Linde, including large-scale lab environments and non-standard conditions such as temporary tent deployments.

Together, these refinements reinforced the system's ability to handle real-world use cases. They also demonstrated a matured balance between human factors, interface clarity and back-end flexibility.

5.4.4 Summary

In summary, Arne Linde's expert evaluation validated several core design decisions while also prompting refinements that enhanced the system's scalability, flexibility, and alignment with institutional IT management practices.

His feedback played a key role in confirming that the final product not only fulfilled its technical requirements but also delivered a user experience grounded in real world workflows.

5.5 Design Principles and Human Computer Interaction (HCI) Considerations

Building on the usability goals described earlier in the GUI chapter, the interface was developed with guidance from established HCI principles, particularly Jakob Nielsen's heuristic evaluation [21], as well as practical insights into IT workflow.

Heuristic evaluation involves assessing whether elements of the user interface conform to recognised usability principles, with the aim of creating systems that are intuitive and easy to use [21].

Three core HCI principles guided the GUI development in this project:

- **Visibility of System Status:** The system should continuously inform users about its current state through appropriate and timely feedback, enabling users to understand system behaviour without confusion [21].

  In the developed monitoring system, this principle is implemented through the *Registered Devices* page (Figure 7), which provides immediate and transparent visibility into each device's status. For every listed computer, attributes such as last check-in time, MAC address, building, and room are displayed, enabling administrators to verify presence and location efficiently.

  Connectivity status is communicated through distinct, colour-coded labels: green for "Online", yellow for "Disconnected", and red for "Offline". These visual cues enable rapid situational awareness across multiple rooms, without requiring additional clicks or navigation. Furthermore, real-time updates ensure that disconnection and reconnection events are reflected immediately, supporting quick incident response and effective system monitoring.

- **Recognition Rather Than Recall:** In designing the monitoring system it was crucial to prioritise recognition over recall, thereby reducing the user's memory load by making key objects visible and easily accessible [21].

  The monitoring system minimises cognitive effort by providing familiar, consistently structured information across key views such as *Registered Devices* and *Checkin Discovery* (Figures 7 and 9). Rather than requiring IT personnel to memorise MAC addresses or device locations, the interface displays relevant details, including Hostname, MAC address, IP Address, building, room, and last check-in time, directly within sortable tables.

  Additionally, filter and search functionalities in Registered Devices and Check-in Discovery pages allow users to quickly locate devices using recognisable attributes. The organisational design promotes rapid recognition, enabling users to focus on their tasks without unnecessary cognitive effort or reliance on memory.

- **Flexibility and Efficiency of use:** This principle refers to a system's ability to accommodate both novice and experienced users by offering shortcuts and customisation options [21]. For inexperienced users, the interface remains simple and accessible, requiring no prior knowledge. Simultaneously, the system enables more experienced users to perform frequent actions more efficiently by accessing advanced features. These enhancements are typically not necessary for beginners but can significantly increase productivity for users who are familiar with the system's functionality.

  Within the monitoring system, the *Settings* page (Figure 11) exemplifies this principle by allowing administrators to customise operational parameters. While the system functions effectively with its default configuration, experienced IT personnel can adjust alert thresholds, define custom check-in intervals, and configure notification recipients, thereby tailoring the system to specific institutional needs and workflows.

5.6 Colour Scheme and Visual Hierarchy

The visual design of the interface was developed with a focus on clarity, accessibility, and long-term usability, particularly for IT staff working in low-light environments such as server rooms. A dark mode theme was introduced to reduce eye strain and enhance visual comfort during extended periods of system monitoring, as illustrated in Figure 12.

To ensure consistent and immediate understanding of device states, colour-coded labels were used to represent system status:

- **Green:** Normal operation (Online)
- **Yellow:** Temporary issues or warnings (Disconnected)
- **Red:** Critical conditions, such as prolonged disconnections (Offline)

During the early iterations, the colour contrast for some of the status labels was insufficient, which potentially reduced readability. In the second iteration (as shown in Appendix A.3), this issue was addressed by adjusting the colour combinations and validating them using WebAIM Contrast Checker tool [25]. These refinements enhanced legibility across various devices and improved the visibility of status indicators, even for users with limited colour perception or visual fatigue.

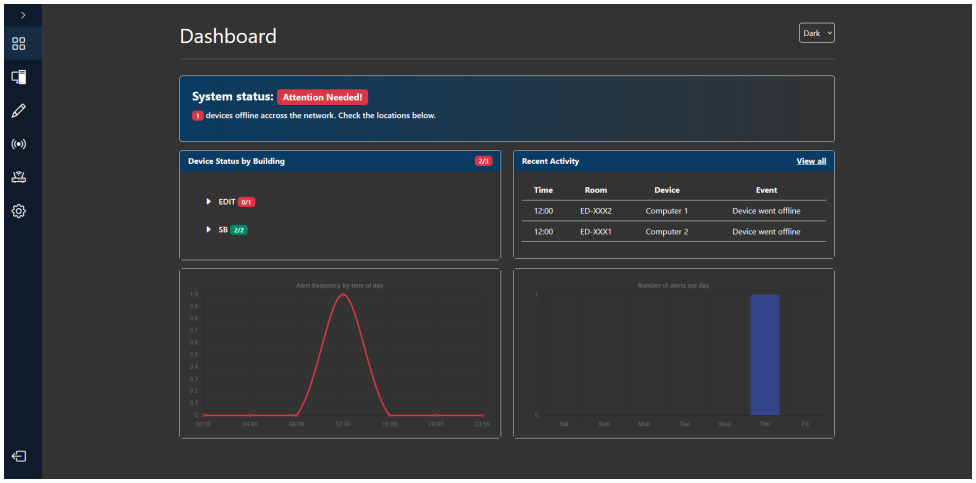


Figure 12: Dashboard illustrating Dark Mode

The final design presents these labels prominently, enabling quick recognition of incidents and reducing the time required to interpret device states.

Additionally, view names were updated to improve semantic clarity:

- **”Online Computers”** in iteration one (shown in Appendix A.2) was renamed **Check-in Discovery**, signalling its function as a real-time detection view.
- **”Tracked Devices”** in iteration two (illustrated in Appendix A.3) was renamed **Registered Devices**, making it clearer that the page deals with actively monitored systems.

These adjustments enhance navigation, reduce cognitive friction, and align with key HCI heuristics such as ”recognition rather than recall” and ”consistency and standards” [21].

5.7 Interaction Flow and Navigational Efficiency

To support the structural layout and usability-oriented interface design, a visual flow diagram was created to illustrate how users interact with the system when performing tasks such as device monitoring, log analysis, and administrative adjustments. This diagram included in Appendix A.4, outlines the logical sequence of interaction and highlights the decision-making processes in typical real-world IT environments.

The interaction model was structured according to User-Centred Design (UCD) principles, with task flows designed to reflect the mental models and daily workflows of administrators. For example, users begin on the *scan page* (Figure 10) and are then immediately presented with a summary of device status across buildings (Figure 6). When the system signals a critical alert, such as multiple devices going offline within a single room, the user can quickly navigate to the *Logs* or *Registered Devices* page to investigate further and determine whether intervention is necessary (Figures 7 and 8).

Each branch in the flowchart corresponds to an intuitive task path and includes a clear decision point, "Are any of the status indicators red?". Which directly support a situational awareness and reduces cognitive overload during high-pressure scenarios.

5.7.1 Navigational Perception and Interaction Efficiency

The interface's layout and flow were informed by Fitt's Law, which states that the time to reach a target is a function of the distance and size of the target [26]. The principle guided key design, such as transforming dashboard status indicators into large, clickable navigational elements to minimise movement effort and enable rapid access to relevant information. These elements functions as pliable affordances, visually indicating interactively and guiding the user's next step through visual and spatial hierarchy[27].

Additionally, the use of a persistent sidebar, combined with context-aware breadcrumbs at the top of the page, for example on the system's recipients page (accessed through the settings page), supports navigation. The breadcrumbs give users the ability to return to previously visited pages in the system without using the navigation panel (Figure 13). The layout helps users maintain their orientation without losing track of their position within the application structure.

