Technical Report - **Product specification**

Count Me In

Course:	IES - Introdução à Engenharia de Software
Date:	Aveiro, <date of submission>
Students:	107849: Alexandre Cotorobai 108073: Bernardo Figueiredo 108215: Hugo Correia 109089: Joaquim Rosa
Project abstract:	CountMeIn is an innovative service that leverages camera technology and advanced algorithms to effectively count the number of people within a designated area.

Table of contents:

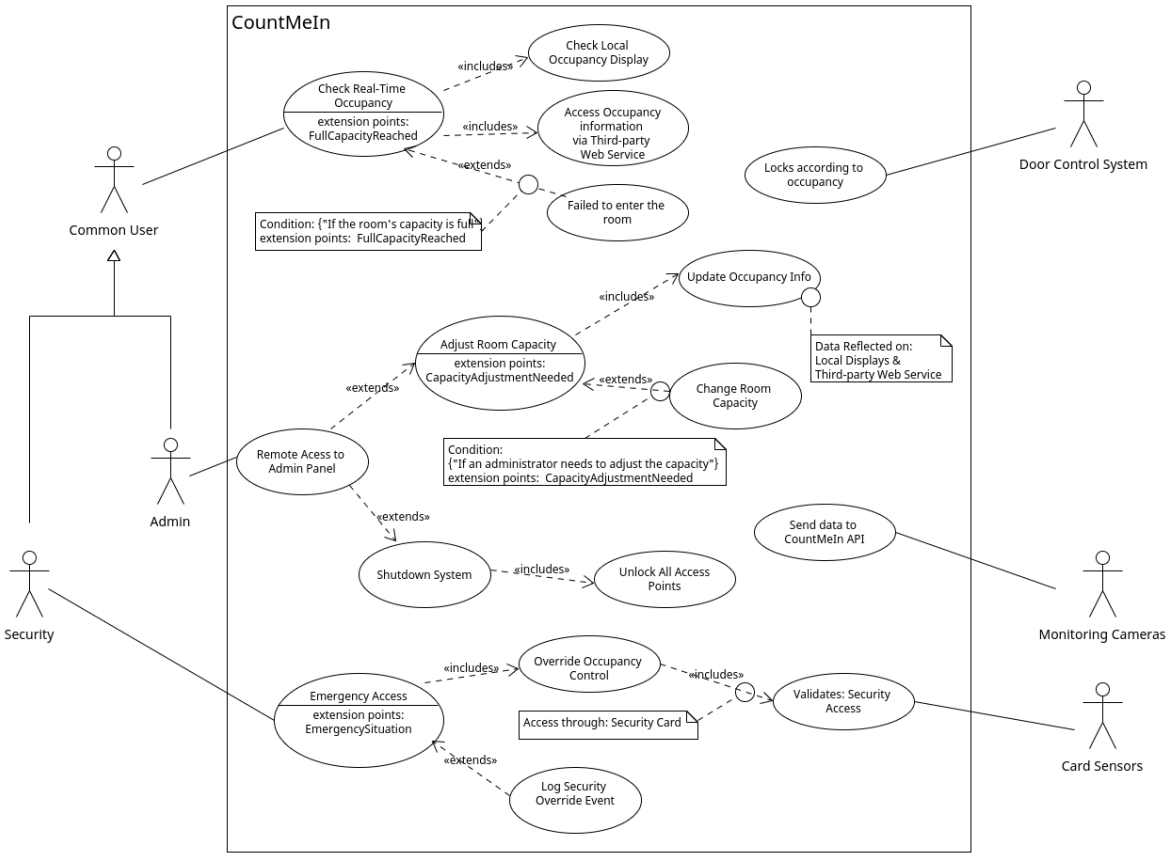
1 Introduction

The CountMeIn project, part of the Integrated Engineering Systems (IES) course, introduces a space management system designed to optimise occupancy in constrained environments. The system integrates real-time monitoring with dynamic access control, distinguishing itself from existing solutions through its automated, data-driven approach to managing foot traffic and enhancing user experience. This report details the product concept, personas, scenarios, and user requirements that shape the development of CountMeIn.

