International Journal of Innovative Technology and Interdisciplinary Sciences
https://journals.tultech.eu/index.php/ijitis
ISSN: 2613-7305
Volume 8, Issue 4
DOI: https://doi.org/10.15157/IJITIS.2025.8.4.1173-1194
Received: 21.10.2025; Revised: 04.12.2025; Accepted: 09.12.2025

Research Article

HealthBeam: Design and Implementation of a Healthcare Automation System

Elior Vila[1]*, Ina Papadhopulli[2], Nikolay Andonov[3], Kamen Spassov[4], Aleksandër Biberaj[5]

[1] Department of Informatics, University of Elbasan „Aleksandër Xhuvani", Elbasan, Albania
[2] Department of Informatics Engineering, Polytechnic University of Tirana, Tirana, Albania
[3] Faculty of Mathematics and Informatics, Sofia University „St. Kl. Ohridski ", Sofia, Bulgaria
[4] Faculty of Economics and Business Administration, Sofia University„ St. Kl. Ohridski", Sofia, Bulgaria.
[5] Department of Electronics and Telecommunication, Polytechnic University of Tirana, 1000 Tirana, Albania
***elior.vila@uniel.edu.al**

Abstract

Many healthcare facilities still rely on outdated communication systems or lack digital tools to support efficient patient interaction and service delivery. This study introduces HealthBeam, a digital platform designed to minimize paper-based processes such as patient examination records and medical charts. The paper presents its conceptual design, system architecture, and initial platform implementation aimed at enabling real-time communication, secure data management, and streamlined workflows in clinical environments. HealthBeam integrates centralized data storage for authorized access, prioritizes rapid response in emergency scenarios, and promotes environmentally sustainable digital documentation. In addition to its technical development, the platform is deployed for public use through an accessible web interface. The findings demonstrate how targeted digital interventions can improve operational productivity, data management, and patient-centered care in modern healthcare systems.

Keywords: Healthcare IT Systems; e-Health Platform; Hospital Information Systems; Programming Languages; Software Testing.

INTRODUCTION

Medicine may be one of the oldest human activities, but it has also been one of the fastest growing in recent years under the influence of new technologies. New electronic healthcare platforms, mobile applications, and various cellular body monitoring devices appear every month and promise to change how we treat or prevent illnesses [1-7]. Healthcare technology includes a wide range of tools such as medical instruments, information systems, artificial intelligence (AI), cloud services, and blockchain applications designed to support healthcare delivery. E-Health is defined as the use of information and

communication technologies (ICT) for health. It is recognized as one of the most rapidly growing areas in health today [1, 8-15].

Modern technologies provide healthcare professionals with diverse tools to diagnose diseases, access medical information instantly, and deliver more accurate treatment. Internet connectivity and cloud-based platforms facilitate secure data exchange, enabling physicians to consult online databases, medical portals, and professional communities to obtain clinical information or collaborate with specialists worldwide [4, 16-22]. These advancements demonstrate how digital systems contribute to faster decision-making and improved medical outcomes.

The growing availability of smart devices has introduced new ways to monitor patients remotely without requiring physical visits. Wearable technologies like smartwatches, fitness bands, blood pressure monitors, and sensor-embedded garments-track physiological signals and transmit them directly to healthcare providers in real time [5]. By offering continuous access to vital data, these tools support early detection of abnormalities, reduce unnecessary hospital visits, and enhance personalized treatment strategies. Pharmaceutical research also benefits from the increasing availability of large-scale health data derived from digital and online sources.

AI has seen significant advancements in recent years, and its impact is increasingly noticeable in healthcare. The value of AI in healthcare data analysis is becoming more apparent, especially with the widespread use of wearable sensors [2]. These devices periodically collect vast amounts of health-related data, which, when combined with Big Data Analytics (BDA), enable deeper insights and more accurate predictions. The integration of AI and BDA allows for efficient processing of complex datasets, leading to improved patient monitoring, early diagnosis, and the development of personalized treatment strategies [3]. Many professionals believe that these technologies can help, both in the widespread use of prophylactic rules and in certain areas of the treatment process. It is now known that these technologies can be used for the benefit of patients and doctors. There is no doubt, of course, the view that mobile technologies can and already have a positive impact on both the development of the medical sciences themselves and the organization of healthcare **Error! Reference source not found.**.

Despite these technological developments, many healthcare institutions still rely on outdated communication systems and extensive paper-based documentation. Such practices slow down medical workflows, restrict access to essential patient data, and reduce efficiency in clinical environments. To address these limitations, this work presents HealthBeam, a digital platform designed to automate the retrieval of essential patient information using a single device. HealthBeam enables medical personnel to access data such as patient identity, health status, allergies, and examination records quickly and securely, thereby supporting clinical decision-making.

The name HealthBeam combines the term “Health,” representing the medical domain, and “Beam,” referring to the transmission technology that facilitates platform communication. The platform is designed to reduce ineffective communication practices

in hospitals and minimize the use of paper-based records, including medical cards, examination forms, and handwritten notes [23-30]. HealthBeam follows environmentally sustainable principles by promoting digital documentation and integrates modern data protection mechanisms to ensure secure storage and communication [33-35]. This work presents the complete architecture and implementation details of the HealthBeam system, including:

- an iOS-based mobile application
- a web-based administrative portal
- a gateway service platform for processing patient-generated events
- a backend application acting as a mediator between all connected services

Beyond its technical development, HealthBeam aims to improve doctor–patient interaction, optimize emergency response time, and securely store medical information in centralized databases accessible only to authorized personnel. By enhancing data accessibility and workflow efficiency, the platform contributes to more sustainable and patient-centered healthcare practices.

Research Gap and Objectives

There exist several Real-Time Location Systems (RTLS). Most of them rely on high-accuracy but high-cost technologies or use proprietary hardware and extensive IT infrastructure. Literature of the last years demonstrates that there is a trend towards the usage of precision RTLS, which are expensive and can exceed the capacities of small or medium health facilities. There is a gap for lightweight, low-cost BLE-only systems specifically optimized for emergency alert propagation. This study defines the following research objectives and questions:

RQ1. To what extent can a low-cost BLE-only RTLS achieve sub-5-second emergency notification latency in a resource-constrained hospital environment?

RQ2. How does the deployment cost and infrastructural complexity of HealthBeam compare to commercial iBeacon-based and enterprise RTLS solutions?