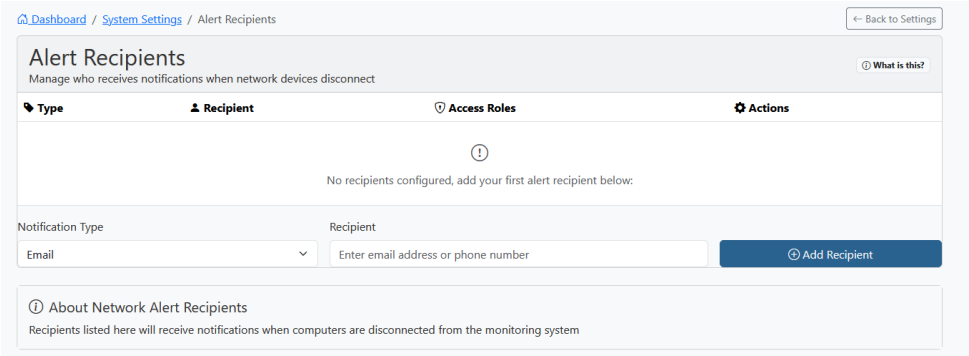


Figure 13: Alert Recipients page

The navigation panel remains accessible from any page, and major actions such as logging out, viewing logs & Events or scanning the network are placed in predictable vertical positions to support motor memory (see navigation panel in Figure 6). In practice, this consistent positioning reduces the need for users to visually scan the entire interface repeatedly, thereby accelerating routine tasks and lowering interaction cost over time.

5.7.2 Dashboard as a Decision Hub

The dashboard page (Figure 6) serves not only as a display area but also a central decision-making hub, a concept inspired by dashboard design patterns in enterprise and surveillance systems [27]. From this view, users can:

- Assess the operational status of all connected lab environments
- Access logs for offline events
- Trace and inspect individual devices
- Navigate directly to configuration views based on urgency of incidents

The structure reflects goal-oriented workflows rather than feature-based navigation, reinforcing the systems user-centred design (UCD) focus.

5.8 Component Selection and functionalities

Each interface view was purposefully constructed using well-established User Experience (UX) design patterns to align with user goals and workflows. From data management tables to form-based configurations, the selected components are consistent with common design patterns to support clarity, task efficiency and intuitive interaction.

- In the *Check-in Discovery* view, the use of segmented cards (Registered vs Unregistered) follows a master-detail pattern that supports prioritisation and rapid assessment of device status (Figure 9). Interactive table rows with sorting and search functionalities support quick identification and organisation of newly detected devices.
- The *Registered Devices* page implements a classic data table pattern with pagination, inline controls (such as toggle switches for tracking), and clear status indicators (Figure 7).
- The *Logs & Events* view supports investigative workflows. Collapsible sections and inline filters follow a dynamic disclosure approach, enabling users to reveal detailed log entries only when needed (Figure 8). The layout reduces cognitive load and aligns with how users process information during urgent log reviews.
- In the *Settings* interface, a tabbed layout separates configuration categories such as Admin Settings, Preferences, and Info (Figure 11). The use of form inputs, dropdowns, and stepper controls for alert thresholds follows well-known patterns from system dashboards, improving learnability and reducing user errors.

5.9 Dashboard

The dashboard was developed as the primary point of situational awareness, designed around modular User Interface (UI) components to communicate system status and support immediate user action. Each component was selected to fulfil a specific function within the dashboard interface (see Figure 6):

- **Cards** were used to group related content together. The cards also stack naturally which can help the design be more responsive when combined with grid and flexbox layouts.
- **List and sublists** were used to organise buildings and rooms. Buildings containing rooms with the highest number of offline devices are prioritised at the top of the view enabling administrators to focus on areas requiring immediate attention. This approach is easier to implement and avoids cluttering the interface with additional filtering components.
- **Drop-down menus** were used throughout the applications to conserve space and present relevant options contextually. For instance, the drop-down in the dashboard allows users to switch between light and dark mode.
- **Graphs and charts** were used to visualise data in an intuitive format, improving interpretability and supporting the identification of trends or anomalies within large datasets [28]. Using this method may help administrators make informed decisions without the need to manually sift through datasets.
- **Tables** were included to present granular details such as room names and individual device statuses, where precision is essential.
- **Status indicators** were used to highlight rooms or buildings requiring immediate attention. A green indicator signifies that all devices are online, yellow denotes that fewer than 100% are online, and red indicates a critical issue where fewer than 75% of devices are online.
- **Badges** were used to draw attention to the overall status of the system and draw attention to the number of devices in each building and each room.

A panel-based structure was chosen for its modular nature and its alignment with monitoring workflows, where operators require both overview and detailed information within a single interface.

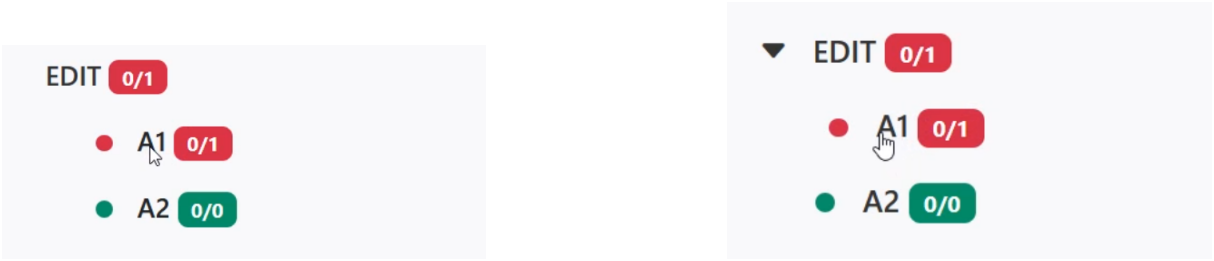
5.10 Pliancy and creating interactive components

Pliancy is about showing that a component in the interface can be clicked or interacted with [29]. The methods used to signal pliancy in the application were inspired by examples provided in *About Face* by Cooper et al., as well as common pliancy indicators observed on various general purpose websites (e.g. Wikipedia) and security platforms (e.g. IBM QRadar).

Static hinting is one of the methods described in the book and was implemented in the application to convey interactivity before the user engages with the component. One example of static hinting implemented in the application is the use of triangle icons. It was observed that a right pointing triangle or arrow usually indicated that additional information is hidden and can be shown when the component is interacted with, while a downwards-pointing triangle suggests that the content is currently visible and can be hidden.

Dynamic hinting is another method discussed in the book and was implemented in the application to communicate that a component can be interacted with at the moment of interaction. The hints implemented include hover effects, animations and cursor hinting. Hover effects were applied to the buttons to give an immediate hint on interactivity, while animations were used for sublists in the dashboard where hover effects seemed less applicable. Additionally, changing the cursor from a pointer to a hand reinforces that a component is clickable, which was also mentioned by the author.

As illustrated in Figure 14, the example highlights the difference between non-pliant and pliant components.



(a) List and sublist without pliancy

(b) List and sublist with pliancy

Figure 14: Comparison of a list and sublist design. (a) shows a static version without any interactive visual cues. (b) includes pliant design features such as a toggle-able triangle icon, subtle animations, and cursor changes on hover, indicating interactivity and improving usability.

5.11 Statistical Dashboard and Operational Justification

The statistical dashboard aggregates system data and presents it in a structured format that supports both quick assessment and deeper analysis. It was designed not only to provide administrators with real-time operational status but also to facilitate the identification of long-term patterns and anomalies. The interface enables exploration across multiple levels of detail, ranging from high-level summaries to specific problem areas that may require follow-up.

At the top of the page, summary cards present key metrics such as average downtime per event, the total number of incidents within the selected time range, and the number of devices currently offline. Below these, bar charts, line graphs, tables, and pie charts provide easy to understand breakdown of when and where events are happening — highlighting, for example, the most frequently occuring tags, the most affected rooms or buildings, and the times of day issues tend to occur. These visualisations also help explain why events

are happening, using tags that administrators added to the individual events. These visualisations support pattern recognition and contextual understanding that would otherwise be difficult to extract from isolated event logs.