2 Product concept

Vision statement

For businesses with limited floor space or who want to restrict foot traffic, our aim is to provide a service that helps put the control of the flow of customers back in your hands. Unlike other products on the market, CountMeIn integrates real-time graphical data, collected via cameras installed at the desired location, with algorithms to provide on-site feedback via graphical displays, with the option of integrating the data with an autonomous door system that unlocks/locks depending on the current information. Our service also offers an online platform where our customers may edit the settings per floor/division and a Private API that returns information about the service for the customer to integrate with their own platform.



Personas and Scenarios

Personas

Name: Sofia

Demographics:

- **Age:** 22
- **Course:** Software Engineering

Profile: Sofia is a dedicated software engineering student, deeply engaged in her academic pursuits. The library is her sanctuary for learning and coding, where she spends considerable time honing her skills for a future in tech. Her life is meticulously organised, as she balances a rigorous study schedule with her personal development in the field of technology.

Motivations: Sofia is driven by her ambition to excel in software engineering. She values a study space that minimises distractions and maximises her productivity. Tools and services that can streamline her study sessions and enhance her focus are essential to her daily academic routine

Name: Dona Celeste

Demographics:

- **Age:** 52
- **Role:** Library Administrator

Profile: Dona Celeste is the Library Administrator who has recently overseen the implementation of the CountMeIn service. She is dedicated to maintaining an environment that is both orderly and welcoming for all library users.

Motivations: Dona Celeste is motivated by her commitment to providing a well-managed space that enhances the educational experience. She values the CountMeIn system for its ability to help her monitor and adjust the library's occupancy, ensuring a balanced and accessible environment for students like Sofia.

Name: Carlos Costa

Demographics:

- **Age:** 40
- **Role:** Library Security

Profile: Carlos is an integral part of the library's security team, tasked with maintaining a safe and orderly environment. His approach is proactive and focused on the well-being of library patrons.

Motivations: Carlos's primary motivation is the safety and security of the library's visitors and staff. He finds the CountMeIn system invaluable for its support in managing occupancy levels, which is crucial for both everyday security and emergency situations.

Scenarios

Scenario 1 - Local Display

Sofia is in the middle of a week of intensive study for a crucial exam to finish her Software Engineering degree. She has planned to spend the day in the library to do one last revision.

When she arrives at the library, the graphic display indicates that the library is currently full.

Based on the information given, she decided to go and study at an alternative location, for example, DETI and didn't need to waste time trying to find a place in the library.

Although she prefers the library, the ability to make this decision saves her time and ensures that she can maintain her productivity and focus throughout the day.

Scenario 2 - Online Website

Sofia has an important exam next week and wants to study in the library, but she's worried about how full it is. She accesses the library's platform, which shows the library's capacity in real-time, and sees that there are currently plenty of places available. This relieves her because she knows she doesn't have to rush and can get to the library calmly.

Scenario 3 - Dona Celeste's Library Management

Dona Celeste, a library keeper, has been using the CountMeIn service to efficiently manage the library's occupancy and improve the experience for students. One day, she encounters a situation that requires her to utilise the system:

Dona Celeste receives a notification that warns her of a possible visitor surge due to an upcoming seminar in the Software Engineering department. She knows that this increased demand for library resources could lead to a crowded and less effective studying environment.

Using her admin access, she logs into the CountMeIn system and makes real-time adjustments. She decides to open a previously closed section of the library to accommodate the additional students. She also extended the library's operating hours for that day to provide more flexibility for the students.

Dona Celeste sends a notification to the library staff, informing them of the changes and

providing guidelines to ensure a smooth experience for the students during this period of high demand.

Dona Celeste's ability to manage the library's capacity and resources in real-time helps ensure that students like Sofia can have a conducive studying environment, even during special events or unexpected situations. This proactive approach improves the overall experience of library users.

Scenario 4 - Library Security Guard

The security guard enters the art gallery to deal with an unforeseen incident. The guard carries an ID card, allowing them to enter and exit the venue ignoring the set capacity.

