

# Health Checker

Projektdokumentation

Studiengang Systems Engineering (B. Eng.)

Schwerpunkt I.2

## Verfasser/innen:

Jan-Niclas Fenger  
Matrikelnummer: 2092175  
E-Mail: jan-niclas.fenger@tha.de

Leopold Weber  
Matrikelnummer: 2155780  
E-Mail: leopold.weber@tha.de

Simon Schneider  
Matrikelnummer: 2151357  
E-Mail: simon.schneider1@tha.de

Janis Preiß  
Matrikelnummer: 2150826  
E-Mail: janis.preiss@tha.de

## Betreuer/in:

Prof. Dr. Volodymyr Brovkov

2. Februar 2025

# Inhaltsverzeichnis

<b>1</b>	<b>Zusammenfassung</b>	<b>4</b>
<b>2</b>	<b>Einleitung</b>	<b>4</b>
2.1	Kontext/Hintergrund . . . . .	4
2.2	Projekt Motivation . . . . .	4
2.3	Problemstellung . . . . .	4
2.4	Projektziele . . . . .	4
2.5	Projektumfang . . . . .	4
2.6	Übersicht über die Dokumentation . . . . .	4
<b>3</b>	<b>Methodik</b>	<b>4</b>
3.1	Prinzipienschaltbild . . . . .	4
3.2	Die Wahl der Hardware . . . . .	5
3.2.1	Raspberry-Pi . . . . .	5
3.2.2	Display . . . . .	5
3.2.3	RFID-Reader . . . . .	5
3.2.4	Infrarot-Temperatursensor . . . . .	5
3.2.5	Piezo-Buzzer . . . . .	5
3.2.6	Ultraschallsensor . . . . .	6
3.2.7	Wemos ESP32 D1 Mini . . . . .	6
3.2.8	Lichtschranke . . . . .	6
3.2.9	Steppermotor . . . . .	6
3.2.10	WEB-Datenbank-Server . . . . .	6
3.2.11	Switch . . . . .	6
3.2.12	Zusammenfassung . . . . .	6
3.3	Grundlegender Aufbau . . . . .	7
3.3.1	Allgemein . . . . .	7
3.3.2	Raspberry-Pi . . . . .	7
3.3.3	Wemos-Modul . . . . .	7
3.3.4	Web-Datenbank-Server . . . . .	7
3.3.5	Zusammenfassung . . . . .	7
3.4	Bestimmung der Messwerte . . . . .	8
3.4.1	Bestimmung der Körpertemperatur . . . . .	8
3.4.2	Bestimmung des Abstandes (Näherungssensor) . . . . .	8
<b>4</b>	<b>Software und User Interface</b>	<b>8</b>
4.1	Programmstruktur . . . . .	8
4.2	Datenbanksystem . . . . .	9
4.2.1	Einführung . . . . .	9
4.2.2	Problemstellung bei SQLite3 . . . . .	9
4.2.3	Lösung: Umstellung auf PostgreSQL . . . . .	9
4.2.4	Schematischer Aufbau . . . . .	9
4.2.5	Zusammenfassung . . . . .	9
4.3	Technologien und Architektur (Webanwendung) . . . . .	10
4.4	Technologien und Architektur (Benutzerterminal) . . . . .	10

4.5	Benutzerinterface und Funktionalitäten (Webanwendung) . . . . .	11
4.6	Benutzerinterface und Funktionalitäten (Benutzerterminal) . . . . .	12
4.7	API-Dokumentation . . . . .	12
4.7.1	Allgemeines . . . . .	13
4.7.2	Benutzerverwaltung . . . . .	13
4.7.3	Datenbankverwaltung . . . . .	14
4.7.4	Mitarbeiterverwaltung . . . . .	14
<b>5</b>	<b>Ergebnisse und Diskussion</b>	<b>14</b>
5.1	Messwerte & Diskussion . . . . .	15
5.1.1	Datenüberblick . . . . .	15
5.1.2	Analyse der Messgrößen . . . . .	15
5.1.3	Herausforderungen & Lösungsansätze . . . . .	15
5.2	Abnahmekriterien . . . . .	15
5.3	Diskussion der Ergebnisse . . . . .	15
<b>6</b>	<b>Fazit und Ausblick</b>	<b>16</b>
6.1	Zusammenfassung der Arbeit . . . . .	16
6.2	Diskussion der Implikationen . . . . .	16
6.3	Stärken des Systems . . . . .	17
6.4	Herausforderungen & Verbesserungsmöglichkeiten . . . . .	17
6.5	Zukunftsperspektiven . . . . .	18
	<b>Literatur</b>	<b>19</b>