The statistics dashboard was designed to support administrative workflows, particularly in busy or time-sensitive situations. The primary purpose is to minimise the time between recognising a problem and understanding its underlying cause. This aligns with established usability research, which highlights that dashboards are most effective when they clearly convey system status and reveal patterns without overwhelming the user [30].

False Positive Estimate

One of the more advanced features of the statistical dashboard is its ability to estimate the number of ongoing offline events that are likely to be false alarms (unless they already have been manually marked as a real or false alarm). This estimation assists in understanding the severity of the current situation. A higher proportion of false positives allows administrators to shift focus from investigating individual disconnections to identifying broader patterns.

The initial version of the estimation system was based on static assumptions regarding specific tags - for instance, "Power Outage" were typically treated as low-risk, while "Hardware Removed" was seen as more critical. In the current version, the system refines these estimations over time by learning from previously confirmed events. When enough historical data is available, it computes the observed false positives rate associated with each tag or location. These empirically derived probabilities are then used to give a more accurate estimation of the proportion of ongoing events that are likely to be false alarms.

The results are displayed directly on the dashboard, both as an aggregate estimate (e.g., "approximately 7.5 of 30 ongoing events are likely false alarms") and through detailed breakdowns by tag and location. This probability-based approach, which is commonly used in modern anomaly detection systems,enhances situational interpretation by offering contextual insights rather than solely presenting numerical data [31].

Operational Value

Beyond its analytical value, the dashboard enhances the system's operational usability. It enables administrators to identify structural weaknesses - such as frequently affected locations, recurring event tags, or devices with repeated offline occurrences — and supports informed decision-making. The system promotes transparency by clearly presenting the reasoning behind alerts, enabling administrators to verify and interpret system behaviour rather than relying on hidden logic.

Overall, the statistical dashboard translates extensive system data into a structured and interpretable format. It supports both quick response and longer-term improvements, and represents a key component in making the monitoring system more trustworthy and manageable in real-world environments.

A high-resolution view of the statistics page is provided in Appendix A.1, illustrating how each component integrates into the overall monitoring workflow.

5.12 Chapter Summary

The chapter contains the iterative development of a graphical user interface for a monitoring system, shaped by UCD principles, HCI methodology, ISO 9241-11 usability standards, and user feedback. The contributing factors in the design process included usability, workflow alignment, and situational awareness.

Initial Figma prototypes defined interface structure and component layout, while the final HTML implementation incorporated feedback from user interviews. Maslow's hierarchy of needs was applied as a conceptual model to guide feature components, moving from foundational monitoring tools to advanced configuration options for experienced users.

The result is a system that integrates theoretical decision principles within an interface, supporting real-world administrative tasks through usability, clarity, and flexibility.

6 Evaluation

The evaluation of the system was guided by the objectives defined in Section 1.3, which set specific targets for detection capabilities, alert responsiveness, and accuracy (minimising false alarms). This chapter describes how the prototype was tested and the extent to which those goals were achieved. The evaluation approach ties back to the original problem statement by examining whether the solution indeed provides the real-time theft detection that was missing in existing monitoring tools. It also assesses the system's overall performance and usability in a simulated lab setting, determining how well the project met its defined objectives.

6.1 Evaluation Setup and Methodology

To assess the system under realistic conditions, a series of controlled tests was conducted that mimicked scenarios in a shared computer lab. The test environment consisted of several computers running the monitoring client, all connected to a central server. Two test scenarios were constructed to simulate potential hardware theft events. First, an unexpected device disconnection was simulated by abruptly unplugging a computer's network cable and powering off the machine without warning. Second, a hardware component removal was simulated by disconnecting a hard drive from a running computer. This action typically caused the machine to crash or go offline, representing an unauthorised removal of a critical component.

In addition to these theft simulations, a benign scenario was included to check for false alarms. This involved performing both scheduled reboots and abrupt, unscheduled reboots of a machine to observe whether the monitoring system would mistakenly flag such normal events as potential thefts. Throughout all tests, the heartbeat interval (the frequency at which each client reports its presence to the server) was configured to 15 seconds. The outcomes of each trial was measured, including whether an alert was triggered and the time taken from the event to the alert notification. This methodology ensured that each project objective could be evaluated: detection of hardware removal, detection of network disconnects, alert delivery speed and avoidance of false positives.

To complement the controlled lab tests, additional evaluations were carried out in a real-world environment at Chalmers University of Technology (ED4225 computer lab) to validate the system under practical conditions and evaluate its usability. With assistance from a Chalmers system administrator, the installation procedure was carried out as documented in Section 4.3.1. The heartbeat client was successfully deployed on the devices used during the test under the SYSTEM account, and the devices appeared immediately in the Device Discovery section of the application.

Several tests were conducted to evaluate the system's behaviour under both typical and edge-case scenarios. These included simultaneous offline events (e.g., two machines being disconnected at the same time) and short-term network interruptions that remained below the alert threshold

6.2 Results and Objective Fulfilment

The combined results from the controlled lab tests and additional real-world testing indicate that the system meets its core functional objectives. Below is a summary of each objective and the corresponding outcomes:

1. **Detecting hardware removal:** In all test cases where the removed hard drive contained the operating system, the monitoring system successfully detected the event. The immediate system crash and loss of network connectivity triggered an alert every time. However, when a hard drive that did not host the operating system was removed (a non-critical drive), the system registered no loss of heartbeat and no alert was raised. Thus, the first objective was only partially achieved, the system reliably detects hardware removal only if that removal causes the device to crash or disconnect from the network.

2. **Detecting unexpected device disconnections:** The system consistently detected unscheduled network disconnections. Multiple trials of unplugging a computer's network connection or suddenly shutting down a device all resulted in the monitor raising an alert shortly after the device went offline. In fact, the detection rate for these disconnect events was effectively 100% in our tests. The system noticed each device's absence within roughly one heartbeat interval of it losing connectivity. This confirms that the second objective, *detecting unexpected device disconnects in real time*, was fully fulfilled with a comfortable margin.

3. **Alert notification speed:** Once a potential theft was detected, the system generated and delivered alerts to the designated recipients within approximately three minutes. In our measurements, the time from an event occurring to an alert being received was determined primarily by the configured check-in interval (15 seconds) and the alert threshold (3 minutes). This performance meets the third objective, as the system was able to provide prompt notifications upon detection of an incident. In other words, the requirement for timely alert delivery was fulfilled under test conditions.

4. **False alarm avoidance:** No false positives were observed during controlled tests or practical evaluations. For instance, when devices were intentionally rebooted, the monitoring system did not issue a theft alert, the device went offline and came back online within the configured timeframe, and the software correctly interpreted this as a normal event by adding it to a log but not sending an alert. This suggests that the fourth objective, *avoiding false alerts*, was fulfilled. It should be noted, however, that this aspect will be examined further in the discussion chapter under 7.3.1 Controlled Environment vs. Real-world Complexity.

Overall, the functional evaluation shows that the system achieved its primary goals, see table 2.

Table 2: Summary of Objective Fulfilment

Objective	**Fulfilled**	**Notes**
Detect unauthorised hardware removal	Partially	Detected only if removal causes system crash or loss of heartbeat.
Detect unexpected device disconnections	Fulfilled	100% detection rate in controlled tests.
Provide immediate alerts	Fulfilled	Alerts triggered within configured interval after event.
Avoid false alarms	Fulfilled	No false positives observed during controlled reboot tests; real-world validation needed.

7 Discussion

The evaluation results provide evidence that the network-driven theft detection approach is effective in a controlled setting. This chapter presents an analysis of these results in relation to the project's goals and the broader challenges outlined in 1 Introduction.

The following sections examine how effectively the system addresses the original problem of unnoticed hardware theft in shared environments and explore the implications of the findings for the solution's viability and limitations. The discussion critically reflects on the project by highlighting the strengths and advantages of the approach, acknowledging its shortcomings and remaining challenges, considering broader sustainability and societal implications, and addressing ethical and legal aspects of deploying such a system. Additionally, this chapter compares the solution to related work, identifies lessons learned, and outlines potential directions for future improvements, ensuring a cohesive reflection on the project's outcomes.