RQ3. What system architecture is most effective for delivering reliable alert propagation using only consumer-grade BLE hardware and embedded IoT gateways?

COMPARATIVE ANALYSIS OF DIGITAL HEALTHCARE PLATFORMS

Table 1 depicts a comparative overview of selected state-of-the-art healthcare tracking solutions. Although several healthcare institutions have adopted digital tracking platforms, HealthBeam distinguishes itself by offering a lightweight Bluetooth Low Energy (BLE) gateway architecture, reduced infrastructure requirements, and a lower deployment cost.

Table 1. Analysis of State-of-the-Art Healthcare Tracking Solutions

System	Features	Strengths	Limitations	Cost per Bed (approx)	Licensing	HealthBeam Advantage
Leiden iBeacon [7]	Cardiology alerts via BLE cloud link	IoT-ready	High cost, EU-restricted	€80–€120	Proprietary	Open architecture, low-cost deployment can be used outside the EU
Sarasota App [8]	Indoor positioning & navigation	Simple UX, patient location tools	Reliant on GPS, poor accuracy indoors	€40–€70	Proprietary	Beacon-based multilayer tracking even without GPS
Dell Connected Health [9]	Enterprise RTLS, analytics dashboards	Strong integration with/ hospital EMR	Scales only in the enterprise tier	€300–€500	Proprietary	Deployable in SMEs, regional clinics, and scalable bottom-up
INSMA [25]	Vital-sign aggregation, ICU alerts	Clinically validated, full support	Vendor lock-in, subscription fees	€500+	Proprietary	Open component architecture, local hosting optional
Ekahau WiFi RTLS [34]	WiFi-based tags, hospital zone tracking	Uses existing WiFi infra	Requires enterprise WiFi controllers	€150–€250	Proprietary	Works without enterprise WiF
Quuppa BLE AoA System [35]	BLE Angle-of-Arrival antennas for sub-meter localization	Excellent precision	Requires calibrated antennas	€300–€600	Proprietary (AoA antennas)	No calibration needed

One example of IoT-enabled healthcare tracking is found at Leiden University Medical Centre (LUMC) in the Netherlands. To accelerate patient response time in the cardiology department, LUMC has implemented an iBeacon-based system on a cloud-supported Internet of Things (IoT) platform. This solution aims to help medical professionals locate patients more efficiently, ultimately supporting improved emergency care and faster treatment initiation [7].

Another prominent implementation is at Sarasota Memorial Hospital in Florida, where iBeacon technology has been integrated into the hospital's mobile application to enhance patient navigation and experience. The application has received positive user feedback; however, technical challenges emerged when attempting to combine GPS with indoor tracking, particularly regarding accurate floor-level recognition within the hospital building [8].

Connected Health, developed by Dell, seeks to provide faster, more accurate interactions between patients, healthcare providers, and digital systems. The platform integrates mobile devices, cloud analytics, and IoT connectivity to link patients with medical staff, connect clinicians to relevant clinical data, and network hospitals through a shared digital infrastructure. Using Real-Time Location Systems (RTLS), mobile applications, and interactive displays, the system improves patient guidance and enhances communication within hospital environments [9].

Further advancement is demonstrated by the ntegrated system for multimodal data acquisition and analysis (INSMA). This solution collects, stores, processes, and visualizes multiple physiological data streams from the Philips IntelliVue patient monitoring system. INSMA has been tested in the intensive care unit (ICU) of University Hospitals Cleveland Medical Center, where it has been used for continuous patient state tracking and clinical decision support through detailed analytics [25].

These and other similar solutions are very interesting and useful, but it is difficult to implement them in hospitals in Southeast Europe. Public hospitals in Southeast Europe, especially in the Western Balkans, have strict budget constraints and limited digitalization funding [33]. Therefore, HealthBeam was designed and implemented.

Additionally, Figure 1 illustrates the key developments in RTLS systems over time.

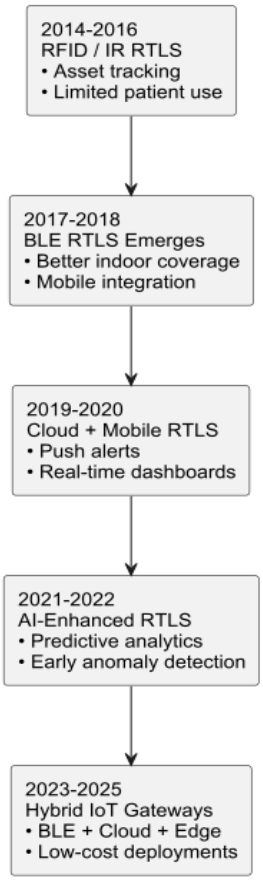

Figure 1. Evolution of Hospital RTLS Technologies (2014–2025)

Early systems (2014-2016) used RFID and infrared tags for asset tracking and had limited patient interaction. During the 2017-2020 period, commercial BLE-based RTLS systems were developed, enabling improved indoor localization and mobile-device integration. Since 2020, advancements in IoT gateways, cloud analytics, and AI monitoring, allowed the development of RSTL with more advanced capabilities like real-time clinical decision support and continuous patient-state tracking. Health beam is built using modern technologies like BLE, cloud messaging, and mobile-centric workflows while maintaining a lightweight deployment suitable for small and medium healthcare facilities.

Recent Advances in Hospital RTLS (2022–2025)

The trend in recent studies in hospital-oriented RTLS is towards higher accuracy, multimodal sensing, and hybrid architecture. Ultra-Wideband (UWB) technologies continue to dominate precision tracking. These technologies are suitable for biomedical sensing and real-time monitoring [27]. UWB is effective for high-precision indoor clinical applications [28]. These systems require proprietary antennas and cost-intensive installation, making them impractical for smaller healthcare institutions.

There is also the possibility of using multi-technology wireless solutions. Meftah et al. (2024) explored innovative ECG monitoring architectures using hybrid BLE, Wi-Fi, and LoRa networks, showing that wireless sensor fusion can improve coverage and reliability across different parts of a hospital [29]. But these systems are complex and have high energy consumption and infrastructure requirements.

Bluetooth Low Energy is affordable and easy to deploy. Recent studies demonstrate BLE's potential for clinical applications beyond navigation. For instance, BLE beacons are deployed for patient and asset tracking [30]. Similarly, proximity-network studies using BLE wearable tags have shown feasibility for monitoring healthcare worker interactions and patient contact patterns [31]. However, BLE-based localization is sensitive to environmental variability and is less accurate compared to UWB.