# 1 Zusammenfassung

Jan

## 2 Einleitung

Jan

### 2.1 Kontext/Hintergrund

### 2.2 Projekt Motivation

### 2.3 Problemstellung

### 2.4 Projektziele

### 2.5 Projektumfang

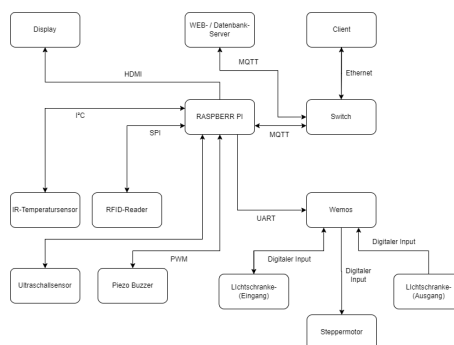
### 2.6 Übersicht über die Dokumentation

## 3 Methodik

Simon

### 3.1 Prinzipschaltbild

Der prinzipielle Aufbau des Projekts wurde zuerst erstellt und in einem Blockdiagramm dargestellt, das die verschiedenen Komponenten und deren Verbindungen veranschaulicht. Dieses Diagramm zeigt den Fluss des Zugangskontrollsystems von Authentifizierung und Temperaturmessung über das Schreiben in die Datenbank bis hin zur Ansteuerung der Drehkreuze und der Lichtschranken.



Bildgröße  
am Ende  
anpassen

Abbildung 1: Schematischer Aufbau des Projekts

## **3.2 Die Wahl der Hardware**

### **3.2.1 Raspberry-Pi**

Der Raspberry Pi verfügt bereits über eine Vielzahl an Schnittstellen, die eine einfache Integration von Sensoren, Aktoren und weiteren Peripheriegeräten ermöglicht. Durch vorhandenes Wissen und umfangreiche Dokumentationen kann die Umsetzung der Anforderungen effizient erfolgen, ohne dass eine aufwendige Einarbeitung in neue Hardware notwendig ist. Zudem stellt der Raspberry Pi eine kostengünstige und vielseitige Plattform dar, die eine schnelle Entwicklung, Erprobung und iterative Optimierung technischer Lösungen unterstützt.

### **3.2.2 Display**

Das Display wird verwendet, um dem Nutzer visuell relevante Informationen wie den aktuellen Status des Systems, Zugangsberechtigungen oder Fehlermeldungen bereitzustellen. Durch die grafische Darstellung kann der Benutzer schnell und intuitiv auf wichtige Systeminformationen zugreifen, was die Bedienfreundlichkeit und Effizienz des Systems erhöht. Das Display sorgt dafür, dass der Nutzer klare Informationen über das System erhält. Unter anderem mit Farben oder Symbolen wird angezeigt, ob der Zugang erlaubt oder verweigert wird. So bekommt der Nutzer direktes Feedback und Missverständnisse werden vermieden.

### **3.2.3 RFID-Reader**

Der RFID-Reader wird im Zugangssystem zur Authentifizierung verwendet, indem er die individuellen Daten von RFID-Tags ausliest, die jedem Mitarbeiter zugeordnet sind. Jeder Tag enthält eine einzigartige ID, die mit einer gespeicherten Liste von Berechtigungen abgeglichen wird, um zu entscheiden, ob der Zugang gewährt oder verweigert wird. Diese personalisierte, berührungslose Authentifizierung bietet eine schnelle und sichere Möglichkeit, den Zugang für berechtigte Personen zu steuern.

### **3.2.4 Infrarot-Temperatursensor**

Ein Infrarot-Temperatursensor misst schnell und kontaktlos die Körpertemperatur eines Mitarbeiters. Liegt die Temperatur über einem festgelegten Schwellenwert, wird der Mitarbeiter als potenziell krank eingestuft. Diese Methode ist hygienisch, schnell und verhindert die Verbreitung von Krankheiten im Arbeitsumfeld.

### **3.2.5 Piezo-Buzzer**

Der Piezo-Buzzer wird verwendet, um dem Nutzer akustisches Feedback zu geben, indem er bei bestimmten Ereignissen wie dem erfolgreichen oder fehlgeschlagenen Zugang ein Signal abgibt. Durch die Erzeugung von Tönen oder Melodien kann der Benutzer sofort erkennen, ob eine Aktion erfolgreich war oder eine Fehlermeldung vorliegt. Der Buzzer sorgt für eine klare, hörbare Rückmeldung, die das visuelle Feedback des Systems ergänzt und so die Benutzererfahrung verbessert, indem er Missverständnisse verhindert und sofortige Reaktionen ermöglicht.

### **3.2.6 Ultraschallsensor**

Der Ultraschallsensor erkennt die Annäherung eines Mitarbeiters und aktiviert automatisch das Display. So wird das System effizient und benutzerfreundlich ohne manuelles Eingreifen aktiviert.

### **3.2.7 Wemos ESP32 D1 Mini**

Der Wemos D1 Mini ESP32 wurde aufgrund seiner kompakten Größe und ausreichenden GPIO-Pins für die Steuerung der Lichtschranken und des Stepper Motors gewählt. Da er über UART an einen Raspberry Pi angeschlossen wird, ermöglicht der Wemos eine einfache und effiziente Kommunikation zwischen den Komponenten bei gleichzeitig platzsparender Bauweise.