7.1 Fulfilment of Security Objectives

A primary question is whether the project succeeded in solving the problem it set out to address. Based on the evaluation, the system did achieve its core purpose within the test scope. The system's ability to detect missing hardware in near real-time directly targets the previously unmet need for immediate alerts when unauthorised removals occur. Tests showed that when critical components (such as the primary storage drive containing the operating system) were removed, the resulting system crash and loss of network connectivity reliably triggered an alert. This outcome validates the core concept of using a continuous network "heartbeat" to monitor device integrity: as soon as a heartbeat stops unexpectedly, the system flags a potential theft. In short, the central idea behind the solution was proven sound under the conditions tested.

However, it must be noted that the system's detection capability is dependent on the failure mode of the removal. If a stolen component does not cause a critical failure or network disconnection, the current design will not notice the theft. For example, if a secondary hard drive (one not hosting the OS) or another non-critical component is removed, the computer may continue operating normally and still send heartbeats. Consequently, such a theft could go unnoticed by the system in its present form. This limitation implies that while the system effectively detects thefts that immediately disrupt a device's functioning, it does not guarantee comprehensive detection of all types of hardware removal. In other words, the approach catches the most damaging incidents (where the machine crashes or goes offline) but could miss more subtle ones.

To mitigate this gap, additional mechanisms could be considered in future designs. For instance, integrating a physical chassis-intrusion sensor (which triggers when a computer's case is opened) would provide another layer of security independent of the device's operational state. Such a sensor could detect that someone has physically accessed the internals of the machine, even if the machine stays powered on and connected. By combining the network heartbeat monitoring with a chassis-open alert, the system would potentially be able to catch theft of components that do not immediately incapacitate the computer.

Of course, introducing hardware sensors or similar measures would add complexity and cost, and these factors would need to be weighed against the project's original goal of a low-overhead, easily deployable solution.

In summary, the project's solution fulfilled its primary security objective in the scenarios tested: it provided prompt detection of hardware theft events that caused a device to go offline. This confirms that a network-centric monitoring approach can indeed address the core problem it was designed for. At the same time, the reliance on a theft causing a system failure in order to be detected highlights an important limitation. Addressing this limitation (for example, through supplementary sensors or smarter detection of subtle hardware changes) is a key consideration for improving the system beyond this initial success.

7.2 Operational Usability and Interface Design

While the system's core functionality lies in its ability to detect and report device disappearance, the interface through which the administrators would interpret and respond to alerts plays a critical role in operational effectiveness. During the evaluation, the web-based dashboard proved highly usable: the ability to quickly filter devices by room, status, or time of last check-in, allowing for efficient situational awareness.

Features such as persistent logging, status colour-coding, and sorting by location enabled quick triage during test scenarios. These elements ensured that even when multiple alerts were active, the interface remained manageable and interpretable. This usability reduces cognitive load on administrators and contributes to faster decision-making.

Looking ahead, continued focus on interface clarity will be essential, especially if the system is deployed at scale. Additional improvements such as customisable alert views, or integration with communication platforms (e.g., Slack or Microsoft Teams) could further enhance responsiveness. In larger teams, role-based interface customisation might also support different operational needs, for example giving security staff and IT administrators tailored views of the same data.

In summary, while the backend enables theft detection, it is the interface that makes the alerts actionable. The system's value therefore depends not only on detection accuracy, but also on how clearly and efficiently the information is presented to human operators.

7.3 Limitations and Remaining Challenges

While the projects outcomes are positive, it is important to recognise the limitations of the current solution. Several challenges became evident that temper the results:

7.3.1 Controlled Environment vs. Real-world Complexity

So far, the system has only been evaluated in a tightly controlled environment. Real-world deployments will likely introduce far more variability and edge cases than those seen in the lab. For example, in a university computer lab or an office, devices might occasionally miss a heartbeat due to benign network issues (such as a brief loss of connectivity or a dropped packet) unrelated to theft. An entire lab could even go offline due to a power outage or a network switch reboot. Events like these would technically cause the system to register multiple devices "missing" heartbeats and could trigger alerts that, while technically correct (the devices did go offline), do not actually indicate any malicious activity. Therefore, it is necessary to be cautious when generalising the positive results from the controlled tests directly to production environments.

Another aspect of this challenge is the need to tune the system's monitoring parameters for real-world conditions. If the heartbeat interval is too short or the threshold for declaring a device "offline" is too sensitive, the system may generate nuisance alerts for minor, transient issues that resolve themselves (e.g., a momentary network hiccup). Conversely, if these settings are too lenient (for instance, a very long timeout before an alert), a legitimate theft could go undetected for an unacceptably long time. Balancing sensitivity and specificity in alert generation will be an ongoing challenge as the system moves to more complex environments. In practice, ensuring reliability outside the lab will require additional testing and calibration: the heartbeat frequency and offline thresholds might need adjustment to find an optimal middle ground that minimises false alarms without sacrificing detection speed. In summary, what worked well in a controlled setting will need careful refinement to handle the complexity of real-world deployment.

7.3.2 Authorised vs. Unauthorised Events

While the system is designed to raise alerts when a device goes offline unexpectedly, it does include a mechanism to account for authorised scenarios, in advance. Tracking can be disabled on a per-device basis. When tracking is turned off for a machine, the system suppresses alerts related to that device, allowing for legitimate downtime without triggering false alarms. This feature provides administrators with a way to avoid unnecessary notifications during planned operations, such as hardware upgrades or software maintenance.

However, this functionality depends on prior knowledge and manual intervention. For instance, if an administrator knows that a machine will be offline due to scheduled work, they must proactively disable tracking beforehand.This reliance on manual tracking management introduces the possibility of human error and administrative burden. If authorised actions are not consistently communicated and tracking is not disabled appropriately, the system could still produce a high number of avoidable alerts.

To reduce this burden and improve reliability, further enhancements could focus on automating or enriching the system's contextual awareness. One option would be to integrate the monitoring logic with external data sources, such as an asset management database or maintenance schedules that are uploaded ahead of time. This could automatically

suppress alerts for devices undergoing approved work. Similarly, linking with door access logs or user authentication records could help verify whether an authorised individual was present when a device was shut down or removed. In summary, while the current system does provide a way to avoid false alarms through manual tracking controls, enhancing its ability to automatically infer context from operational data would reduce the risk of unnecessary alerts and improve administrative efficiency.

7.3.3 Scalability and Performance

Although the prototype performed well with a handful of devices, scaling up to a much larger deployment presents additional challenges in both technical and operational dimensions. If the system were deployed, for instance, across an entire university campus with hundreds of lab computers, several scalability issues would need to be considered.

From a technical standpoint, a larger number of monitored devices means a higher volume of heartbeat messages and a greater processing load on the central server. Each individual heartbeat message is very lightweight, so network bandwidth consumption per device is negligible. However, when hundreds or thousands of devices are sending heartbeats, the cumulative network traffic could become non-trivial. Careful planning of network architecture might be required to handle this additional load; for example, segmenting the network or optimising the heartbeat protocol to prevent even minor congestion. Similarly, the server software must be efficient in handling many simultaneous connections and in updating the device status database in real time for a large pool of clients.

Another important aspect of scalability is alert management in a large environment. In a small lab, an alert or two at a time is easy for an administrator to handle. In a large deployment, however, a single event could generate a flood of alerts. Consider a widespread incident like a network switch failure or a power outage in a building: this could cause dozens of devices to drop offline at once. The system as designed would dutifully trigger an alert for each device, which could overwhelm the administrators or security personnel receiving the notifications. Handling such scenarios effectively would require smarter tools or features. For instance, the system could automatically group related alerts and identify a common root cause (e.g., "50 devices went offline in room X of house Y") so that administrators see one consolidated incident instead of 50 separate alarms. It might also be useful to prioritise alerts, so that if many alerts fire at once, those indicating critical assets or locations are flagged first. Additionally, as organisations often have IT staff working in shifts, the notification mechanism might need to ensure that only the on-duty staff are alerted for incidents (to avoid unnecessarily disturbing off-duty personnel). Implementing time-windowed alert routing or role-based filtering for notifications could improve the system's effectiveness in a large-team setting by sending the right alerts to the right people at the right time. These kinds of operational refinements become increasingly important as scale grows.