Product requirements (User Stories)

Epic 1: Real-Time Occupancy Monitoring
User Story 1: Real-Time Occupancy Information Access
Priority: High

As a library patron,
I want to access real-time occupancy information of the library,
So that I can plan my visit to ensure a study spot without the need to wait or return at a later time.

Acceptance Criteria:

1. Given a user desires to view the current occupancy status of the library,

2. And the library has successfully integrated CountMeIn's API into their website,

3. When the user navigates to the designated occupancy information section on the library's website,

4. Then the website should display the up-to-date occupancy levels, reflecting real-time changes as they occur.

5. Additionally, the displayed information should be clear, and easily understandable, and should indicate if the space is available for immediate use or if there are any restrictions due to full capacity.

User Story 2: Local Occupancy Display Check

Priority: High

As a library patron,

I want to be able to check the library's space occupancy through a local display at the entrance,

So that I can immediately know if there is available space for me to use.

Acceptance Criteria:

1. Given a user approaches the library's entrance,

2. And the library has a local display connected to the CountMeIn system,

3. When the user views the display,

4. Then the display should show the current occupancy status, updated in real-time.

5. The display should indicate clearly, possibly with a colour-coded system, whether the library is full or if space is available.

6. If the library is full, the display should provide an indication of the expected wait time or peak hours to assist in planning a future visit.

Epic 2: Administrative Control and Space Optimization

***User Story 3*:** Space Optimization for Enhanced Student Experience

Priority: High

As the library administrator,

I want to utilise the CountMeIn system to adjust room capacities,

So that I can manage space more effectively and enhance the study experience for students.

Acceptance Criteria:

1. Given the library administrator identifies a need to adjust the space based on user flow,

2. And the administrator has access to the CountMeIn admin panel,

3. When the administrator inputs new capacity limits or opens/closes sections of the library within the system,

4. Then the system should immediately reflect these changes in real-time, both on the local display and on the library's website.

5. After the changes, the system should automatically adjust the access control mechanisms (like door locks) to comply with the new capacity settings.

User Story 4:: Real-Time Data Monitoring and Visualization for Admins

Priority: High

As an administrator of the CountMeIn system,

I want to monitor and visualize real-time data for each room I manage,

So that I can have an informed overview of occupancy trends and make data-driven decisions to optimize space utilization.

Acceptance Criteria:

1. Given an administrator needs to access occupancy data for a specific room,

2. And the administrator has authenticated access to the CountMeIn admin panel,

3. When they navigate to the room monitoring section in the admin panel,

4. Then they should be presented with real-time data visualizations, including current occupancy, historical trends, and peak usage times.

5. And this data should be updated dynamically to reflect the latest information.

6. And the interface should provide intuitive and easy-to-understand graphical representations of the data.

Epic 3: Overriding the implemented system in case of emergency

User Story 5: Emergency Access for Security Personnel

Priority: High

As a security guard hired by the client,

I want to be able to bypass the occupancy control system to access any area within the premises,

So that I can perform my duties effectively, especially in urgent situations.

Acceptance Criteria:

1. Given a security guard needs to access a room or area in an emergency or for security reasons,

2. When they present their authorised security card to the card reader,

3. Then the system should recognize the security clearance and immediately unlock the door,

4. The system should log the security override event for accountability and reporting purposes.

User Story 6: Emergency System Shutdown

As a client admin,

I want to have the ability to quickly shut down the occupancy control system in case of an emergency (e.g. Fire),

So that all occupants can exit the building safely without any delay.

Acceptance Criteria:

1. Given a client admin identifies an emergency situation where an immediate evacuation is necessary,

2. When they access the emergency shutdown feature in the admin panel,

3. And activate the shutdown procedure,

4. Then the system should unlock all controlled access points instantly,

5. The system should also signal the emergency mode on displays and the client's website,

6. The system should notify the appropriate personnel about the activation of the emergency protocol.

3 Architecture notebook

Key requirements and constraints

The CountMeIn system is conceived to address the unique demands of space management and occupancy regulation within public and private environments. The architectural design is driven by several key requirements and constraints that dictate the system's development:

Integration and Compatibility:

1. ***Integration with Existing Infrastructure:***

   The system must integrate seamlessly with existing security and monitoring infrastructures within facilities such as libraries or galleries. This involves leveraging legacy hardware and software systems to ensure cohesive functionality.

2. ***External System Integration:***

   CountMeIn will offer a Private API to allow third-party systems, such as a client's website, to retrieve occupancy information. The architecture must define clear interfaces for this integration, ensuring compatibility and maintainability.

System Performance:

3. ***Performance and Scalability:***

   CountMeIn is required to process real-time data efficiently from multiple sensors and cameras. The architecture must support high throughput and low latency to ensure timely updates to occupancy status, especially during peak hours or events,

as stipulated by the user scenarios.

Example:

- Given the library's dependence on the CountMeIn system for day-to-day operations, as seen in Dona Celeste's scenario, the architecture must ensure high availability and fault tolerance to minimise disruptions in service.

4. ***Network Reliability:***

   Reliable network connectivity is a foundational requirement, as the system's effectiveness is dependent on the uninterrupted flow of data between sensors, Raspberry Pi units, and the central server.

User Interaction:

5. ***Local Graphical Interface:***

   The system is tasked with delivering a local graphical interface that conveys real-time information regarding the occupancy and capacity of the room. This feature is pivotal for the user's experience, as it directly informs them of the current state of the environment, enabling decisions such as whether to enter or seek alternative locations. Drawing from User Story 1, the interface plays a key role for patrons like Sofia, who rely on the immediacy and accuracy of the displayed data to effectively manage their study schedules without unnecessary delays.

Security and Privacy:

6. ***Data Privacy and Security:***

   Robust measures must be implemented to protect sensitive user data and prevent unauthorised access. This is critical, especially given the system's capabilities for real-time monitoring and control over access to various sections within a facility.

7. ***Emergency Protocols:***

   The system must include a robust set of emergency protocols, allowing for immediate system shutdown or override by authorised personnel in case of emergencies, as detailed in the user stories.

8. ***Room Access Management:***

   The administrative interface should provide the capability to remotely lock and unlock doors of associated rooms. This feature empowers administrators to control room usage and maintain security, ensuring that spaces can be secured when not in

use or in response to security concerns.

System Operation and Maintenance:

9. ***Hardware Dependencies:***

   The system relies on Raspberry Pi units to interface with cameras, card sensors, door locks, and displays. It is imperative that the architecture isolates these hardware dependencies to minimise the impact of potential hardware-related issues on the overall system.

10. ***Long-Term Maintenance:***

    Consideration for long-term system maintenance is crucial. The architecture should be designed in a way that facilitates updates, scaling, and integration with future technologies.

11. ***Deployment Concerns:***

    The architecture must be deployable on cloud platforms to leverage scalable resources and ensure system availability. Additionally, considerations for on-premises deployment should be addressed to cater to clients with specific security requirements.

Admin Web Interface:

12. ***Configuration Management:***

    The system must provide a secure admin web interface where authenticated administrators can adjust room capacity settings associated with their account. This interface should facilitate quick modifications to accommodate changing occupancy needs.

13. ***Access Control:***

    Entry into the administrative interface requires validation of email and password credentials, ensuring that only authorised users can adjust system settings. This security measure is essential to maintain the integrity of the management operations.

Data Visualization

14. ***Analytics Dashboard:***

    The admin interface should include an analytics dashboard that presents various data visualisations, such as the number of occupants over time, peak usage hours, and historical occupancy trends.

Architectural Framework:

15. Client-Server Architecture:

The processing and decision-making capabilities will reside on a central server, while clients will handle user interaction. The server side will utilize Kafka for message brokering and will be supported by SQL and Redis databases for data persistence.

Architectural view

Our project architecture is divided into 3 main parts:

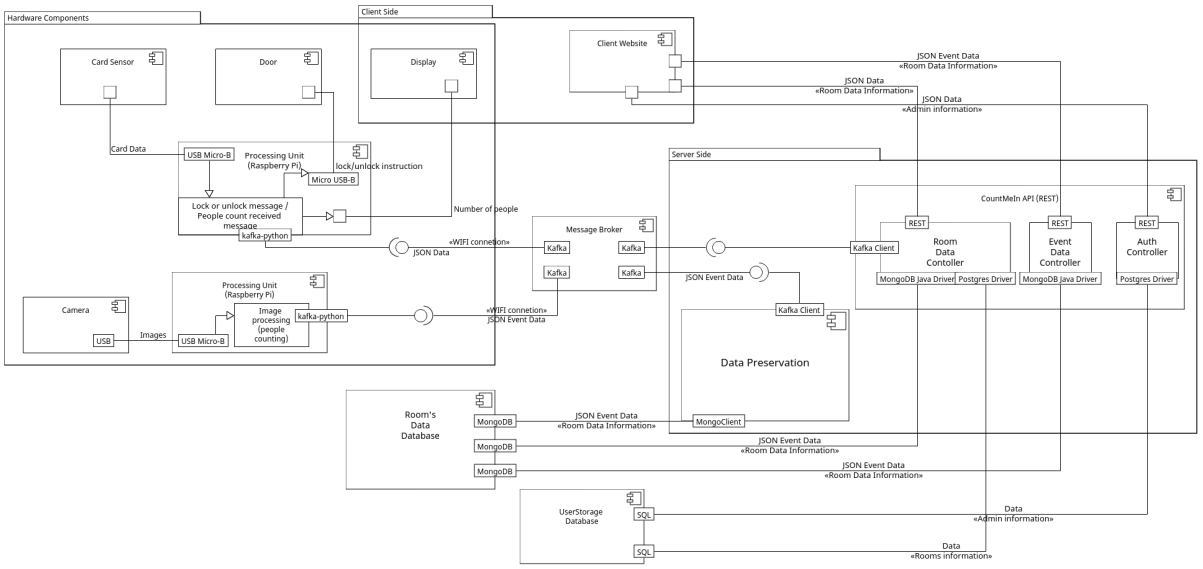
The Hardware Components consist of the camera that captures images of the room, the display that shows its current capacity, the door and card sensor, and two Raspberry Pis. One Raspberry Pi processes the images captured by the camera, while the other keeps the display updated and manages room access, locking and unlocking the door.

The Server Side handles data processing and operational logic. It comprises two databases: one for storing user information and another for real-time room occupancy tracking. Additionally, it includes two APIs— one focused on business logic, responsible for processing data and managing information flow, and the other based on Spring Boot, facilitating interactions between the system and the client's website. Also, there's a message broker that manages the connection and data transmission among hardware components, the business logic API, and the Spring Boot API.

The Client Side is the part of the system with which end-users interact to obtain information and perform actions. This includes a real-time display showing the current room capacity and a website accessible to customers. On the website, users can check the real-time capacity of specific rooms and perform administrative operations.

Module interactions

The interactions between modules are described by the following diagram:



Interaction examples between the modules:

Admin changes room capacity:

- Once logged in, the admin can change the room capacity, then the Admin Panel (web) will request to the API which will forward the request to both the database (to update its values) and the display via the Message Broker.

Admin closes the room door:

- Once logged in, the admin can manually force to lock a door, then the Admin Panel (web) will request to the API which will forward the request to the Message Broker to lock the door.

Security Unlocks door:

- The security can open the door even if it is locked. By having an authenticated card it is recognized and the door unlocked.

Camera counts people:

- Between fixed intervals of time, the camera will take shots and the number of people inside the room will get counted, that information will be retrieved and forwarded by the Message Broker to the processing unit responsible for the display, which will update its numbers.

4 Information perspective

Admin Class:

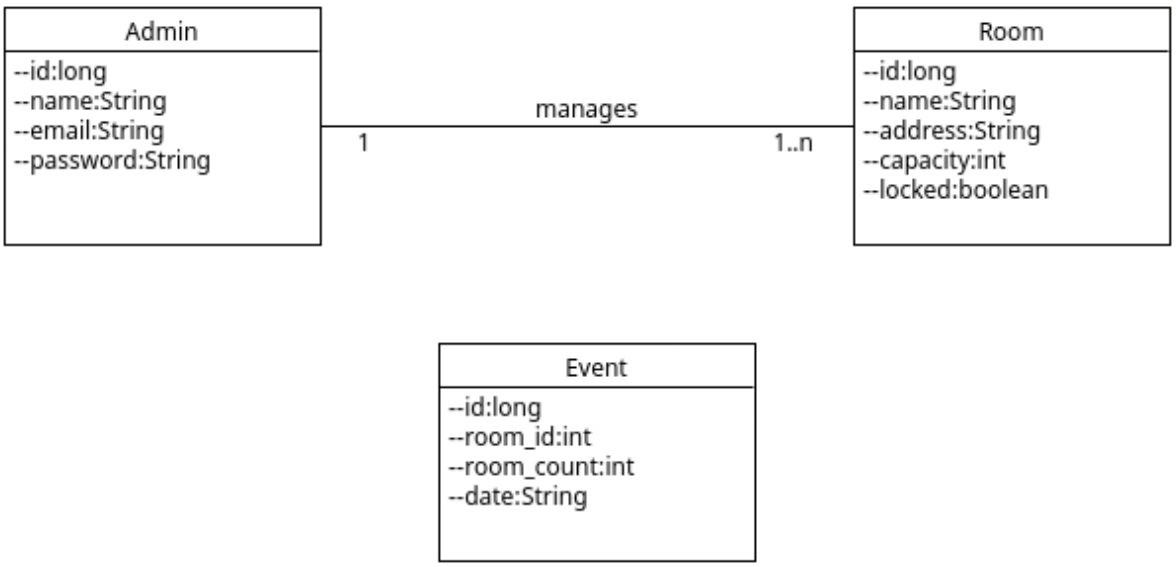
- Manages ***'Room'*** entities;
- Attributes include identification, name, email, and password, indicating the admin's credentials and access rights within the system.

Room Class:

- Managed by an ***'Admin'***.
- Attributes such as identification, name, address, capacity, and a locked status, define the characteristics and state of a room within the library or space monitored by CountMeIn.

Event Class:

- Associated with ***'Room'*** via room_id.
- Records events with attributes like identification, the associated room's identification, count of people in the room, and date of the event, capturing occupancy data over time for a specific room.



The ***'Admin'*** class is responsible for managing multiple ***'Room'*** entities, indicating a one-to-many relationship where a single admin can oversee several rooms. Each ***'Room'*** can have various ***'Event'*** records associated with it, which track occupancy-related activities and changes in the room's state. The locked boolean in ***'Room'*** suggests that the system can also manage access control to the rooms.

These classes and their relationships support the CountMeIn system's functionality, allowing for the management of space occupancy and providing a structured approach to maintaining a balanced environment for users like students or library visitors.

5 References and resources

- https://www.mongodb.com/docs/
- https://www.postgresql.org/
- https://kafka.apache.org/
- https://spring.io/projects/spring-boot
- https://react.dev/
- https://ui.shadcn.com/
- https://tanstack.com/query/v3/
- https://www.typescriptlang.org/
- https://axios-http.com/docs/intro
- https://lucide.dev/guide/packages/lucide-react
- https://recharts.org/en-US/
- https://tailwindcss.com/

In the frontend development of our project, we adopted React as the core library, complemented by several other essential libraries. Shadcn UI provided a collection of pre-designed components for our application. For data visualization, we seamlessly integrated Recharts, allowing us to present data in an interactive and visually appealing manner.

Additional frontend enhancements included the use of TanStack Query for simplified state management and data fetching, TypeScript for static typing and improved code quality, Axios for efficient HTTP requests, Lucide React for customizable SVG icons, and Tailwind CSS for a utility-first approach to styling, ensuring a responsive and modern design.

Spring Boot served as the backbone of our backend, providing a robust and convention-driven approach to application development. For data storage, we utilized MongoDB and PostgreSQL, ensuring efficient and scalable data management.

To facilitate communication between various components, we incorporated Apache Kafka. This enabled real-time data pipelines, enhancing the overall performance and responsiveness of our system.