RFID and UHF-based tracking systems continue to serve as low-cost alternatives for basic patient and asset management tasks. These technologies can be used in resource-limited health sectors, but they need manual scanning for fixed radars and this is a limitation for real-time continuous tracking. [32]

Positioning HealthBeam: In contrast to recent systems that prioritize sub-meter accuracy or hybrid multi-radio architectures, HealthBeam intentionally adopts a BLE-only, RSSI-based event-driven approach. This design ensures low cost, rapid deployment, and emergency alert responsiveness. The recent literature reinforces that while high-accuracy solutions exist, they remain financially or technically inaccessible to many hospitals. HealthBeam is more suitable for emergency-notification scenarios rather than precise indoor localization.

HEALTHBEAM'S LOGIC AND PROGRAMMING

HealthBeam, as presented in Figure 2, is centered on a Cloud Service platform that provides independent operation among all premises integrated into HealthBeam. The server and the database are hosted in the cloud. Healthcare facilities have their own medical users, patient devices, and BLE-based iBeacon tags on premises. The events that are generated by the patients are forwarded by the Gateway and are handled by the HealthBeam server.

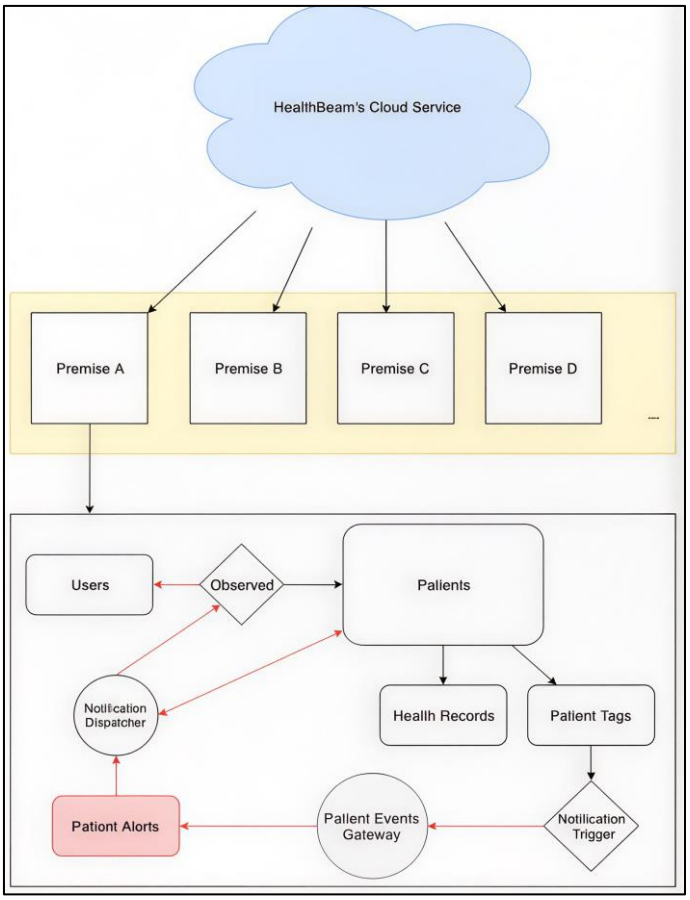

Figure 2. HealthBeam's logic flow of data

On that account, it should be noted that the term "premise" signifies a medical establishment, as well as all the medical personnel and patients located in that establishment. Although the Cloud Service platform uses a shared database containing all the premises' information, there is a well-defined separation between the premises, and thus, a premise cannot access or interact with any information not explicitly related to it. As far as each separate premise is concerned, several operations take place within its boundaries, initiated and administered by HealthBeam. One of those operations is facilitating the users' interaction, namely the interaction between patients and medical providers. Each user has a specific and strictly individual account type, establishing his or her data access controls. Additionally, users can subscribe and later unsubscribe from

patient-generated events. Once a subscription for an event occurs, notifications are dispatched to all observers, allowing them to respond. Moreover, users can assign patient tags to the patients by applying iBeacon external devices. The tags can act as personal patient identification and generate the occurrence of events once the patient's condition worsens. Those events are to be intercepted by the Gateway Service platform, which directs them to the HealthBeam Server. The latter's task is to dispatch notifications related to the event' type, making it possible for the medical personnel to respond promptly.

Users sharing the same premise can create and modify the patients' profiles. The profiles include, but are not limited to, health records. Once generated by the users, all relevant data is accessible not only through the mobile application but also through the web-based platform. Depending on the individual user account type, this information can be either subject to modification or only viewed. After a user responds to an alert triggered by a patient, a record signifying the response and the relevant information associated with it is stored in the Server database. The stored data can currently be viewed only on the web-based platform, and it cannot be modified by any user regardless of her or his account type. Below is an overview of the programming languages and principles used in HealthBeam's development across all platforms 1011. Swift is used by the iOS mobile application, the server-side, as well as the web-based front-end, aided by the Vapor framework 12. On the other hand, Python and Node.js, in conjunction, are used for the gateway's service platform implementation 13.

Swift is a language used for writing software applications for phones, peripheral devices, desktops, and servers 14. In the Healthcare application, Swift is used for the iOS mobile application and the backend API services. It enables encrypted communication and real-time user interactions through Vapor's server-side framework. Both the server-side and client-side implementations of HealthBeam use Swift version 5.10.

Python is a general-purpose scripting language that has a wide range of applications, ranging from web development and computation services to desktop software. Here, it is used to orchestrate BLE scanning and message packaging on the Gateway Service Platform. These tasks are implemented in Python: reading BLE advertisement frames, extracting UUID/Major/Minor values, timestamping events, and preparing the payload for RSA encryption before transmission.

Node.js is an open-source and cross-platform language. In HealthBeam, we use Node.js in a limited way within the gateway to invoke the noble BLE library that is needed for continuous scanning. The BLE device discovery is managed by Mode.js, whereas the execution pipeline and the secure communication layer are handled by Python.

The Raspberry Pi 3B+ serves as the embedded scanning gateway. In HealthBeam, it is configured to (1) detect nearby iBeacons, (2) generate encrypted alert messages, and (3) forward them to the cloud server.

CLIENT-SIDE IMPLEMENTATION

Beacons at Leiden University Medical Centre; to be sure that the patients in the cardiology department of the Netherlands are treated as fast as possible, they have improved their chances of survival and recovery with the current technology. Leiden University Medical Center has been using iBeacon technology established on a cloud-based Internet of Things (IoT) platform **Error! Reference source not found.**.

HealthBeam implements an iOS application representing the platform's client side. The application is written using the Swift programming language and supports iOS versions 16.0 or later. The mobile application is planned to be deployed on Apple's App Store, making it public and available for anyone to install. However, unless the user has already created an account by their premise's administration, they will not be able to operate the software. The iOS app is dependent upon HealthBeam's server for its normal functioning. Only a small batch of data is stored on the device locally, which is required to be readily available for use, such as the user's personal information and their discovery regions. This data is constantly updated. Due to this fact, the application is always synchronized with the latest chunks of data. If additional platforms supporting HealthBeam's logic are implemented in the future, they will be consistent, since most data is shared by the server. The application supports receiving and handling push notifications signifying alert events. It also offers the location management capabilities provided by Apple, which allows it to monitor devices conforming to the iBeacon protocol. These devices represent patient tags used for patient identification as well as for generating alerts.

Most of the services provided by the server require authorization. After a successful login, the client receives an access bearer token stored in Apple's keychain, which represents their secure data storage. This process is aided by the SSKeychain library for Swift. If this token is revoked, the user will get unauthenticated and presented with the login screen. It should be noted that although this access token never expires, it could get revoked if the account holder is authenticated on another platform. Upon every launch, the application sends its device token to the server. This is a unique identifier required by the back end to dispatch push notifications for the particular iOS device.

The application provides a simple, intuitive, and functional user interface for operating with it. Ease of use is the main client-side focus. Currently, only iPhone devices are supported by the application. Nevertheless, iPad devices are considered for future phases. The application supports the following main features:

- User authentication
- User authorization revocation
- Subscribing and unsubscribing to patients
- Patients search and overview
- Patient creation, deletion, and update
- Assigning and unassigning patient tags to patients
- Health records overview
- Health records creation, deletion, and update for patients

- Pending alerts overview
- Pending alert response mechanism
- Legal aspects overview

The application is implemented using the Clean-Swift VIP (View Interactor Presenter) architecture 15. It incorporates a variety of design patterns, including dependency injection, protocol-oriented programming, facade-layer abstraction, and the singleton pattern [19].

HealthBeam's representation of patient health records follows structured-data principles compatible with the ISO/IEEE 11073 Personal Health Data standard [26].

HealthBeam's iOS client uses Google Analytics for Firebase 16. This software development kit gives insight into the way people interact with mobile applications. The service automatically captures numerous events and user properties. It further allows for creating custom events to measure specific aspects required by business logic. Once the data is captured, it is accessible over a web-based console. This console provides detailed information about the data collected, including summary data, such as active user statistics and demographics. Analytics describe how users operating the mobile application behave. It allows the making of informed decisions about how to market their product. SDK can perform custom analysis by joining the extracted data with other sources, such as the BigQuery service. It allows for more complex analysis, e.g., querying large data sets and joining multiple data sources.

SERVER-SIDE IMPLEMENTATION

HealthBeam services, which are provided for both the iOS client and the web-based front-end, are written in Swift, using the 4th iteration of Vapor's framework. The focus of the server is to host both API services and a shared database, which can be operated synchronously among multiple platforms. The server side allows for the creation of user profiles, which could operate the application throughout every platform it is implemented on. This process is limited by predefined business logic, allowing only users with a specific account type to be able to create new user profiles.

The database integrated with the HealthBeam server-side is PostgreSQL. The server-side implementation relies on two operational environments for both development and production purposes, which point to different database entities. All sessions established between the server and third parties require authentication unless otherwise specified. The iOS client uses the Bearer authentication model while the web-based front-end uses cookie-based authentication. All server-side services are hosted by Heroku (a platform operating as a service provider), which generates AWS databases for operation within the cloud. HealthBeam server enforces the usage of the HTTPS protocol when communicating with external parties. This requires a valid TLS certificate for the domains linked to the hosted environments, which is validated during every session initiation using the handshake mechanism that is in the session layer of the OSI model. The server also supports the ability to send multiple push notifications for different events. To send push notifications to iOS

devices, the server is required to obtain a device token from the particular device. When a specific event is triggered, the back end sends a push notification request to the APNS server, a server maintained and owned by Apple, which acts as a mediator between the server and the iOS push recipient, validating every push notification request. The device token is held in the server's database if the corresponding user is authenticated. Once the user logs out of his account, the device token is deleted from the server's database. For the server to support sending push notifications, a certificate issued by Apple's development platform is required, which is encrypted with a password. During the validation stage performed by the APNS server, the certificate gets decrypted and validated as well.

Besides the iOS client and the web-based front-end, the hosted services are reachable by the gateway embedded platform, which is responsible for collecting events generated by the BLE emitting iBeacon devices, as well as analyzing the events and notifying the server. Unlike mobile and web applications, the gateway does not rely on an authorization mechanism, since ideologically, no user is associated with the gateway. Instead, the gateway and the server use an asymmetric encryption mechanism for communication, which guarantees that no data will be exposed to malicious middleware.

GATEWAY SERVICE PLATFORM

The Gateway service platform is responsible for observing events generated by iBeacons, which possesses a proximity ID that is configured to be discoverable by the gateway's BLE scanner [36-38]. The platform is developed mainly using Python combined with Node.js. The latter is injected to begin the process of initialization of the noble library used for discovering devices. Those devices are supposed to conform to the iBeacon protocol as defined by Apple. Furthermore, the gateway can continuously monitor events generated by BLE devices. However, currently, only the alerting mechanism has been implemented in HealthBeam. The hardware used for implementing the gateway is a Raspberry Pi model 3B+ with the following specifications:

- 1.4GHz 64-bit quad-core processor
- Dual-band wireless LAN
- Bluetooth 4.2/BLE
- Power-over-Ethernet support (with separate PoE HAT)

The Raspberry Pi is a series of single-board computers developed by the Raspberry Pi Foundation, designed to provide affordable yet powerful computing capabilities. Its low cost, compatibility with multiple operating systems—including Linux-based distributions such as Raspbian and Windows IoT—and integrated General-Purpose Input/Output (GPIO) interfaces make it a versatile tool for physical computing and rapid prototyping. For HealthBeam, the use of a Raspberry Pi as the core of the gateway service platform is crucial because it offers a scalable, energy-efficient, and cost-effective hardware solution, making the system suitable not only for large medical facilities but also for smaller clinics with limited budgets. The device operates with a stable 5 V / 2.5 A DC power supply,

ensuring reliable 24/7 functionality, which is essential for continuous patient monitoring and communication. During its initial configuration, the Raspberry Pi is assigned a unique gateway identifier along with defined discovery regions or supported proximity identifiers, allowing precise BLE-based device detection. This configuration enables HealthBeam to accurately locate and track patients' devices, reinforcing the platform's real-time monitoring goals and improving healthcare workflow efficiency [17].

iBeacon Technology

The Gateway Service Platform is configured to intercept only devices conforming to the iBeacon protocol defined by Apple, see Table 2. Such devices can be powered using coin cell batteries for a month or even years. Moreover, they can also be powered externally for extended periods. Consequently, iOS devices can generate iBeacon advertisement data. However, this functionality is limited. An iBeacon advertisement provides the following information via Bluetooth Low Energy (BLE):

Table 2. Beacon characteristics

Field	Size	Description
UUID	16 bytes	UUID is application application-specific identifier, which should depend on the use cases it is designated for.
Minor	2 bytes	It specifies a local subdivision for the represented iBeacon
Major	2 bytes	It specifies a global subdivision for the represented iBeacon

The UUID, major and minor values provide identifying information for the iBeacon. This information is hierarchical with the major and minor fields, allowing for subdivision of the identity established by the UUID. UUIDs can be generated by using the 'uuidgen' utility provided by the OS X operating system or by using the NSUUID Foundation class embedded in Swift. The examined iBeacons rely on BLE, and this is the reason why it is a hardware-dependent technology requiring an iPhone 4S (or later), iPod touch (5th generation), iPad (3rd generation or later), or iPad mini. Moreover, iBeacons are devices that repeatedly transmit a singular signal that other devices can discover and interpret. Instead of emitting visible light, those devices broadcast a radio signal that is comprised of a combination of letters and numbers transmitted on a regular interval of approximately 1/10th of a second. Hardware with a Bluetooth module, such as a smartphone, can "see" a beacon once it is within its range. Some of the additional iBeacon features developed by third-party vendors include the following:

- Modular and open IoT platform
- Adaptive location and telemetry engines
- On-device and edge processing
- Blockchain beacon firmware

Healthbeam uses iBeacons supplied by a startup vendor based in Poland called Kontakt 18.

The HealthBeam's gateway service platform architecture and its interaction with its server are illustrated in Figure 3. The Raspberry Pi gateway scans for iBeacon signals. These signals are encrypted using an RSA key before they are transmitted. The server decrypts and validates. All active observers are notified by the server.

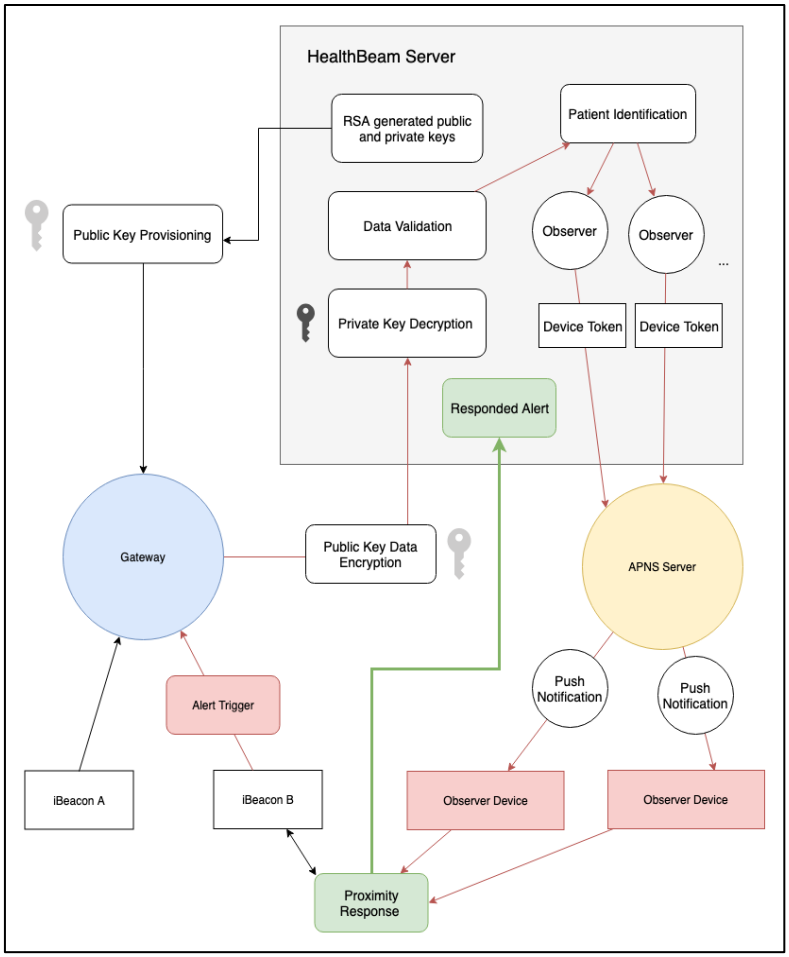

Figure 3. Architectural view of the Gateway Service Platform

The HealthBeam server generates a pair of public and private keys using the RSA encryption algorithm. The public key is provisioned to the gateway by the server. When a button is pressed on an iBeacon device, it generates an additional advertisement packet intercepted by the gateway service platform. It bundles the following data: (1) its code identifier, (2) a date parameter specifying the current data and the discovered (3) minor and (4) major iBeacon values.

This data gets encrypted using the provided public key and is then passed to the server. Afterward, the back end decrypts the data using its private key, and if it is unable to do so, a "bad request, error code: 400" is fired. As a result, the data gets validated. Each parameter is extracted from the bundle, and if the date specified by the gateway is older than 10 seconds in comparison to the current date, the above-mentioned error is fired. Then the server tries to identify the patient tag record from the extracted minor and major value

parameters. Using this record, the server tries to locate the patient associated with it. If no patient is found, the same error is fired.

All errors described are processed by HealthBeam's Middleware. When the patient is identified, HealthBeam retrieves all their observers and their "Device" records. They contain the device token value used by the HealthBeam server to make requests to the APNS server, demanding that push notifications be dispatched to the observing users' devices. When an observing user receives the alert push notification on their iOS application, they can respond to it by placing their device in immediate proximity to the corresponding patient's tag. When this happens, a web request to the server is made. The request specifies that the alert has been responded to, because of which all other observers are notified of this response.

The Gateway service is initialized with a two-step pre-configuration. It requires a code identifier string value, specifying the Gateway record stored in the server's database. The Gateway record provides information regarding the scanning hardware's whereabouts so that when a patient alert is triggered and intercepted by the gateway, the observed users are notified not only about the patient who needs immediate medical assistance but also about the location from which the alert has been generated. The gateway also requires a list of proximity IDs in a standard broadcasting packet represented by string UUIDs. These identifiers specify which iBeacons can be discoverable by the gateway. The input gets validated, and if the strings are not convertible to a valid UUID format, they get ignored.

Message Encryption

The "encryptMessage" encodes a string value provided as a parameter. It loads the public key provisioned by the server into the memory and serializes it as a variable parameter. This parameter is not only used for the actual encryption but is also necessary to explicitly specify the decryption algorithm padding, which, in HealthBeam's case, is PKCS1v15. The term padding refers to several distinct methodologies, which include adding portions of data to the beginning, middle, or end of a message before the encryption process. The decrypted message recipient should perform the decryption using the very same padding. The encrypted "ciphertext" parameter is in a binary string format. It is converted into a hexadecimal string format so that it can be interpreted by the server more easily.

```
    #Encrypt a message using the provided public key
def encryptMessage(message):
   with open("publicKey.pem", "rb") as key_file:
      public_key = serialization.load_pem_public_key(
         key_file.read(),
         backend=default_backend()
      )
   ciphertext = public_key.encrypt(
      message,
```

```
    padding.PKCS1v15()
  )
  return binascii.hexlify(ciphertext).decode()
```

SYSTEM MODEL AND ARCHITECTURAL DESIGN

Component Diagram

Figure 4 shows the system architecture that includes a backend hosted in the cloud, mobile and web clients, the PostgreSQL database, and the gateway layer.

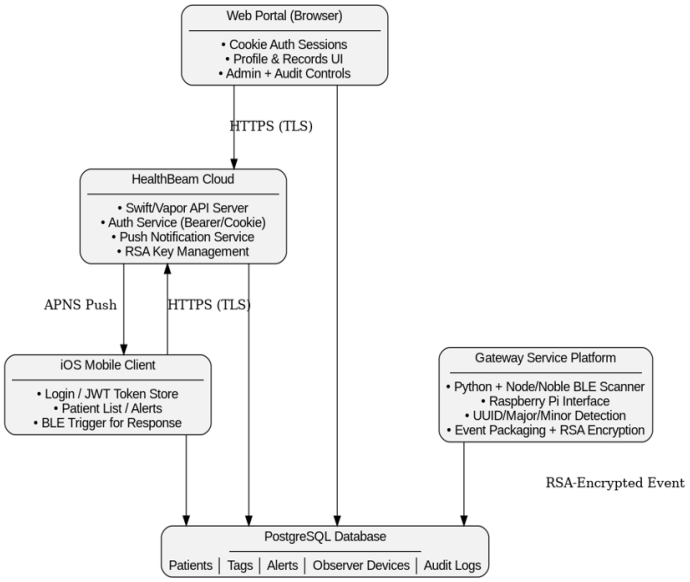

Figure 4. UML Component Architecture of the HealthBeam System

Database Schema

Figure 5 shows the Entity-Relationship diagram for the database schema. The Gateway nodes represent the Raspberry Pi devices that scan BLE frames, encrypt data, and submit alerts to the cloud.

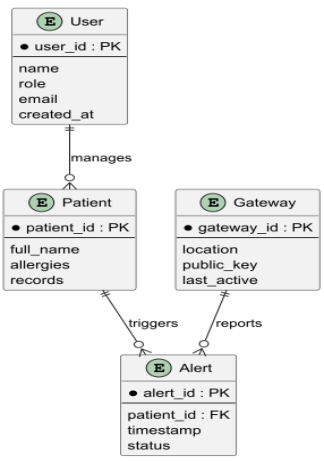

Figure 5. Entity-Relationship Diagram for the HealthBeam Database Schema

METHODOLOGY

The methodology we have used to implement the HealthBeam system aligns with the INCOSE lifecycle principles and has four phases: (1) requirements specification, (2) system design, (3) implementation, and (4) verification and validation.

1. Requirements specification: The functional and non-functional requirements were collected through discussions with personnel in the healthcare system. The design documents were created using Confluence [22].
    a. Functional:
        i. View patient records remotely
        ii. Tag patients via iBeacon
        iii. Receive alerts in real time
        iv. Respond to events
    b. Non-functional:
        i. Secure data transport (RSA/TLS)
        ii. Usability for non-technical staff
        iii. Notification latency less than 1 second
2. System design: The requirements were converted to architectural blocks and diagrams using UML and tools like Figma, Notion. The large architectural blocks were then used to create user stories in accordance with the INVEST criteria [24]. All stories were available in Jira and were grouped into epics.
3. Implementation: The implementation followed the sprint cycles of 3 weeks. After each sprint, the new features were deployed and validated. GitHub Actions were used for automatic deployment [21].
4. Verification and Validation (V&V): We used several types of testing to ensure the correctness of the system:

Table 3. Testing methods used for V&V

Testing method	Purpose
White-box testing (Junit/Mockito [23])	Testing single components in isolation
Black-box testing (Postman/Karate)	Validate external behavior
Load testing (JMeter)	Evaluate scalability

RESULTS AND EVALUATION

We performed two quantitative analyses to evaluate the HealthBeam system: (1) alert-latency evaluation under increasing simulated patient volume, (2) gateway hardware performance, see Table 4.

Latency Evaluation: 50 patients were simulated. Each of them had an iBeacon ID. Alerts were triggered in bursts, and the measurements were sampled over 100 trials.

Table 4. Latency evaluation results

Load Level	Alerts/sec	Mean Latency (s)
Low Load	5	2.1
Medium Load	15	2.5
High Load	25	3.1
Very High Load	50	4.7

The latency results show that HealthBeam performance is stable even in cases where there is a high load of alert traffic. At the highest simulated load of 25 alerts/second, the mean latency is 3.1 seconds, which means that the system scales linearly compared to medium and low loads.

Raspberry Pi Gateway Performance: The efficiency of the gateway was measured on a Raspberry Pi 3B+ (1.4GHz ARM Cortex-A53, 1GB RAM) with BLE scanning active. For the experiments, the packets were transmitted at increasing frequency. Results show that we are far from reaching the hardware limits, even when there are 100 events/minute.

The results displayed in Table 5 show that even at 100 events/min, the CPU utilization is below operational limits, leaving open the possibility for future extensions such as more complex on-device processing. RAM consumption is below 35%. This means that the current gateway hardware is appropriate for small or medium-sized healthcare facilities without the need to use more expensive embedded systems.

Table 5. Measurements using a Raspberry Pi 3B+ (1.4GHz ARM Cortex-A53, 1GB RAM).

Event Intake Rate (events/min)	CPU Load (%)	RAM Load (%)
20	14%	23%
60	19%	30%
100	27%	34%

DISCUSSION

The evaluation of HealthBeam demonstrates that a lightweight BLE-only RTLS can achieve reliable emergency-alert propagation with latency that is within clinically acceptable limits. The average end-to-end delay is less than 4.7 seconds. This delay addresses RQ1 and shows that a 5-second responsiveness is possible even when we have an increased alert volume. BLE RSSI measurements are not suitable for sub-meter localization, but are appropriate for event-driven alert activation, and this is the primary purpose of the Healthbeam system.

Regarding RQ2, the results indicate a cost and complexity advantage when compared to commercial iBeacon, UWB, or enterprise RTLS systems. Healthbeam does not require proprietary hardware, on-site calibration, or subscription platforms. The ability to operate

under inexpensive BLE beacons, an embedded gateway, and open-source software components reinforces the idea that this system can be used in environments with limited budgets.

The tested architecture for RQ3 demonstrates that the BLE, Gateway, Cloud, and Mobile architecture is reliable for alert propagation. Under simulated peak loads, the usage of the CPU and the RAM of the Raspberry Pi gateway is less than 30% and 35% respectively.

Limitations

First, the mobile client is currently implemented only for iOS. This restricts accessibility in regions where Android devices are more commonly used and limits generalizability.

Second, in the proposed architecture, there is only one gateway deployment. This can lead to a single point of failure and a potential bottleneck. Although our test results show low resource utilization, in the future, multiple gateways may be needed, and there will be a need for coordination and load balancing.

Third, the HealthBeam system does not offer continuous monitoring to capture vital-sign data. Its functionality is limited to presence detection and emergency alert transmission.

Fourth, privacy and data protection should be considered carefully before clinical usage, especially around BLE beacon spoofing or replay attacks.

Future work

Future work should be directed at addressing the limitations described above.

The mobile client should be implemented to support Android devices. Multi-gateway architecture needs to be designed to provide redundancy and load balancing. A user study with healthcare staff is planned to evaluate usability, acceptance, and workflow impact. Finally, deployment in a real clinical environment will allow measurement of real-world latency, robustness, and operational constraints.

ETHICAL & GDPR COMPLIANCE STATEMENT

No real patient data were used to evaluate the HealthBeam system. All tests were performed using synthetic records and simulated beacon identifiers, so no medical-ethics approval was required.

The system architecture has been designed with GDPR principles in mind. Communication channels use TLS transport encryption. RSA public-key encryption is used to transmit sensitive identifiers, ensuring confidentiality. All personal data stored by HealthBeam is processed under data minimization, purpose limitation, and explicit access-control policies.

CONCLUSION

The development of the HealthBeam application has been accomplished through the systematic use of the platforms and technologies examined in this study. As part of the development lifecycle, a functional prototype was produced to facilitate testing, validation, and demonstration, which allows assessing the system's usability, performance, and potential impact. The prototype is ready for integration into real-world healthcare infrastructures.

Under typical load, mean alert latencies are between 2.1 and 3.1 seconds, and stable gateway utilization is below 30%. These results confirm the feasibility of a BLE-only architecture for emergency alerting in resource-constrained environments. The findings show that a low-cost RTLS based exclusively on BLE beacons and a single embedded gateway can reliably support emergency-notification workflows without requiring specialized infrastructure.

In summary, the HealthBeam platform represents a meaningful contribution to the ongoing digital transformation of healthcare systems. With further development and integration, it is well-positioned to serve as a robust, scalable, and innovative solution for enhancing patient care, clinical workflows, and the broader organizational efficiency of healthcare institutions.

AUTHOR CONTRIBUTIONS

Conceptualization, E.V., I.P., N.A., and K.S.; methodology, E.V., I.P., and N.A.; software, N.A. and I.P.; validation, E.V., I.P., and K.S.; resources, N.A. and I.P.; writing - original draft preparation, E.V., I.P., N.A and K.S.; writing - review and editing, E.V., I.P., N.A., K.S., and A.B.; supervision, E.V., K.S., and A.B. All authors have read and agreed to the published version of the manuscript.

ACKNOWLEDGEMENT

This paper is part of the research project titled "*Ndërtimi i një modeli informatik parashikues i tipit machine learning për trajtime terapeutike*" (*English: Creating a digital prediction model for therapy treatment using machine learning*) funded by National Agency for Scientific Research and Innovation (NASRI) in Albania.

CONFLICT OF INTERESTS

The authors declare no conflicts of interest associated with this publication

REFERENCES

1. Global Observatory for eHealth. Available from: https://www.who.int/goe/en/ (Accessed date: December 15, 2024)

2. Adewole, K.S., Akintola, A.G., Jimoh, R.G., Mabayoje, M.A., Jimoh, M.K., Usman-Hamza, F.E., Balogun, A.O., Sangaiah, A.K., Ameen, A.O. Chapter Five - Cloud-based IoMT framework for cardiovascular disease prediction and diagnosis in personalized E-health care. *Intelligent IoT Systems in Personalized Health Care*, **2021**, 105–145.
3. Alshehri, F., Muhammad, G. A Comprehensive Survey of the Internet of Things (IoT) and Edge Computing in Healthcare. *IEEE Access*, **2021**. *PP*, 1-9.
4. Furht, B., Agarwal, A. Handbook of Medical and Healthcare Technologies, Springer, **2013**.
5. Wachter, R. The Digital Doctor: Hope, Hype, and Harm at the Dawn of Medicine's Computer Age. McGraw-Hill Education, **2015**.
6. Thimbleby, H. Technology and the Future of Healthcare. *Journal of Public Health Research*. **2013**, *2*(3), 1-8.
7. Bluetooth Beacons Market Analysis 2014-2025 Available from: https://ixtenso.com/technology/bluetooth-beacons-market-analysis-2014-2025.html (Access date: September 25, 2024).
8. Beacons light the way at Sarasota Memorial Hospital. Available from: https://www.digitalcommerce360.com/2015/04/28/beacons-light-way-sarasota-memorial-hospital/. (Access date: September 15, 2025)
9. Connected Health The Path to Better, More Integrated Care and Health Outcomes, DELL Available: https://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell-Connected-Health-Whitepaper-final.pdf (Access date: September 15, 2025).
10. Clean Swift. Clean Architecture. Available from: https://clean-swift.com. (Access date: January 30, 2025)
11. Kelmendi, Dh. SOLID principles made easy. Available from: https://medium.com/@dhkelmendi/solid-principles-made-easy-67b1246bcdf (Access date: January 15, 2025
12. Condon, T., Nelson, T., Schwartz, J., Wright, L. Server-Side Swift with Vapor: Building Web APIs and Web Apps in Swift. Available from: www.raywenderlich.com (Access date: November 30, 2024).
13. Node.js. API Reference Documentation. Available from: https://nodejs.org/en/docs (Access date: September 30, 2025).
14. Kugler, F., Eggert, D., Core Data: Core Data best practices by example, from persistency to multithreading and syncing. (3rd ed), **2017**.
15. Martin, C. Robert., The Clean Architecture (The Clean Code Blog, 2012). Available from: http://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html. (Access date: September 14, 2025)
16. Google Analytics for Firebase. Available from: https://firebase.google.com/docs/analytics. (Access date: September 04, 2025)
17. Monk, S. Programming the Raspberry Pi: Getting Started with Python (3rd. ed.), McGraw-Hill Education, **2021**.
18. Kontakt iBeacons Provider. Available from: https://kontakt.io. (Access date: January 30, 2025)
19. Robert., M.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series), **2017**.

20. Healthcare Technology. Available from: https://www.ibm.com/think/topics/healthcare-technology. (Access date: January 12, 2025).
21. Automate your workflow from idea to production. Available from: https://github.com/features/actions. (Access date: September 30, 2025).
22. The AI-powered Jira: from teams to dreams. Available from: https://www.atlassian.com/. (Access date: September 25, 2025)
23. Tasty mocking framework for unit tests in Java. Available from: https://site.mockito.org/. (Access date: November 25, 2025).
24. Cohn, M. User Stories Applied. Publisher: Addison-Wesley Professional. **2004**.
25. Sun, Y., Guo, F., Kaffashi, F., Jacono, F.J., DeGeorgia, M., Loparo, K.A. INSMA: An integrated system for multimodal data acquisition and analysis in the intensive care unit. *J. Biomed Inform*. **2020**, 106, 103434.
26. Generic Health Sensor Design and Implementation Guide. http://standards.ieee.org/ieee/11073-10206/10311/. (Access date: September 4, 2025)
27. Jing, F., Liang, J., Wang, Y., Chen, P., Harmonics and intermodulation products-based fuzzy logic (HIPBFL) algorithm for vital sign frequency estimation using a UWB radar. *Expert Systems with Applications*. **2023**, 228, 120294.
28. Tiberi, G., Ghavami, M. Ultra-Wideband (UWB) Systems in Biomedical Sensing. *Sensors (Basel)*. **2022**, *22*(12), 4403.
29. Meftah, E.H., Benmahmoud S., and Rabehi, A. Innovative Wireless Solutions for Real-Time ECG Monitoring: Leveraging BLE, Wi-Fi, and LoRa Technologies. International Conference on Telecommunications and Intelligent Systems (ICTIS), Djelfa, Algeria, **2024**, p. 1-6,
30. Skýpalová, E., Boroš, M., Loveček, T., Veľas, A. Innovative Indoor Positioning: BLE Beacons for Healthcare Tracking. *Electronics* **2025**, *14*, 2018.
31. Curtis, S.J., Rathnayaka, A., Wu, F., Al Mamun, A., Spiers, C., Bingham, G., Lau, C.L., Peleg, A. Y., Yuce, M.R., Stewardson, A.J., Feasibility of Bluetooth Low Energy wearable tags to quantify healthcare worker proximity networks and patient close contact: A pilot study. *Infection, Disease & Health*. **2022**, *27*(2), 66-70.
32. Abugabah A., AL Smadi, A., Houghton, L. RFID in Health care: A review of the real-world application in hospitals. *Procedia Computer Science*, **2023**, 220, 8-15.
33. European Health Management Association (EHMA), Health Management in South Eastern Europe challenges and opportunities. Available from: https://ehma.org/app/uploads/2022/12/Health-Management-in-South-Eastern-Europe-challenges-and-opportunities.pdf (Access date: September 5, 2025).
34. Accurate Real-time Location Tracking (NEC IP DECT and Ekahau RTLS). Avaiable from: https://uk.nec.com/pdf-download/nl/10-054-01_NEC_SS_ENG_Real-time_Location_Tracking.pdf (Access date: October 8, 2025).
35. Save Lives with Real-time Location System in Healthcare. Available from: https://www.quuppa.com/segments/healthcare/ (Access date: October 8, 2025).
36. Kwok, C.Y.T., Wong, M.S., Griffiths, S., Wong, F.Y.Y., Kam, R., Chin, D.C.W., Xiong, G., Mok, E. Performance Evaluation of iBeacon Deployment for Location-Based Services in Physical Learning Spaces. *Appl. Sci.* **2020**, *10*, 7126.

37. Wu, X., Shen, R., Fu, L., Tian, X., Liu, P., Wang, X. iBILL: Using iBeacon and inertial sensors for accurate indoor localization in large open areas. *IEEE Access* **2017**, *5*, 14589–14599.
38. Griffiths, S., Wong, M.S., Kwok, C.Y.T., Kam, R., Lam, S.C., Yang, L., Yip, T.L., Heo, J., Chan, B.S.B., Xiong, G., et al. Exploring bluetooth beacon use cases in teaching and learning: Increasing the sustainability of physical learning spaces. *Sustainability* **2019**, *11*, 4005.