Crucially, feedback from stakeholders highlighted that isolated offline devices are not always critical, for instance, if a single machine goes down, the impact is often limited. However, if multiple devices in the same room go offline simultaneously, particularly during sensitive periods such as exams, the consequences can be significant. In such sce-

narios, the system's ability to provide a high-level view of room or area status becomes more important than flagging individual device failures. Supporting this type of situational awareness would enable administrators to act quickly when disruptions occur in critical spaces.

In summary, while no performance bottlenecks or overwhelming alert situations were encountered during prototype testing with a small number of machines, scaling the system to a larger, real-world environment will require deliberate planning and further development. The architecture may need enhancements to maintain responsiveness and reliability at scale. Potential improvements include optimising network usage, enabling the automatic correlation of multiple alerts (to reduce noise during large incidents), and refining the alert delivery process for team workflows. The current design provides a strong starting point for moderate-scale deployments, but ensuring the system remains effective and user-friendly at an enterprise scale is an open challenge that future work must address.

7.4 Future Work and Improvements

Given the limitations and challenges discussed above, there are clear directions for future work to strengthen and extend the system. The evaluation has shown promise, but further steps are needed to make the solution more robust and applicable in production environments. Several key areas for improvement and expansion are outlined below:

7.4.1 Real-world Pilot Deployments

A top priority is to test the system in a real operational environment over an extended period. To date, the evaluation has been conducted in a simulated lab setting; moving to an actual deployment (for example, in a university computer lab with daily student use) would provide invaluable feedback. A long-term field trial would reveal how the system behaves under normal conditions: How often do false alerts happen, if at all, when real users are involved? Are there any unexpected failure modes or user behaviours that the current design doesn't account for? Such a pilot deployment would also help determine the optimal heartbeat interval and offline timeout settings for a real environment, balancing responsiveness (catching thefts quickly) with reliability (not triggering on every minor glitch). Gathering data and feedback from IT staff during these trials would guide refinements. In summary, conducting real-world trials is an essential next step to validate the system's effectiveness outside the laboratory and to fine-tune it for practical use.

7.4.2 Enhanced Context Awareness and Intelligence

In parallel with field testing, improving the system's ability to make intelligent decisions is an important area of development. As noted, distinguishing between legitimate device removals or shutdowns and actual theft remains a challenge. Future iterations of the software should incorporate more context to reduce false alarms. One idea is integration with institutional IT management systems, for instance, integrating with a maintenance

schedule database so that the monitoring system is informed in advance if a particular machine is supposed to be taken down for service at a given time. If a device goes offline during an approved maintenance window, the system could either suppress the alert or label it differently (as an expected event rather than a theft). Another improvement would be to implement a secondary verification step when an alarm condition is met: if a device stops sending heartbeats, the system could perform a secondary check (e.g., by pinging the device) before declaring it missing. This could filter out transient issues. Additionally, exploring machine learning techniques could be valuable. For example, an anomaly detection model could be trained on the heartbeat patterns and known events from many devices; over time, it might learn to recognise complex patterns that distinguish a theft scenario from normal operational anomalies. Any such intelligence added to the system must be carefully evaluated to ensure it truly improves accuracy and does not introduce too much complexity or new failure modes. The goal of these enhancements is to make the system more "aware" of context so that it can autonomously handle the distinction between benign and suspicious events, thus improving reliability and reducing the burden on human operators.

7.4.3 Broadening Detection Scope

There are also opportunities to broaden the scope of what the system monitors, extending its capabilities beyond the current network-heartbeat mechanism. At present, the system focuses on detecting a device's absence from the network, which is one indicator of theft or removal. In the future, the system could be expanded to watch additional signals of tampering or misuse. For instance, hardware-based sensors could be added to detect if the chassis of a computer has been opened or if a component has been physically removed or changed. If such sensors were integrated, the system would then have multiple channels of detection: a heartbeat loss and a chassis-open event together would provide very strong evidence of a theft in progress. Furthermore, integration with other security infrastructure could amplify the system's effectiveness. For example, when an alert is triggered, the system could automatically cross-reference other data sources, such as electronic door access logs, for the same time frame. Although these ideas (chassis sensors, motion detectors, etc.) extend beyond the current project's implemented scope, they sketch a path by which this work could evolve into a more comprehensive theft detection and prevention solution.

7.4.4 Lessons Learned from Development

Throughout the project, valuable insights were gained that can guide these future improvements. One key lesson was the value of simplicity in the monitoring approach. Early in development, a prototype of an active network scanning method (using a tool like Nmap) was explored and even implemented to keep track of devices on the network. Partway through, it became clear that this active scanning introduced unnecessary complexity, added network overhead, and raised potential privacy issues (since scanning can reveal information about devices that might not be relevant to theft detection). It was realised that a simpler solution, having the devices periodically report their presence (the heartbeat), was sufficient to fulfill the requirements and at the same time avoid those

downsides. This shift to a passive heartbeat approach proved very successful. The takeaway is that focusing on the core requirement (in this case, detecting the absence of a device) and avoiding over-engineering leads to a cleaner, more robust, and more privacy-friendly design. Another key insight was the importance of a well-designed user interface for the system's administrators. A significant effort in the project went into designing an intuitive dashboard and notification mechanism. During evaluation, this paid off: administrators found the interface easy to use and could quickly interpret device statuses and respond to alerts. This experience underscores that future work should continue to emphasise not just what the system can detect, but also how that information is presented to and handled by human users. The best technical solution can be undermined if the people using it find it confusing or cumbersome, so usability must remain a priority.

7.4.5 Opportunities for Automation and Scalability

Real-world testing revealed opportunities for enhancing system scalability and automation. Notably, the devices in the ED4225 lab at Chalmers used semantically meaningful hostnames (e.g., ED4225-01), which encode room and machine identifiers. Since the current registration process is primarily used to associate devices with metadata such as room and building, future versions of the system could extract this information automatically from hostnames. This would significantly streamline the onboarding process, reduce administrative effort, and make the system more scalable in larger environments.

7.5 Broader Impact and Sustainability

The prototype developed in this project supports both social and economical sustainability in organisations that rely on shared computing resources. The quick detection of hardware removal helps to protect valuable equipment and avoid disruptions to operations.

A missing computer during an exam or a critical project can be the cause for significant delays in educational settings. This prototype allow administrators to respond quickly before such incidents escalate. That in turn improves reliability in environments where scheduled access to devices is critical.

Economically, the prototype reduces the need for replacements, insurance claims, and follow-up administration. Reducing theft helps to preserve budgets and opens up resources for improvements rather than recovery.

The impact of this prototype goes beyond just physical loss. As Dertouzos *et al.* note, hardware that are stolen often contain sensitive data, posing risks of data breaches, intellectual property loss, and compliance violations, [32]. These consequences can be severe for institutions, which makes early detection an important safeguard.

7.6 Ethical and Social Aspects

While the earlier sections of this discussion considered the system's impact and sustainability in broad terms, it is also essential to focus on the ethical and legal considerations specific to implementing a network-based monitoring system. Any solution that involves monitoring must be evaluated not just on technical merit, but on how it aligns with privacy rights, user trust, and regulatory requirements. This section examines how these issues were addressed.

7.6.1 Privacy and Data Scope

To address privacy concerns, the scope of data that the system collects and stores was deliberately limited. The monitoring system does not capture any personal user data, such as login identities, usage logs, keystrokes, or content accessed. Instead, it confines itself to a few technical metrics that are necessary for its function: for each device, it keeps track of the device's MAC address (the hardware network identifier), the last time the device checked in (heartbeat timestamp), and the device's label or location identifier in the lab. No information is gathered about who is using the device or what they are doing on it. By keeping the data scope this narrow, it was possible to significantly reduce the privacy impact. In practice, the data says, for example, "Computer #5 in Lab X was last seen at 12:34", which is a security-relevant fact but not personally sensitive information.