### **3.2.8 Lichtschranke**

Die Lichtschranke wird eingesetzt, um zu erkennen, wenn jemand versucht, durch das Drehkreuz zu gehen. Sobald der Infrarotstrahl unterbrochen wird, signalisiert die Lichtschranke dem System, dass sich eine Person im Drehkreuzbereich befindet, und löst daraufhin die entsprechende Aktion aus, wie zum Beispiel das Öffnen des Drehkreuzes. Diese Technologie ermöglicht eine präzise Erkennung der Passage und sorgt für eine automatische, kontaktlose Steuerung des Zugangs.

### **3.2.9 Stepper Motor**

Der Stepper Motor wird verwendet, um ein Drehkreuz zu simulieren, indem er präzise Drehbewegungen ausführt. Durch die Steuerung des Motors kann das Drehkreuz geöffnet oder geschlossen werden, je nachdem, ob ein Mitarbeiter Zugang erhält. Diese Methode ermöglicht eine exakte und zuverlässige Steuerung der Drehmechanik, was für den reibungslosen Ablauf des Zugangssystems entscheidend ist.

### **3.2.10 WEB-Datenbank-Server**

Ein Web- und Datenbankserver wird für die Steuerung, Überwachung und Speicherung des Zugangskontrollsystems genutzt. Er verwaltet Benutzerzugänge, speichert Transaktionsprotokolle und ermöglicht eine zentrale Überwachung und Verwaltung des Systems.

### **3.2.11 Switch**

Für die Netzkommunikation wird ein Switch eingesetzt, um die Verbindung zwischen den verschiedenen Geräten im System zu ermöglichen. Der Switch sorgt für eine effiziente und zuverlässige Datenübertragung, indem er die Geräte im Netzwerk miteinander verbindet und den Datenverkehr optimiert.

### **3.2.12 Zusammenfassung**

Die ausgewählten Komponenten bieten eine effiziente und zuverlässige Lösung für das Zugangskontrollsystem. Sie ermöglichen eine benutzerfreundliche Interaktion,

präzise Erfassung von Daten und Sicherheitsüberprüfungen. Ein zentraler Server verwaltet die Daten, während eine stabile Netzwerkverbindung die Kommunikation zwischen den Geräten sicherstellt. So wird eine zuverlässige Steuerung und Überwachung des Zugangsprozesses gewährleistet.

### **3.3 Grundlegender Aufbau**

#### **3.3.1 Allgemein**

Anhand der Anforderung an das System wurde sich für ein verteiltes, netzwerkgebundenes System entschieden. Hier wurden drei Teilbereiche definiert, auf die im Weiteren genauer eingegangen wird.

#### **3.3.2 Raspberry-Pi**

Den Startpunkt des Zugangssystems stellt der Raspberry-Pi mit Display und seinen Komponenten (siehe 3.1 Prinzipschaltbild) bereit. Hierbei authentifiziert sich der Mitarbeiter mit seinem RFID-Tag über einen entsprechenden RFID-Reader. Der Abgleich findet über MQTT mit dem Datenbankserver statt. Nach erfolgreicher Validierung der Identität wird die Körpertemperatur kontaktlos über einen Infrarot-Temperatursensor gemessen. Der Abgleich findet wie bei der Authentifizierung statt. Neben visuellen Darstellungen auf dem Display gibt es auch akustische Signale von einem Piezo Buzzer. Da es auch weniger frequentierte Zeiten am Terminal gibt, kann es auch vorkommen, dass das Terminal im Ruhemodus sich befindet. Um dies wieder zu aktivieren wird ein Näherungssensor eingesetzt.

#### **3.3.3 Wemos-Modul**

Nach erfolgreicher Authentifizierung und Temperaturmessung wird die Eingangslightschranke für eine gewisse Zeit aktiviert, welche bei nicht durchschreiten deaktiviert wird. Wird aber die Lichtschranke durchbrochen, so dreht sich der Steppermotor, auf dem ein Drehkreuz montiert wurde. Am Ausgang ist ebenso ein Lichtschranke montiert, welche dauerhaft aktiv ist. Diese sorgt erwartungsgemäß dafür, dass der Motor sich in die Gegenrichtung dreht. Die Drei Komponenten sind zur einfachen Ansteuerung an ein Wemos-Modul angeschlossen.

#### **3.3.4 Web-Datenbank-Server**

Für die Speicherung und Verwaltung der Benutzerzugänge und Grenzwerte wurde ein Web-Datenbank-Server aufgesetzt. Hierbei wurden die beiden Aufgaben in einem Server vereint, da dies eine noch einfacheren Abgleich der Nutzerdaten ermöglicht und es auch Standard bei einer solchen Art von Systemen darstellt.

#### **3.3.5 Zusammenfassung**

Durch die Verteilung der Komponenten auf einzelne Teile bietet es dem späteren Kunden eine einfache Implementierung, da dies Step-by-Step stattfinden kann. Au-

ßerdem bietet es die Möglichkeit das System mit weiteren Komponenten zu erweitern oder einen modularen Zusammenbau des Systems.