7.6.2 Legal Compliance and Policy Alignment

Given that the system only monitors institutional devices that have the heartbeat client installed, legal and privacy concerns are relatively limited. The data collected consists solely of technical identifiers, such as MAC addresses, and timestamps indicating the last check-in. These are not linked to individual users and do not contain any personal or behavioural data. The system does not monitor personal devices on the network, nor does it perform any active scanning of unknown endpoints.

Nonetheless, care was taken to ensure that the design aligns with ethical standards and institutional policies. Early in the project, the option of using active network scanning (e.g., via Nmap) was considered, but deliberately avoided. Active scanning can unintentionally detect personal or unauthorised devices on the network and may raise unnecessary privacy concerns. Instead, a passive heartbeat-based approach was chosen, where only approved devices voluntarily report their presence. This decision reflects a principle of data minimisation: only the information necessary for theft detection is collected, and only from devices known to the organisation.

While the collected data is not sensitive in itself, it should still be treated as confidential operational data within the institution's IT security framework. Access to the monitoring interface and logs should be limited to authorised personnel, and retention policies should follow the organisation's standard practices for security logs. However, there is no immediate legal obligation to delete or anonymise MAC addresses used solely for operational monitoring of institutional equipment.

In summary, the system was designed with a narrow, well-defined purpose that avoids unnecessary data collection. This reduces the potential for legal complications and ensures that the solution can be deployed responsibly within existing policy frameworks, without infringing on personal privacy or triggering regulatory concerns.

7.7 Critical Distinctions from Related Work

The landscape of device monitoring solutions is diverse, but most existing approaches share assumptions and design priorities that differ from the needs of semi-supervised environments. This section highlights fundamental differences between conventional methods and the proposed system, focusing on detection strategy, infrastructural demands, privacy implications, and functional applicability.

7.7.1 Coverage Gaps in Conventional Monitoring Systems

Most conventional network monitoring tools - such as InterMapper, OP5, or IBM QRadar - are designed to ensure service uptime, detect network anomalies, or aggregate security logs. These tools offer substantial value in enterprise IT contexts but are not optimised for detecting hardware-level events like unauthorised physical device removal.

Unlike these platforms, the proposed system interprets absence of presence - not just lack of connectivity - as a potential anomaly. This provides a subtle but critical distinction within a networked environment. This tailored perspective is essential in settings like academic labs or shared classrooms, where devices are more exposed and physically accessible.

7.7.2 Post-Incident vs. Pre-Incident Security Approaches

One important distinction concerns the timing and operational philosophy of theft protection mechanisms: whether they are reactive (post-incident) or proactive (pre-incident). Reactive measures include, for example, services that attempt to track a device's location after it reconnects to the internet, or systems that identify stolen devices when they appear on a network. While such approaches can be useful for recovery, they offer limited support for immediate intervention. Their effectiveness is also increasingly undermined by modern privacy features—such as MAC address randomisation— which hinder traditional tracking methods.

By contrast, the approach taken in this project is distinctly pre-incident and proactive. By monitoring heartbeat signals in real time and issuing alerts the moment a device fails to check in, the system aims to detect theft as it occurs—or within minutes of the event. This enables an early response, allowing security personnel to act immediately, potentially intercepting the perpetrator or preventing further loss. The system's design effectively turns device disappearance into an immediate alarm trigger, complementing traditional asset recovery tools with a preventive layer.

Rather than relying on a stolen device to eventually reappear online, the system focuses on detecting the incident at the moment it happens. This real-time alerting represents a significant shift from the conventional approach of "discover later and try to recover", towards a strategy of "detect now and intervene". In environments where equipment is critical or rapid intervention is necessary, such as a lab before an exam, this proactive approach offers a clear advantage.

7.7.3 Reducing Infrastructure and Privacy Barriers

As described in Section 2.3, physical tamper detection systems such as NFC-based tracking, Anti-Tamper Radio (ATR), or camera-equipped solutions can offer high physical specificity. However, this often comes at the cost of increased infrastructural complexity, privacy concerns, and reduced cost efficiency. These approaches are typically suitable for enclosed or high-security environments, but become impractical in semi-public computing spaces with limited resources.

The strength of the proposed system lies in its ability to avoid these trade-offs. By relying on heartbeat-based monitoring, it reduce the need of invasive hardware or visual tracking and offers a practical, scalable alternative where institutions lack resources- or legal confidence - to deploy intrusive monitoring.

7.7.4 Functional Fit for Semi-Supervised Spaces

Many related systems address either general network security or highly controlled environments. The proposed solution fills a structural gap: monitoring devices in semi-supervised, moderately secure settings such as university campuses, digital labs, or co-working spaces.

The proposed solution does not replace advanced cybersecurity systems or physical tracking infrastructure. Instead, it complements them by extending device monitoring to an overlooked domain - locations where theft risks exist, but full-scale protection is infeasible.

In this way, the system introduces a new functional layer, providing lightweight behavioural monitoring without user profiling, deep packet inspection, or invasive surveillance mechanisms.

7.8 Chapter Summary

This chapter provides a concluding discussion of the developed prototype. It covers how well the prototype objectives were met and reflects on its performance in a controlled environment, highlighting both strengths and areas for future development. Limitations and challenges are identified, including scalability, real-world applicability, and event interpretation. Potential improvements were also outlined to enhance the prototype's robustness and usability in broader contexts, such as automation, contextual awareness, and system integration. Ethical and legal considerations are discussed, along with a comparison to existing solutions and the environment where the prototype may offer particular value.

8 Conclusion

A network-based monitoring solution for detecting unauthorised device removal was developed and evaluated in this project. The proposed system monitors the "heartbeat" of connected devices and automatically triggers alerts when a device goes offline unexpectedly, indicating a potential theft or disconnection. Testing of the prototype in a simulated environment showed that it can promptly and reliably detect missing devices without requiring specialised hardware or intrusive scanning of the network. The system thereby achieves its core objective of enabling timely detection of possible equipment theft using only standard network communications.

In addition to core functionality, the solution features an intuitive administrative dashboard and a role-based notification mechanism to support effective incident management. The interface proved easy for administrators to use, allowing them to quickly interpret device statuses and respond to alerts. The design also emphasises privacy and compliance by avoiding the collection of personal data, aligning the system with relevant legal and ethical requirements for deployment. Comparisons with conventional monitoring approaches suggest that this proactive method can fill important gaps in physical IT security — providing real-time alerts for device disappearance where traditional security measures might only detect incidents after the fact.

While the results are promising, the work also highlighted areas for future improvement. In particular, conducting extensive field trials in real operational environments is recommended to validate the system's performance under diverse conditions and to gather feedback from a broader user base. Further enhancements, such as integrating supplementary sensors or refining the alert logic to reduce false positives, could strengthen the system's capabilities. Overall, the project demonstrates a feasible and effective approach to bolstering physical device security in networked environments and lays a strong foundation for continued development and integration into institutional security strategies.

References

[1] Devoteam. *Cybersecurity: The Importance of Physical Security*. 2025. URL: https://www.devoteam.com/expert-view/cybersecurity-the-importance-of-physical-security/ (visited on 02/10/2025).

[2] Cisco. *What is cybersecurity?* 2025. URL: https://www.cisco.com/site/us/en/learn/topics/security/what-is-cybersecurity.html (visited on 02/25/2025).

[3] Yu Han, Zhiliang Chen, and Tinghang Guo. "Design of equipment anti-theft tracker based on wireless sensor network". In: *2017 First International Conference on Electronics Instrumentation & Information Systems (EIIS)*. 2017, pp. 1–5. DOI: 10.1109/EIIS.2017.8298681.

[4] Fortra. *InterMapper User Guide*. 2023. URL: https://ressources.intermapper.eu/docs/InterMapper_User_Guide.pdf (visited on 02/05/2025).

[5] Marin Vidaković and Davor Vinko. "Hardware-Based Methods for Electronic Device Protection Against Invasive and Non-Invasive Attacks". In: *Electronics* 12.21 (Nov. 2023), p. 4507. DOI: 10.3390/electronics12214507.

[6] J. Kim and J. Denk. "The Illusion of One Hundred Percent Uptime". In: *Proceedings of the SIGUCCS Annual Conference*. ACM, 2012.

[7] IBM. *IBM QRadar SIEM User Guide*. 2025. URL: https://www.ibm.com/docs/en/SS42VS_7.5/pdf/b_qradar_users_guide.pdf (visited on 02/28/2025).

[8] ScienceSoft. *SIEM Solution for 70 US State Agencies: Case Study*. 2025. URL: https://www.scnsoft.com/case-studies/siem-solution-for-70-us-state-agencies (visited on 04/25/2025).

[9] ITRS Group. *OP5 Monitor – Network Monitoring Product Page*. 2025. URL: https://www.itrsgroup.com/products/network-monitoring-op5-monitor (visited on 04/25/2025).

[10] Kartik Mandaville et al. "Theft detection of computers using MAC address by map-reduce programming model on a cluster". In: *2012 International Conference on Recent Advances in Computing and Software Systems*. 2012, pp. 250–253. DOI: 10.1109/RACSS.2012.6212676.

[11] Xi Yu. *University Uses Device Registration Information to Track Down Stolen Devices. The Harvard Crimson*. May 2011. URL: https://www.thecrimson.com/article/2011/5/26/mac-network-harvard-address/ (visited on 04/25/2025).

[12] Mario Alberto Flores-Cruz et al. "Computer Device Theft Detection System". In: *ECORFAN Proceedings: Young Researchers in Engineering and Biotechnology for Society* (2024), pp. 34–48. DOI: 10.35429/P.2024.1.34.48. URL: https://www.ecorfan.org/proceedings/Young_researchers_Engineering_and_Biotechnology_for_Society/Engineering_and_Biotechnology_for_Society_Proceedings_TII_4.pdf (visited on 04/25/2025).

[13] Paul Staat et al. "Anti-tamper radio: System-level tamper detection for computing systems". In: *Proceedings of the 2022 IEEE Symposium on Security and Privacy (SP)*. 2022, pp. 1722–1736. DOI: 10.1109/SP46214.2022.9833631.

[14] Ajay Kushwaha et al. "Theft-Detection using Motion Sensing Camera". In: *International Journal of Innovative Science and Research Technology* 2.11 (2017), pp. 2456–2165.

[15] Dakun Shen et al. “Virtual Safe: Unauthorized Walking Behavior Detection for Mobile Devices”. In: *IEEE Transactions on Mobile Computing* 18.3 (2019), pp. 688–701. DOI: 10.1109/TMC.2018.2843801.

[16] Fyodor (Gordon Lyon). *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. 2009. URL: https://nmap.org/book/man-performance.html (visited on 02/18/2025).

[17] David C. Plummer. *An Ethernet Address Resolution Protocol*. Request for Comments: 826. Nov. 1982. URL: https://datatracker.ietf.org/doc/html/rfc826 (visited on 02/18/2025).

[18] Gordon Fyodor Lyon. “Nmap: The Network Mapper”. In: *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Elsevier, 2008. ISBN: 978-1-59749-241-6. DOI: 10.1016/B978-1-59749-241-6.X0001-5.

[19] Microsoft Docs. *Security Contexts for Running Tasks - Task Scheduler*. 2023. URL: https://learn.microsoft.com/en-us/windows/win32/taskschd/security-contexts-for-running-tasks (visited on 04/08/2025).

[20] The Go Authors. *Go Standard Library Documentation*. URL: https://pkg.go.dev/std (visited on 04/08/2025).

[21] Jennifer Preece, Helen Sharp, and Yvonne Rogers. *Interaction Design Beyond Human Computer Interaction*. 5th ed. John Wiley & Sons, Inc, 2019.

[22] J. Finkelstein. *Needs Before Wants in User Experiences – Maslow and the Hierarchy of Needs*. Accessed 2025-04-27. 2016. URL: https://www.interaction-design.org/literature/article/needs-before-wants-in-user-experiences-maslow-and-the-hierarchy-of-needs.

[23] Interaction Design Foundation. *Self-Actualization: Maslow's Hierarchy of Needs*. Accessed: 2025-04-28. URL: https://www.interaction-design.org/literature/article/self-actualization-maslow-s-hierarchy-of-needs.

[24] Usability Partners. *ISO Standards*. Accessed: 2025-04-28. URL: https://www.usabilitypartners.se/about-usability/iso-standards.

[25] WebAIM. *Contrast Checker*. Accessed: 2025-05-02. URL: https://webaim.org/resources/contrastchecker/.

[26] I. Scott MacKenzie. “Fitts' Law as a Research and Design Tool in Human-Computer Interaction”. In: *Human–Computer Interaction* (1992). Accessed: 2025-04-28, pp. 91–139. URL: https://www.yorku.ca/mack/hci1992.pdf.

[27] Jenifer Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. 3rd. O'Reilly Media, 2020. ISBN: 9781492051961.

[28] T. J. Brigham. “Feast for the Eyes: An Introduction to Data Visualization”. In: *Medical Reference Services Quarterly* 35.2 (2016), pp. 215–223. DOI: 10.1080/02763869.2016.1152146.

[29] Alan Cooper et al. “About Face: The Essentials of Interaction Design”. In: Fourth. John Wiley & Sons, Inc., 2014.

[30] Sohrab Almasi et al. “Usability evaluation of Dashboards: A systematic literature review of Tools”. In: *BioMed Research International* 2023.1 (2023), p. 11. DOI: 10.1155/2023/9990933.

[31] Ivan Shubin. *Anomaly detection in time series using statistical analysis*. Apr. 2025. URL: https://medium.com/booking-com-development/anomaly-detection-in-time-series-using-statistical-analysis-cc587b21d008.

[32] James N. Dertouzos, Eric V. Larson, and Patricia A. Ebener. *The Economic Costs and Implications of High-Technology Hardware Theft*. Santa Monica, CA: Rand Corporation, 1999.

Appendix A: GUI Design Iterations and Prototypes

A.1 Statistics Page

Figure A.1: Overview of statistics page.