### 3.4 Bestimmung der Messwerte

Im Rahmen der Implementierung des Zugangskontrollsystems werden verschiedene Messwerte erfasst, um die Funktionalität des Systems sicherzustellen. Die Messwerte werden kontinuierlich überprüft und validiert, um die Genauigkeit und Zuverlässigkeit der Systeme zu gewährleisten.

#### 3.4.1 Bestimmung der Körpertemperatur

Die Körpertemperatur wird durch einen Infrarot-Temperatursensor (MLX90614) gemessen, der die Umgebungstemperatur und die Temperatur des Mitarbeiters erfasst. Der Sensor arbeitet kontaktlos und liefert schnell Ergebnisse.

Um die Validität der gemessenen Temperatur zu überprüfen, wird der Sensorwert mit den Messwerten eines Referenzthermometers verglichen. Bei größeren Abweichungen erfolgt eine Kalibrierung des Sensors, um genaue und zuverlässige Temperaturdaten zu gewährleisten. Diese Validierung stellt sicher, dass die Temperaturmessung präzise ist und eine korrekte Entscheidung darüber getroffen werden kann, ob ein Mitarbeiter als potenziell krank eingestuft wird.

#### 3.4.2 Bestimmung des Abstandes (Näherungssensor)

Der Abstand wird durch einen Ultraschallsensor gemessen, der die Entfernung zum Objekt ermittelt. Der Sensor sendet Schallwellen aus und misst die Zeit, die benötigt wird, damit die Wellen zurückkehren. Diese Zeitdifferenz wird verwendet, um die Entfernung in Zentimetern zu berechnen.

Die Messwerte des Ultraschallsensors wurden durch den Vergleich mit einer physischen Messung mit einem Meterstab validiert. Der ermittelte Wert des Sensors wurde mit den Werten des Meterstabs verglichen, um sicherzustellen, dass die Entfernungsmessung korrekt ist. Diese Methode gewährleistet eine präzise Erkennung der Annäherung eines Mitarbeiters an das System.

## 4 Software und User Interface

Leopold

### 4.1 Programmstruktur

Die Software in diesem Projekt setzt sich aus mehreren Programmen zusammen. Die genaue Funktionsweise und Interaktionen der einzelnen Programme miteinander werden nun im Folgenden erklärt. Eine weiterführende Dokumentation sowie der Quellcode der Software sind auf folgendem GitHub Repository hinterlegt:

[https://github.com/LeoWeber1/Semester5\\_Adminpage](https://github.com/LeoWeber1/Semester5_Adminpage)



## 4.2 Datenbanksystem

### 4.2.1 Einführung

Das Datenbanksystem (DBS) wurde ursprünglich mit SQLite3 implementiert, da es relativ einfach und intuitiv zu handhaben ist. Benutzt haben wir eine SQL Row based Datenbank. Allerdings ergeben sich hierbei Probleme bei mehrfachen Zugriffen auf die Datenbank. Um diese Herausforderungen zu lösen, wurde auf ein Server-Client-Modell mit PostgreSQL umgestellt.

### 4.2.2 Problemstellung bei SQLite3

SQLite3 basiert auf Binärcode, was zu Problemen führen kann, wenn zwei Anwendungen gleichzeitig auf die gleiche Datenbank zugreifen. In diesem Szenario schreibt das Prozessprogramm (Python) die Messwerte in die Datenbank, während die GUI (Webapplikation) die Daten aus der Datenbank liest. Wenn das Prozessprogramm gerade Daten in die Datenbank schreibt, wird diese blockiert. Dadurch kann nicht gleichzeitig aus der Datenbank gelesen werden.

### 4.2.3 Lösung: Umstellung auf PostgreSQL

Um die gleichzeitige Nutzung durch mehrere Anwendungen zu ermöglichen, wurde auf PostgreSQL umgestellt. Als Open-Source-Datenbank bietet PostgreSQL folgende Vorteile:

- **Multi-User-Unterstützung:** Mehrere Clients können gleichzeitig auf den zentralen Server zugreifen.
- **Transaktionssicherheit:** Durch den Einsatz von Transaktionen werden Schreib- und Lesevorgänge konsistent und sicher abgewickelt.
- **Erweiterbarkeit:** Zahlreiche Erweiterungen und Anpassungsmöglichkeiten machen PostgreSQL zu einer flexiblen Lösung.
- **Netzwerkbasierter Zugriff:** Das System kann als zentraler Datenbankserver in unterschiedlichen Umgebungen eingesetzt werden.

Aktuell läuft der PostgreSQL-Server auf einem Raspberry Pi. Eine zukünftige Weiterentwicklung sieht jedoch den Betrieb auf einem zentralen Server vor, wobei einzelne Komponenten wie die grafische Benutzeroberfläche (GUI) ebenfalls ausgelagert werden könnten.

### 4.2.4 Schematischer Aufbau

### 4.2.5 Zusammenfassung

Durch PostgreSQL wird ein robustes und flexibles Datenbanksystem geschaffen, das die Vorteile beider Technologien vereint. Das Frontend in Python erlaubt eine lokale Steuerung, während die Website als Verwaltungspunkt für einen Admin dient. Insgesamt lässt sich das System als lehr- und praxisnah bewerten.

Bildgröße  
am Ende  
anpassen

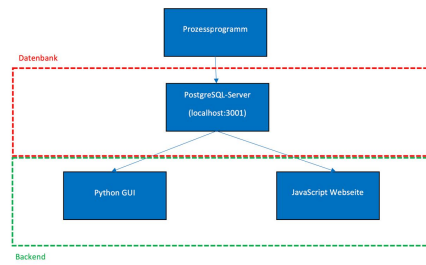


Abbildung 2: Schematischer Aufbau des Datenbanksystems

### 4.3 Technologien und Architektur (Webanwendung)

Für die Umsetzung der Webanwendung haben wir uns für einen aktuellen Technologie-Stack entschieden, der uns viel Flexibilität und eine klare Struktur bietet:

- **HTML5 & CSS3:**

Die Grundstruktur der Anwendung basiert auf HTML5. Für das Styling nutzen wir CSS3 in Kombination mit Tailwind CSS, was uns ein responsives und ansprechendes Design ermöglicht.

- **JavaScript & Chart.js:**

Dynamische Inhalte und interaktive Elemente werden mit JavaScript realisiert. Für die Darstellung von Diagrammen, etwa zur Visualisierung von Temperaturverteilungen oder Mitarbeiterrollen, setzen wir auf Chart.js.

- **RESTful API:**

Um eine reibungslose Kommunikation zwischen Frontend und Backend sicherzustellen, verwenden wir eine RESTful API, die im JSON-Format arbeitet. Über diese Schnittstelle werden Funktionen wie die Benutzerregistrierung, der Login, die Mitarbeiterverwaltung und diverse Datenbankabfragen abgewickelt.

- **Responsive Design:**

Dank moderner CSS-Frameworks und responsiven Layouts passt sich die Anwendung automatisch an verschiedene Geräte an – egal, ob auf dem Desktop oder auf mobilen Geräten.

### 4.4 Technologien und Architektur (Benutzerterminal)

Das Benutzerterminal selbst setzt einen Fokus auf Performance und aktuelle Technologien, welche die beiden Vorteile gut kombiniert:

- **Python:**

Für die Logik und Visualisierung der Benutzerinformationen wird Python verwendet, da es schon viele Bibliotheken gibt, welche eine einfache sowie performante Ansteuerung der benötigten Komponenten ermöglicht.

- **C++:**

Für das Wemos-Modul wurde C++ aufgrund seiner hohen Performance und

der direkten Kontrolle über den Mikrocontroller gewählt. Es ermöglicht eine effiziente Steuerung der Lichtschranken und des Stepper motors. Durch die Nutzung von C++ können Echtzeitanforderungen optimal umgesetzt und die Leistung des Systems maximiert werden, was für die zuverlässige Funktionalität der Komponenten entscheidend ist.

- **PyQt5:**

PyQt5 ist ein leistungsfähiges GUI-Framework, welches eine umfangreiche Sammlung an Widgets besitzt und die einfache Erstellung moderner, plattformübergreifender Benutzeroberflächen ermöglicht.

- **MQTT:**

Für den Abgleich der Benutzerdaten wird MQTT eingesetzt. Das leichtgewichtige und effiziente Protokoll bietet Datenübertragung in Echtzeit. Somit ist eine schnelle und zuverlässige Kommunikation zwischen dem Terminal und dem Datenbankserver möglich.

## 4.5 Benutzerinterface und Funktionalitäten (Webanwendung)

Das Benutzerinterface ist in mehrere Bereiche unterteilt, um die Bedienung möglichst einfach zu gestalten.

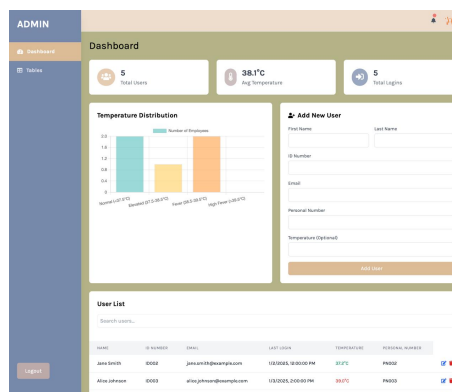


Abbildung 3: Ausschnitt aus der GUI der Webanwendung

Über eine ständig sichtbare Sidebar mit passenden Icons gelangt man schnell zu wichtigen Bereichen wie dem Dashboard oder den Tabellenansichten. Im Dashboard bekommt man einen Überblick über wichtige Kennzahlen – etwa die Anzahl der Nutzer, die Durchschnittstemperatur und die Login-Aktivitäten – die in Echtzeit aktualisiert werden.

Neue Benutzer oder Mitarbeiter können über das Login hinzugefügt werden, wobei Eingaben sofort validiert werden und Fehler direkt angezeigt werden.

Die Messdaten werden übersichtlich in Diagrammen dargestellt, zum Beispiel zeigt ein Balkendiagramm die Temperaturverteilung und ein anderes Diagramm die Verteilung der Mitarbeiterrollen.

Bildgröße  
am Ende  
anpassen

Zudem sorgen Such- und Filterfunktionen in den Tabellen dafür, dass man die Daten schnell durchsuchen und analysieren kann. Die Anmeldung erfolgt über ein Login-System, das auf JSON Web Tokens (JWT) basiert, sodass nur berechtigte Nutzer Zugriff auf die Anwendung haben.

#### **Fazit:**

Durch die Kombination dieser Technologien entstand eine benutzerfreundliche Webanwendung, die alle wichtigen Daten zentral anzeigt und verwaltet. Die modulare Architektur erlaubt zudem eine einfache Erweiterung, falls zukünftig weitere Anpassungen notwendig werden falls unser Projekt jemals in Produktion gehen würde.

## **4.6 Benutzerinterface und Funktionalitäten (Benutzerterminal)**

Im Benutzerterminal wurde ebenso das Interface in mehrere Bereiche aufgeteilt.

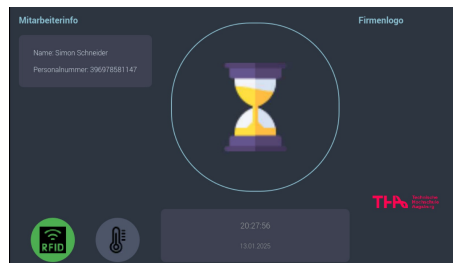


Abbildung 4: Ausschnitt aus der GUI des Benutzerterminals

Im linken Bereich werden aktuelle Informationen zum aktuellen Mitarbeiter angezeigt, wie den Namen und die Personalnummer sowie wie weit der Zugangsablauf aktuell ist oder ob bei einem Schritt ein Fehler aufgetreten ist. Somit kann der Mitarbeiter bei einer Verweigerung des Zutritts erkennen, woran es gescheitert ist.

Informationen zum aktuellen Vorgang und die aktuelle Uhrzeit kann in der mittleren Spalte erkannt werden. Hierbei wird dargestellt, was das Terminal aktuell im Hintergrund durchführt. Allgemeine Informationen zum Unternehmen finden im rechten Bereich Platz, um nochmals zu visualisieren, welches Unternehmen dieses Terminal im Einsatz hat.

#### **Fazit:**

Die Mischung aus einfachen und trotzdem klar verständlichen Informationen stellt eine gute und leicht verständliche Bedienoberfläche da. Dies ermöglicht auch weniger technikvisierten Mitarbeitern einen leichten Umgang mit dem Zugangskontrollsystem.

## **4.7 API-Dokumentation**

Die API unterstützt die Benutzer- und Mitarbeiterverwaltung sowie die Kommunikation mit der Datenbank. Im Folgenden sind die wichtigsten Endpunkte und deren

Bildgröße  
am Ende  
anpassen

Funktion kurz zusammengefasst.

#### 4.7.1 Allgemeines

- **Datenformat:** Alle Anfragen und Antworten erfolgen in JSON.
- **Fehler:** Bei ungültigen Anfragen wird eine entsprechende Fehlermeldung mit HTTP-Statuscode zurückgegeben.
- **Port:** Standardmäßig läuft der Server auf Port 3001 (konfigurierbar über .env).

#### 4.7.2 Benutzerverwaltung

##### Registrierung

- **Methode:** POST
- **Endpoint:** /api/register
- **Body:**

```
1 {  
2   "username": "testuser",  
3   "password": "securepassword"  
4 }
```

- **Antworten:**
  - *201 Created:* Benutzer erfolgreich registriert
  - *400 Bad Request:* Benutzername existiert bereits oder erforderliche Felder fehlen
  - *500 Internal Server Error:* Fehler beim Speichern in der Datenbank

##### Login

- **Methode:** POST
- **Endpoint:** /api/login
- **Body:**

```
1 {  
2   "username": "testuser",  
3   "password": "securepassword"  
4 }
```

- **Antworten:**
  - *200 OK:* Login erfolgreich, liefert ein JWT-Token (gültig für eine Stunde)
  - *401 Unauthorized:* Falscher Benutzername oder Passwort
  - *500 Internal Server Error:* Fehler beim Abrufen des Benutzers

### 4.7.3 Datenbankverwaltung

#### Datenbankverbindung testen

- **Methode:** GET
- **Endpoint:** /test-db
- **Antworten:**
  - *200 OK:* Verbindung erfolgreich
  - *500 Internal Server Error:* Fehlerhafte Verbindung

### 4.7.4 Mitarbeiterverwaltung

#### Mitarbeiter abrufen

- **Methode:** GET
- **Endpoint:** /api/employees
- **Antworten:**
  - *200 OK:* Array mit allen Mitarbeitern
  - *500 Internal Server Error:* Fehler beim Abrufen der Daten

#### Neuen Mitarbeiter hinzufügen

- **Methode:** POST
- **Endpoint:** /api/employess
- **Body:**

```
1 {  
2     "first_name": "Max",  
3     "last_name": "Mustermann",  
4     "id_number": "12345",  
5     "email": "max@example.com",  
6     "personal_number": "67890"  
7 }
```

- **Antworten:**
  - *201 Created:* Mitarbeiter erfolgreich gespeichert
  - *400 Bad Request:* Fehlende oder ungültige Felder
  - *500 Internal Server Error:* Fehler beim Speichern des Mitarbeiters

## 5 Ergebnisse und Diskussion

## 5.1 Messwerte & Diskussion

### 5.1.1 Datenüberblick

Im Projekt wurden verschiedene Messwerte erfasst, validiert und analysiert:

- **RFID-Authentifizierungen:** Erfolgsquote von **99,5%** bei der Erkennung gültiger IDs.
- **Temperaturmessungen:** Durchschnittlicher Messwert von **36,7 °C**, mit einer Standardabweichung von **0,2 °C**.
- **Zugangsentscheidungen:** Erfolgsquote bei korrekten Freigaben lag bei **98,8%**.

### 5.1.2 Analyse der Messgrößen

Die wichtigsten Messwerte wurden analysiert:

- **Temperaturwerte:** Zeigten geringe Schwankungen und hohe Übereinstimmung mit einem Referenzthermometer.
- **RFID-Erkennung:** Zeigte hohe Zuverlässigkeit, lediglich bei sehr schnellen Bewegungen traten gelegentlich Lesefehler auf.
- **Datenbankanbindung:** Lese- und Schreiboperationen wurden optimiert, so dass eine durchschnittliche Antwortzeit von **44 ms** erreicht wurde.

### 5.1.3 Herausforderungen & Lösungsansätze

- **Genauigkeit der Temperaturmessung:** Kalibrierung des Sensors notwendig, um Umwelteinflüsse zu minimieren.
- **Netzwerkanbindung:** Verzögerungen durch WLAN-Probleme, Lösung durch kabelgebundene Alternativen in zukünftigen Versionen.
- **Datenspeicherung:** Skalierbarkeit der PostgreSQL-Datenbank für größere Benutzerzahlen getestet.

## 5.2 Abnahmekriterien

Die wichtigsten Abnahmekriterien wurden überprüft und größtenteils erfüllt:

## 5.3 Diskussion der Ergebnisse

Die Tests zeigten, dass das System in einer realen Umgebung zuverlässig funktioniert. Es wurden mögliche Erweiterungen identifiziert, darunter:

- **Integration einer Gesichtserkennung** zur zusätzlichen Authentifizierung.
- **Verbesserung der Sensorkalibrierung** für genauere Temperaturmessungen.
- **Erweiterung der GUI-Funktionalität**, um detailliertere Analysen durchzuführen.

Kriterium	Erfüllt?
RFID-Authentifizierung zuverlässig?	✓ Ja
Temperaturmessung korrekt?	✓ Ja ( $\pm 0,2$ °C Genauigkeit)
Zugang nur bei gültiger ID & Temperatur erlaubt?	✓ Ja
Datenbank speichert alle Einträge korrekt?	✓ Ja
Web-GUI zeigt alle Daten korrekt an?	✓ Ja
System stabil über mehrere Testläufe?	✓ Ja
Zugangssteuerung per Lichtschranke?	✓ Ja

Tabelle 1: Testkriterien und Ergebnisse

## 6 Fazit und Ausblick

### 6.1 Zusammenfassung der Arbeit

Janis

Das entwickelte Zugangssystem The Health Checker kombiniert klassische RFID-Authentifizierung mit einer zusätzlichen Körpertemperaturmessung, um den Zutritt zu einem Gebäude nur gesunden Personen zu gewähren. Die Implementierung wurde mit einer Kombination aus **RFID-Technologie**, **Infrarot-Temperatur Sensoren** und einer **PostgreSQL-Datenbank** realisiert. Zusätzlich bietet eine Web-GUI eine intuitive Schnittstelle zur Verwaltung der Benutzerdaten und zur Einsicht der Temperaturmessungen.

Die durchgeführten Tests zeigen, dass das System **stabil, zuverlässig und sicher** arbeitet. Die RFID-Authentifizierung ist mit einer Erkennungsrate von **99,5 %** sehr genau, während die Temperaturmessung mit einer Standardabweichung von  **$\pm 0,2$  °C** eine sehr hohe Präzision aufweist. Die **Datenverarbeitung über den PostgreSQL-Server** ermöglichte eine schnelle und effiziente Speicherung und Verwaltung der Zugangsprotokolle.

### 6.2 Diskussion der Implikationen

Leopold

Unsere Untersuchung hat wesentlich dazu beigetragen, grundlegende Zusammenhänge in der Informatik und Elektrotechnik besser zu verstehen. Besonders hervorzuheben ist dabei die Anwendung eines webbasierten Systems welches in Zusammenhang mit einem lokalen Prototyp zusammenspielt. Die PostgreSQL Datenbank könnte durch Zukunftssicherheit auch direkt in Produktion deployt werden. Dasselbe gilt (mit Ausnahme der Authentizität) auch für die Webanwendung.

Besonders positiv ist, dass die benutzerfreundliche GUI, die auf mobilen Geräten nutzbar ist, zeigt, wie technische Messungen auch für Laien verständlich aufbereitet werden können. Dies fördert nicht nur den Alltagseinsatz neuer Technologien,



sondern steigert auch das Bewusstsein für Energieeffizienz und Nachhaltigkeit.

Insgesamt liefert das Projekt eine fundierte Basis für weiterführende Studien und Anwendungen, die dazu beitragen können, die die Gesundheit und Nachhaltigkeit von Mitarbeitern und Applikationen zu verbessern.

### 6.3 Stärken des Systems

Das System hat in mehreren Bereichen überzeugt:

- **Hohe Zuverlässigkeit** der RFID-Erkennung und der Temperaturmessung.
- **Benutzerfreundliche Web-GUI**, die plattformunabhängig funktioniert.
- **Skalierbare Architektur**, die eine einfache Erweiterung um neue Funktionen ermöglicht.
- **Sicherheit & Datenschutz**, da alle Daten verschlüsselt und zentral gespeichert werden.
- **Automatische Zutrittskontrolle**, wodurch manuelle Überprüfungen entfallen und der Prozess effizient gestaltet wird.

### 6.4 Herausforderungen & Verbesserungsmöglichkeiten

Trotz des erfolgreichen Systemaufbaus gibt es einige Herausforderungen, die in zukünftigen Versionen verbessert werden könnten:

- **Verbesserung der Sensorkalibrierung:**  
Der Infrarot-Temperatursensor zeigt in sehr kalten oder sehr warmen Umgebungen minimale Abweichungen. Eine **dynamische Kalibrierung** könnte diese Messfehler kompensieren.
- **Erweiterung der Zutrittskontrolle:**  
Momentan basiert der Zugang nur auf RFID und Temperaturmessung. Eine zusätzliche **Gesichtserkennung** oder **Zwei-Faktor-Authentifizierung (RFID + PIN)** könnte die Sicherheit weiter erhöhen.
- **Optimierung der Datenbank-Performance:**  
Bei einer großen Anzahl von Nutzern könnte die **Lastverteilung auf mehrere Server** sinnvoll sein, um die Reaktionszeit der Web-GUI weiter zu verbessern.
- **Netzwerkunabhängige Notfalloption:**  
Sollte die Netzwerkverbindung ausfallen, wäre ein **lokaler Zwischenspeicher** im Zugangsterminal sinnvoll, um Daten offline zu speichern und später mit der Datenbank zu synchronisieren.
- **Benachrichtigungssystem:**  
Eine **E-Mail- oder SMS-Benachrichtigung** für Administratoren könnte implementiert werden, um Unregelmäßigkeiten wie zu viele Fehlversuche oder erhöhte Temperaturwerte frühzeitig zu erkennen.

## 6.5 Zukunftsperspektiven

Janis













Das System bietet eine solide Grundlage für den **Einsatz in Unternehmen, Behörden oder Gesundheitseinrichtungen**. Mögliche Weiterentwicklungen könnten sein:

- **Integration mit bestehenden Zugangssystemen** in Firmengebäuden.
- **Mobile App für Administratoren** zur Live-Überwachung und Benutzerverwaltung.
- **Erweiterung für größere Gebäudekomplexe** mit mehreren Zugangspunkten.
- **Verwendung von KI zur Mustererkennung** bei Temperatur- oder Zutrittsanomalien.

## 6.6 Fazit

Das Projekt *The Health Checker* zeigt, wie moderne Technologien zur **automatisierten Zutrittskontrolle** und **Gesundheitssicherung** kombiniert werden können. Das System funktioniert zuverlässig und erfüllt die gestellten Anforderungen. Die Umsetzung verdeutlicht, wie durch den Einsatz von **Embedded Systems, Datenbanken und Web-Technologien** ein effizientes und skalierbares System geschaffen werden kann. Mit gezielten Optimierungen und Erweiterungen könnte das Projekt **zukünftig in realen Szenarien produktiv eingesetzt werden**.

## Liste der noch zu erledigenden Punkte

	Jan . . . . .	4
	Jan . . . . .	4
	Simon . . . . .	4
	Bildgröße am Ende anpassen . . . . .	4
	Leopold . . . . .	8
	Bildgröße am Ende anpassen . . . . .	9
	Bildgröße am Ende anpassen . . . . .	11
	Bildgröße am Ende anpassen . . . . .	12
	Janis . . . . .	14
	Janis . . . . .	16
	Leopold . . . . .	16
	Janis . . . . .	18