A.2 Iteration 1: Early Prototype Layout

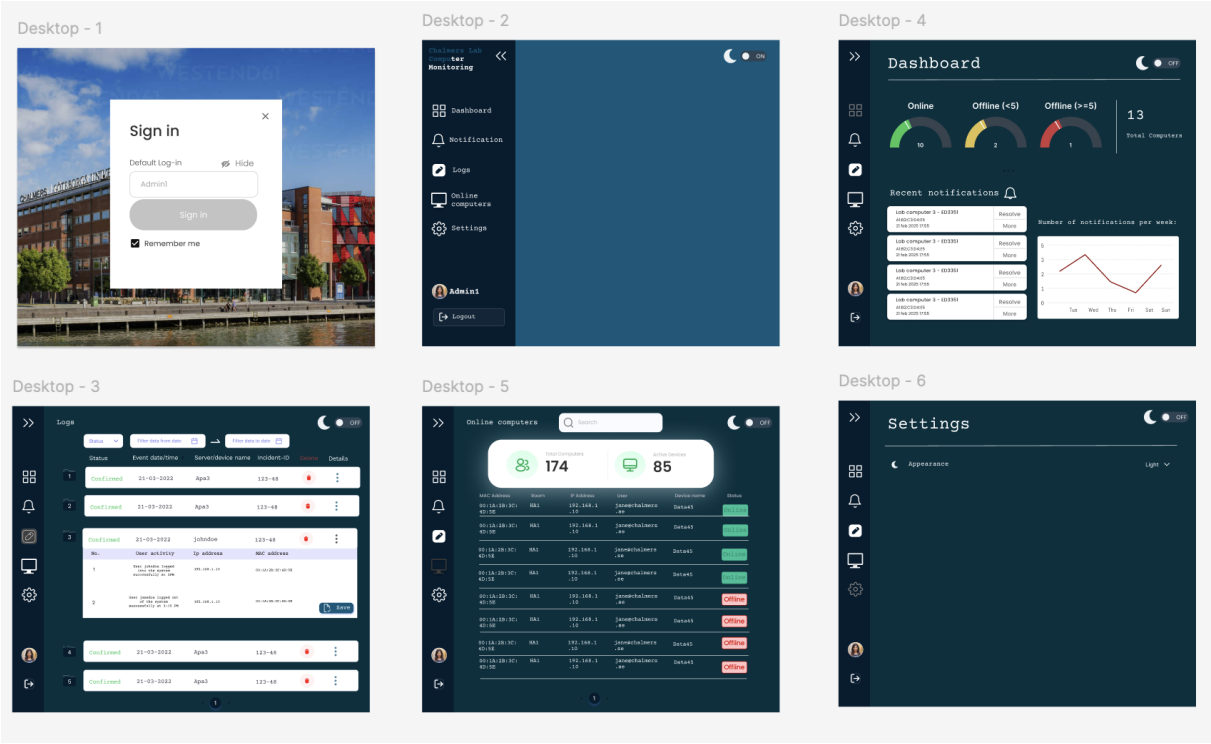


Figure A.2: An early prototype of the monitoring system's user interface, designed in Figma. This prototype includes key sections such as the login screen, dashboard, logs, online device status, and system settings, aimed at providing administrators with an intuitive overview and control of the system.

A.3 Iteration 2: Prototype refinement

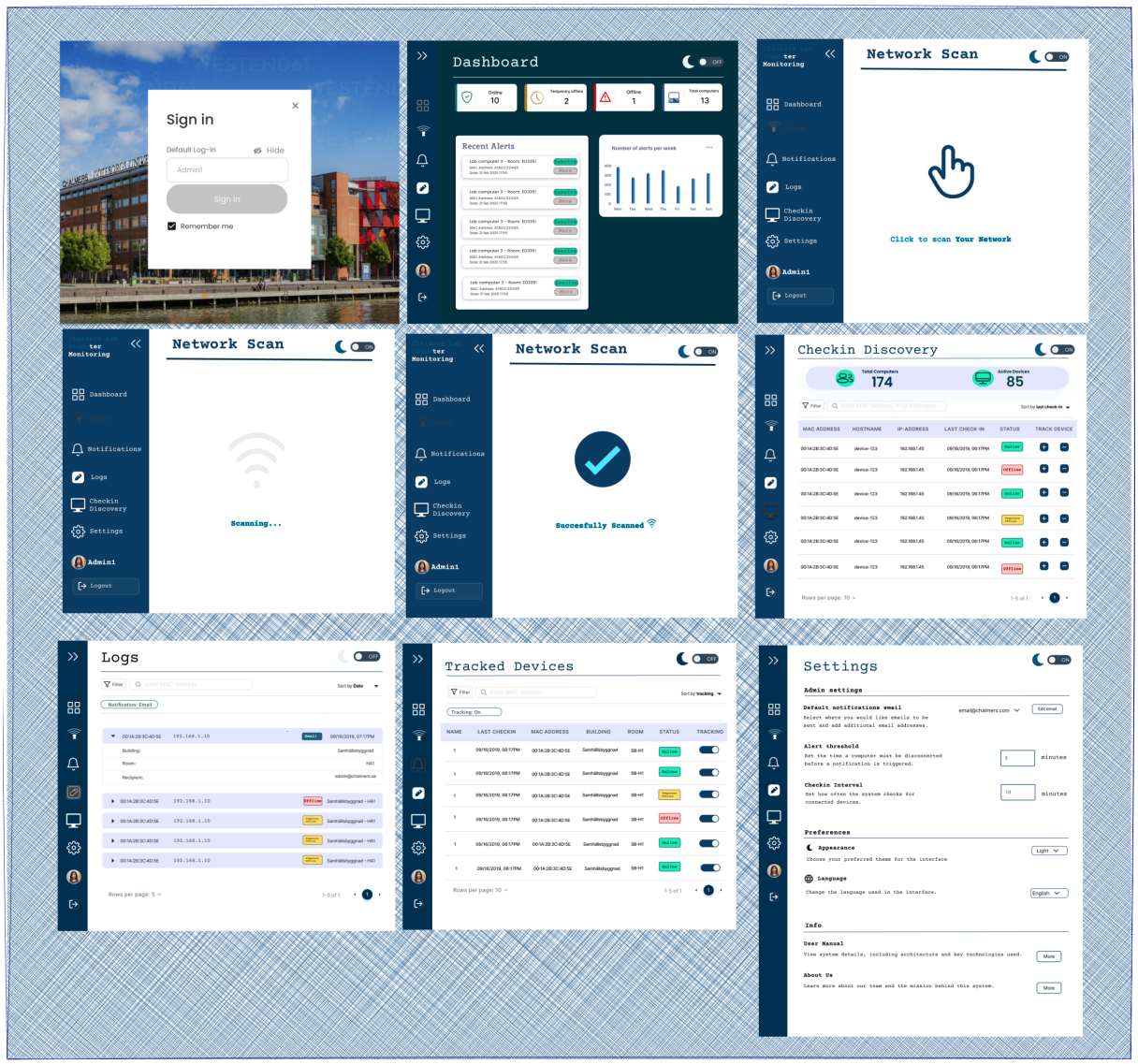


Figure A.3: Overview of prototype screens developed during Iteration 2.

A.4 Interaction Flow Diagram

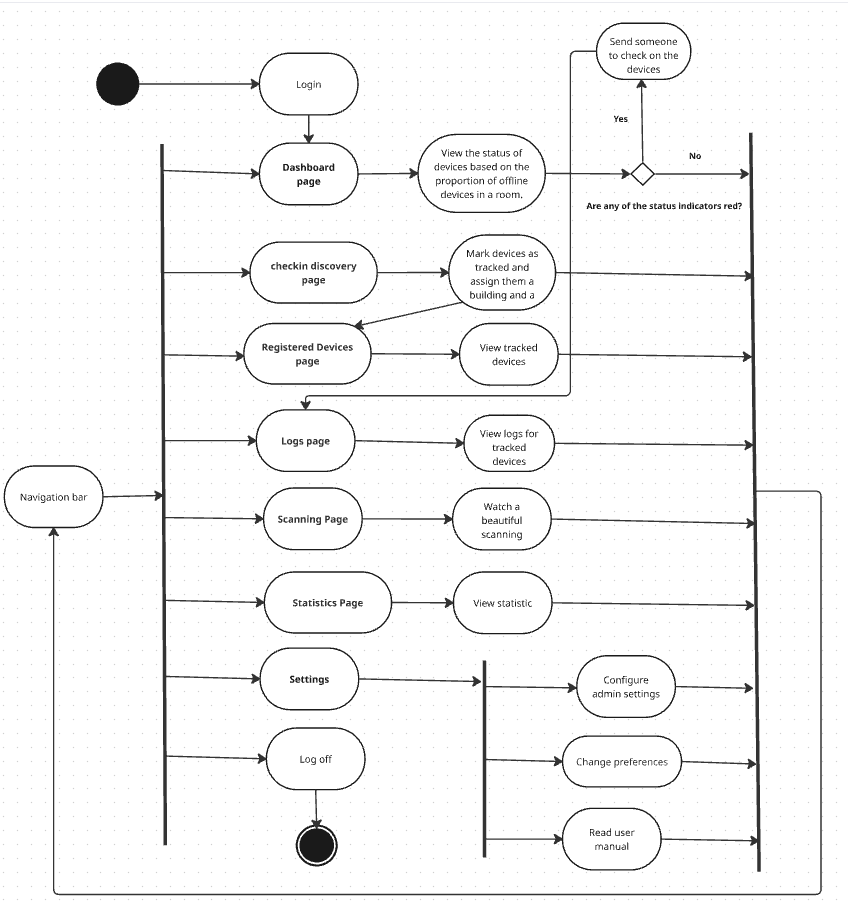


Figure A.4: Flowchart illustrating the user journey through the monitoring system. The diagram shows how users navigate core pages, such as the dashboard, logs, check-in discovery and settings, in order to assess system status, respond to incidents and adjust configurations.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY