Computer Systems Science & Engineering
DOI:10.32604/csse.2021.014376
Article

Tech Science Press

A Model Transformation Approach for Detecting Distancing Violations in Weighted Graphs

Ahmad F. Subahi*

Department of Computer Science, University College of Al Jamoum, Umm Al Qura University, Makkah, 21421, Saudi Arabia
*Corresponding Author: Ahmad F. Subahi. Email: AFSubahi@uqu.edu.sa
Received: 17 September 2020; Accepted: 24 October 2020

Abstract: This work presents the design of an Internet of Things (IoT) edge-based system based on model transformation and complete weighted graph to detect violations of social distancing measures in indoor public places. A wireless sensor network based on Bluetooth Low Energy is introduced as the infrastructure of the proposed design. A hybrid model transformation strategy for generating a graph database to represent groups of people is presented as a core middleware layer of the detecting system's proposed architectural design. A Neo4j graph database is used as a target implementation generated from the proposed transformational system to store all captured real-time IoT data about the distances between individuals in an indoor area and answer user predefined queries, expressed using Neo4j Cypher, to provide insights from the stored data for decision support. As proof of concept, a discrete-time simulation model was adopted for the design of a COVID-19 physical distancing measures case study to evaluate the introduced system architecture. Twenty-one weighted graphs were generated randomly and the degrees of violation of distancing measures were inspected. The experimental results demonstrate the capability of the proposed system design to detect violations of COVID-19 physical distancing measures within an enclosed area.

Keywords: Model-driven engineering (MDE); Internet-of-Things (IoTs); model transformation; edge computing; system design; Neo4j graph databases

1 Introduction

Indoor tracking and monitoring systems have witnessed increased interest recently because of potential technological advancements in the domain of the Internet of Things (IoT), wireless networks, and pervasive computing. As a result of the current trend of artificial intelligence (AI) and computer vision, different localization, positioning, and tracking techniques and mechanisms have been introduced, providing indoor localization services to users. From a data engineering perspective, AI and machine learning systems that deal with a massive amount of data, or "big data", require appropriate architecture for a data pipeline system design to collect, clean, transform, and route the source data, which are captured from multiple resources, to the backend data storage of the system [1,2]. The data pipeline can be designed in the form of layered architecture consisting of an ingestion layer and transformation layers. In the context

of this work, data are collected from various IoT devices and represented in an initial format at the ingestion layer. Then, a number of transformations, including filtering, aggregation, and enriching steps, are applied to the source data forming the final format of the target storage (relational-based, document-based, or graph-based data) [1,2]. These stored data are retrieved using several predefined queries that report results to users.

Model-driven engineering (MDE) is a software development methodology that considers—rather than general purpose codes—the use of models constructed using appropriate modelling languages to develop software systems. Model transformation mechanisms and code generators are two key principles or components in the MDE software system's development lifecycle. Model transformation plays important roles in database reengineering, including data migration, schema construction, and normalization [3,4].

This study considers a model transformation approach, which is an aspect of the overall proposed system design, to generate the executable graph data model and related queries from the sensor-based data captured in the real-world situation. This research adopts MDE principles to introduce a lightweight model transformational approach that detects the degree of violation of social distancing measures based on personal space in groups. The mathematical foundation of the presented system is a complete weighted graph model of a restricted network size consisting of groups of individuals who represent a real-world crowd in an indoor scenario.

The system design supports capturing the status of individuals in a group according to determined distancing measures. The system analyzes the distances between group individuals and decides the degree to which the social distancing measure is being violated in a covered place. When a violation is detected, the system sends a message to the authorized person of the area to take the appropriate corresponding action when required. Because various applications of graph theory exist in the domain of crowd control systems (e.g., [5–9], a wireless sensor network (WSN) technology based on Bluetooth Low Energy (BLE) was considered the design choice for the infrastructure of the proposed system design. BLE provides an efficient method of transferring data in a WSN using low energy, which helps mobile Bluetooth beacons produce data for extremely long durations.

Additionally, as a result of ongoing global spread and the absence of a known treatment, the cumulative total of confirmed cases of coronavirus (COVID-19) reached 6 million by June 2020. Consequently, various Internet of Things (IoT)–based initiatives have been introduced using energy efficient communication technologies and smart devices to automate physical distancing detection, tracking, and monitoring [4,10]. Hence, a case study on detecting violations of the COVID-19 physical distancing measures was designed as proof of concept to evaluate the introduced system architecture.

The reminder this paper is organized as follows. Section 2 briefly highlights the techniques and technologies related to current trends that might be considered when designing an automated solution for detecting the implementation of physical distancing measures based on IoT and wireless networks. Section 3 theoretically presents a weighted graph-based model for detecting violation of social distancing measures, including the model's mathematical foundation and implementation. Section 4 discusses our vision of the architecture of the system's major components based on MDE model transformation, the IoT, and edge computing. Section 5 highlights the adopted strategy of the model-to-code transformation approach as a middleware layer of the proposed data pipeline architecture. Section 6 evaluates the proposed system design using a case study and a basic discrete-time simulation, based on real-world actions taken by international retail supermarkets in selected countries. Finally, Section 7 concludes our proposed work.

2 Related Techniques and Technologies

This section briefly highlights the domains related to the implementation of COVID-19 social distancing measures using the theories and applications of computer science from five main perspectives: Application

domains, device/sensor technologies, communication technologies, overall system architectures, and background theories or algorithms.

To explore the internet technology (IT) and computer-based approaches and systems most relevant to the automatic implementation of COVID-19 distancing measures, a literature survey was conducted using different research databases available in the following globally recognized academic collections: Scopus and ISI Web of Science, Institute of Electrical and Electronics Engineers (IEEE Xplore), Association for Computing Machinery (ACM) Digital Library, Multidisciplinary Digital Publishing Institute (MDPI), ScienceDirect, and Springer. Technical terms, including wireless sensor network (WSN), Internet of Things (IoT), wireless fidelity (Wi-Fi), ultrawideband (UWB), radio frequency identification device (RFID), Bluetooth Low Energy (BLE), indoor tracking, positioning localization, monitoring, and crowd control were utilized as search strings. Tab. 1 provides a summary of the research articles reviewed.

Table 1: A summary of research articles reviewed in the literature

Ref.	Device/Sensor	Application domain	Communication technology
[11]	Camera	Vehicle tracking	Industrial, scientific & medical (ISM)–band
[12]	Healthcare devices	Monitoring of patient vital signs	Peer-to-peer network
[13]	Tag	Elderly people monitoring	UWB and BLE
[14]	Wearable camera + headset	Museum audio guide	Vision WSN
[15]	Wearable device	Indoor localization and motion analysis	UWB
[16]	COTS RFID Devices (Tags)	Indoor localization system	RFID
[17]	User device	Indoor location-based control system	BLE
[18]	BLE sensors	Indoor localization and tracking system	BLE
[19]	Smart watch	Indoor positioning system	Wi-Fi fingerprinting
[20]	Smartphones	Traffic monitoring	Bluetooth beacons
[21]	ECG sensor and smartphones	Monitoring system	BLE
[22]	Active RFID tags	Indoor locating system	RFID
[23]	Wearable tag	3-D indoor localization	RFID
[24]	Passive tag	Indoor localization	RFID
[5]	UHF RFID tags	Indoor tracking	RFID
[25]	Smartphones	Indoor localization	Beacons BLE + Wi-Fi
[26]	Smartphones	Indoor positioning at smart museum	Beacons BLE
[27]	Bracelets	Crowd monitoring and control	Beacons BLE

2.1 Application Domains

Indoor tracking, positioning, localization, and monitoring systems can be described as a collection of networked devices used for locating or tracking objects. This includes people inside buildings, train stations, airports, underground garages, and other locations sharing aspects of systems to detect violation of distancing measures. These systems are related to this study's design aim.

Different types of indoor localization applications exist that utilize various sensor and communication technologies. For example, according to Jamil et al. [12], in the domain of e-health a localization system for monitoring elderly people was introduced. Two kinds of communication technologies (UWB and BLE) were used with two algorithms, hybrid localization and a TDOA-based algorithm (time difference of arrival), to enable the evaluation of elderly behaviors, monitoring of health status and wellbeing, and detection of emergency situations. Further, an economical design of wearable devices for indoor localization and motion analysis using data fusion algorithms is presented in zhang et al. [15] as another example. An interesting integration between an inertial measurement unit and UWB localization was introduced using a Mahony filter and quaternions as a solution that achieved high-accuracy, signal stability, and motion tracking capability.

Technologies and techniques related to outdoor monitoring and tracking approaches were omitted from the review conducted for this study. These approaches are considered more complex than required for the current study's aim and normally involve more than one communication interface attached to the utilized IoT devices to cover both short- and long-range communication [11,18,20]. At this stage of development, introducing a detection system of distancing measures with wide coverage capability for highly movable objects is not part of our aim.

2.2 Sensors and Communication Technologies in the Related Approaches

During the review, we found four main alternative technologies were used when designing wearable devices for indoor localization systems, namely, RFID, BLE and UWB, smart devices [19–21,24–26], and visual detection cameras. These alternatives strongly relate to the kinds of communication technologies used in the systems. As the basis of these technologies, a WSN was found a suitable communication technology adopted in the systems reviewed in this work. According to Nikodem et al. [11], for instance, detection cameras for monitoring road traffic and detecting and classifying vehicles were utilized and connected to a WSN. These cameras were installed on the sides of roads to track and monitor vehicles. In another study, wearable devices equipped with cameras and headsets and connected via a low-power Wi-Fi module could recognize paintings in a museum from the image captured by the camera using computer vision techniques and transfer the captured data to audio [14].

In addition, accuracy and energy consumption are critical factors that must be considered when designing wearable IoT devices and selecting their intercommunication technology. BLE is used to address energy consumption issues in many IoT applications [13,17,18,22]. However, UWB technology provides a high accuracy WSN for IoT systems [14,15]. In Kolakowski et al. [13], for instance, a balance between these two factors was discussed for the proposed system. UWB- and BLE-enabled devices were utilized in the proposed radio positioning system. UWB consumed higher energy during packet transmission than did BLE, although BLE provided more accurate detection results. A hybrid algorithm that offers the benefits of both technologies was designed and used for object detection and tracking [13].

In addition, a real-time intelligent crowd engineering analytical approach using mobility characterization and energy-efficient BLE to enhance mobility characterization and signal analysis (ICE-MoCha) was proposed in Jabbari et al. [27]. Crowd individuals wear the BLE tracker bracelets to manage the power consumption on the bracelet. Each tracker bracelet has a unique identification code for identifying each pedestrian in the crowd. The communication system of ICE-MoCha supports various messaging

techniques for sending and receiving messages or probes from or to the tracker bracelets based on type: active or passive. Generally, the approach works to translate the BLE beacon signals from various transmitters into group semantics for tracking the status of the crowd and predicting potential accidents using a smart video surveillance technology.

Further, RFID-based wearable devices were utilized in several IoT indoor localization systems [5,16,22,23]. RFID technology can transmit wearable device IDs and data captured by the attached sensor to an RFID reader over a range of about 100–200 m with low power consumption because it works on radio frequency signals [25–28]. The RFID-based devices are normally attached by RFID tags. Each tag consists mainly of two major components: an electronic chip for storing data corresponding to the individual object attached to it and an electronic antenna for transmitting sensed data to the associated readers [29].

For instance, the research discussed in Zhang et al. [22] introduced an active RFID-based indoor locating system that uses a frequency-hopping technique to reduce the noise of the captured RSSI fingerprinting data. Two kinds of low-cost tags, with a tag–tag communication protocol, are utilized: virtual reference tags and object tags. The reference tags are distributed in the covered area to improve accuracy, whereas the object tags present RFID coordinates to RFID readers. Further, an RFID-based 3D indoor tracking system for eHealth was presented in Paolini et al. [23]. The study introduced a customized low-cost reader that is remotely controllable and supported by real-time 3D scanning features to detect the dynamic location of a patient within an indoor area. The algorithm developed retrieves the reader–tag distance and its height. The approach demonstrated high accuracy detection in the tags' height, with an error rate below 20 cm in all the simulation experiments conducted, in which the average error of 10 experiments performed was less than 18 cm for distances up to 4.50 m [23].

Various sensors and existing communication technologies for indoor tracking and positioning systems were reviewed and investigated based on four main factors: energy-saving capability, accuracy, costs, and coverage range. The following table (Tab. 2) summarizes each technology considered in the review and its features regarding these four factors.

Table 2: A comparison between selected communication technologies regarding four factors

Technology	Energy saving	Accuracy	Cost	Range
Radio frequency identification (RFID)	✓	✓	✓	Up to 1 m
Bluetooth Low Energy (BLE)	✓	Acceptable	✓	✓
Ultrawideband (UWB)	–	✓	–	✓

2.3 Related Graph-Based Approaches

Graph theory played a significant role in developing collective adaptive systems (CASs). CASs can be defined as complex systems with a massive number of heterogeneous entities that interact with each other without a central control to achieve an individual or shared goal. In [30], a model of pedestrian movement was introduced as a CAS using an expressive process calculus language (CARMA). The model is based on graph structure representing a network of pedestrian paths. Each individual entity, which represents a pedestrian, acts as a decision-making element when it produces probabilistic choices about the next movement in the future and how that affects other pedestrian movements. Pedestrians are expressed as graph nodes in the model, whereas the current paths of each pedestrian are expressed via graph edges and determined by the pedestrian's current and next position.

In Allam et al. [5], the design of a wearable wireless sensor is presented to ensure crowd safety in high density situations. The device communicates with other devices in its sensing range to send simple warning and safety messages or status notifications across devices. This approach is based on the theory of graph density, in which each device in a crowd counts the number of surrounding devices and then calculates the density around it. Consequently, the previously determined threshold is examined to decide whether the device is in a safe situation. If not, an appropriate warning message is propagated to other devices within range, and guidance and instructions through messages are provided to prevent overcrowding and dangerous situations. The technique is based on an existing agent-based crowd simulation model called Multi-Agent Simulator of Neighborhoods (MASON), which is written in Java and used in 2D environment simulation tools.

Additionally, graph theory applications have a long history in the domain of crowd analysis, monitoring and control. In Matthias et al. [29], for instance, a graph-based framework for detecting and tracking pedestrians, dense crowds, and events is proposed using hidden Markov models. The framework requires a short sequence of images for effective pedestrian tracking and location of possible danger areas in a crowd. All results are implemented and demonstrated using a simulation model.

Further, in Angela et al. [7], a graph-based approach obtains indoor dense locations from tracking data using the semantic location. The dense location queries, processing, and indexing are performed using a proposed technique called the Dense Location Time Index, which combines the information of locations with time points. In another approach [8], a decision graph finds the cluster centers in the proposed pre-screening method. This method is presented to the density-based clustering algorithm to obtain possible shapes of the clusters and the internal density between each point in the graph [8].

Moreover, Xu et al. [9] introduces an approach for analyzing and recognizing crowd dynamics based on a static proximity graph. The proximity graph data structure represents crowd textures as graph vertices. Spatial closeness relationships between persons in the crowd are recognized in terms of their texture over time. The graph is supported by annotation extension attached to the vertices to add further semantics representing crowd context. Thereby, it provides a precise picture for the decision maker. Wearable devices on graph individuals collect information and are connected via a Bluetooth-based wireless communication technology and utilized as a network infrastructure.

3 The Mathematical Graph Model

Graphs are an appropriate data structure for managing and processing connected data. They enable tracing of interconnections between data entities to understand dependencies between them. In regard to the context of this work, tracing links among persons who spend time together as a group in public places, representing this in a graph data structure, and identifying possible clusters might provide deeper insights into the spread of the COVID-19 virus in the community.

Unlike the construction of direct simulations and predictions or video tracking systems, which are based on substantial processing of images and videos, and complex AI techniques for detecting abnormal events in crowds, the proposed graph-based mathematical model is defined here as a formal basis for all utilized concepts that are stored in the backend database. The proposed approach only requires information such as locations, distances, and other attributes associated with a group of people that are modelled using graph theory.

3.1 Foundation of the Graph Model

In previous research, a substantial number of real-world problems are represented mathematically and formalized using either directed or undirected graph theory. The cyber security knowledge (directed) graph and route planning (directed) graph presented in Jia et al. [31] and [32], respectively, are examples of the

variety of applications of graph theory. In this study, it is assumed that the direction of each individual is not vital information in the model. Thus, the (undirected) complete weighted graph model is considered a theoretical base of this work.

In this section, the notation of first-order predicate logic (FOPL), with extensions for equality (=), and membership ($\in$), is employed to express the mathematical foundation of the graph concepts used in the proposed system design. A syntactic sugar notation is used for a unary predicate to assign a type to a variable, for example, $Graph(g)$ is re-expressed as $g : Graph$. Further, the notation (,) is utilized to shorten and replace the logical *AND* ($\wedge$) operator.

In the proposed FOPL formalization system, there are four basic types of variables, namely *Graph*, *Vertex*, *Edge*, *Attribute*, and *Value*. Each variable is utilized to represent concepts of graph theory in the system. The following formal definition describes the notation of the computational graph model adopted in this work:

$$A\ Graph\ is\ a\ tuple\ G = (V,\ E,\ P,\ w, t) \tag{1}$$

V:A finite set of all vertices in graph G. Vertices correspond to individuals within a specific place who form a crowd. A vertex is expressed using the following formula:

$$\forall\ v, u : Vertex \cdot (v, u \in V) \tag{2}$$

E:A finite set of all edges in graph *G*. Edges are undirected and correspond to the actual distances between the graph vertices. This is expressed in the formula:

$$\forall\ e_1, e_2 : Edge \cdot (e_1, e_2 \in E) \tag{3}$$

The notation e_{vu} is used to represent an edge between two vertices v and u, where v and u are called the endpoints of e_{vu}.

P_v: A set of all attributes that belong to the value of a vertex v expressed as a pair of keys. Similarly, some edges may also have attributes in the model. The notation $P_{e_{vu}}$ is used to represent the set of all attributes of the edge e_{vu}. The pair of keys' values is described by the following formula:

$$\forall i,j \in \{1, 2, \ldots, n\}, i \neq j$$

$$\forall v : Vertex, \forall\ key_i, key_j : Attribute,\ val_i, val_j : Value \tag{4}$$

$$\exists P_v \cdot (P_v = \{\{key_i, val_i\},\ \{key_j, val_j\}\})$$

w : When modelling a distance between individuals in the crowd, it is important to determine a weight function w, where $\mathbb{R}$ is a real number associated with each edge e in the graph. This is represented via the following expression:

$$w : E \rightarrow \mathbb{R}$$

$$\forall\ v, u : Vertex,\ e_{vu} : Edge, g : Graph \cdot (v, u, e_{vu} \in g) \wedge\ (e_{vu} \in E) \tag{5}$$

$$\exists (x \in \mathbb{R}\ ) \cdot w(e_{vu}) = x$$

The weight is computed using the traditional distance formula as:

$$DISTANCE = \sqrt{u^2 - v^2} \tag{6}$$

where v and u are two points that have x and y coordinates $v(x_1, y_1)$ *and* $u(x_2, y_2)$.

Unlike other crowd modelling approaches that consider including all edges in the model, only edges between individuals that reflect the social distancing rules for reducing the transmission rate of

COVID-19 are included in the current study. Thus, the range of distance considered in the approach falls between 0 m and 4 m. Thus, it is unnecessary to represent all of the connections between individuals in the crowd that have distances (weights) over 4 m. Function t is used to indicate that two individuals (vertices), v and u in the model, fall within the range of interest for evaluation of COVID-19 physical distancing measures. Therefore, edge e links the pair v and u. This is expressed using the formula:

$$t : E \rightarrow (N \times N)$$

$$\forall\, g : Graph,\ (v,u) : Vertex,\ e : Edge \cdot (v,u,e \in g) \wedge t(e,v,u) \tag{7}$$

The convention e_{vu} is adopted to represent each edge that connects the pair of vertices (v,u) and appears in E for simplicity, where v and u are said to be the end vertices of an edge e_{vu}. The following formula describes this concept of the model:

$$\forall\, v,u : Vertex,$$

$$\exists e_{vu} : Edge \cdot t(e_{vu},v,u) \wedge (e_{vu} \in E) \tag{8}$$

3.2 The Creation of the Graph Model

The graph model of the crowd in a determined place is created by locating each individual as a graph point (vertex) within a specific area at an exact time using appropriate technology, for example, BLE. Each graph vertex, in the real-world set, has attributes to store the intersection of latitude and longitude lines, which represent the coordinates of that vertex. The coordinates are used to identify the exact location of the vertex in the place.

The proposed model focuses on modelling the distance between crowd individuals to observe the violation of social distancing rules. The distance (in meters) between each pair of vertices is calculated to be the weight of the undirected edge between that pair. Times and dates are other possible pieces of information (attributes) that might be assigned to each edge.

The type of graph model based on a related technology for capturing vertex locations is a complete undirected weighted graph. Graph G is considered a complete graph if there is a unique edge connecting every pair of its distinct vertices. If G has n vertices the graph is called K_n and the number of its edges is $n(n-1)/2$.

Mathematically, graphs can be represented in several ways, such as via adjacency lists and adjacency matrices. The adjacency matrix is considered one of the graph representation alternatives adopted in this work. To create a corresponding adjacency matrix A of the complete graph K_n, suppose the total number of vertices is n; then, the definition of the adjacency matrix A can be expressed with the following formula:

$$\forall i,j \in \{1,2,\ldots,n\}, i \neq j$$

$$\forall\, v,u : Vertex, e_{vu} : Edge \cdot ((v,u,e_{vu} \in K_n) \wedge (w(e_{vu}) = x)$$

$$\rightarrow \tag{10}$$

$$[A]_{i,j} = w(e_{vu})$$

4 Conceptual Design and Implementation Choice

This section presents a conceptual view of the system's major components. Our vision of the overall system architecture falls within the scope of IoT edge-based architecture. The traditional IoT edge-based architecture consists of four layers: A sensing layer, an edges layer, a communication layer, and a cloud layer [33].

The high-level architecture of the core system components, shown in Fig. 1, consists of three main layers, namely, the sensing layer, middleware layer, and data storage layer. Fig. 1 demonstrates the communication between these three layers in terms of their inputs and outputs. The following subsections briefly describe each layer, focusing mainly on the detail of the data storage layer.

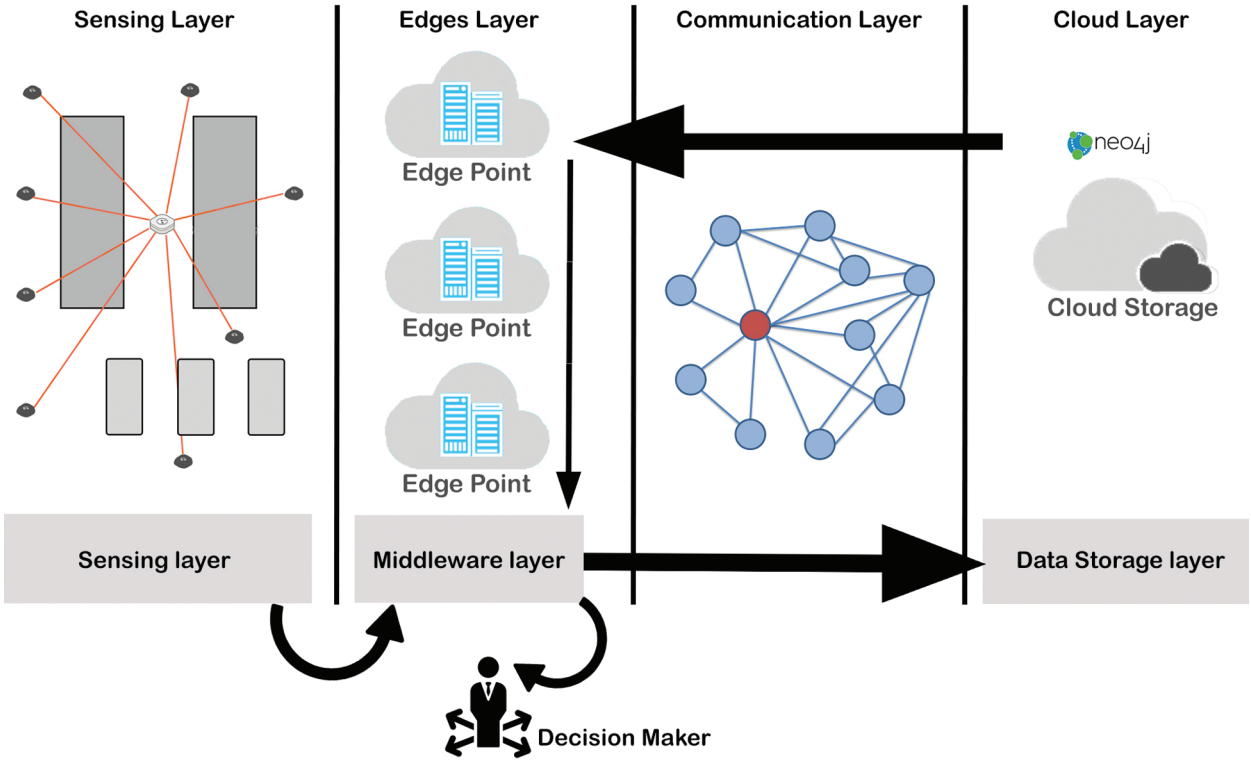

Figure 1: High-level architecture of the proposed system design

4.1 Implementation of the Graph Model

Managing a massive amount of stored data (big data) requires a type of data model that can support its characteristics, namely, volume, velocity, variety, and veracity. These characteristics are not fully supported in the traditional relational data models. However, owing to the dynamic nature of the NoSQL data model, it is suitable to deal with critical issues of big data that are related to its characteristics, such as efficiency, scalability, and availability [33,34]. A graph database, as a form of the data model, has demonstrated the ability to express and manage connected sensor-based data in various IoT systems [35,36], and is an example of a dynamic big data schema.

This study introduces a Neo4j graph database implementation of the complete weighted graph data model, discussed previously in Section 3, to model the violation of distancing regulations within a particular place. Neo4j is one of the most popular Java-based open-source native graph database systems and is used to implement the property graph data model efficiently. In the Noe4j graph model, data are structured as nodes, relationships, and properties attached to the nodes or relationships. It provides robust transactional storage of big data and supports full ACID features (atomicity, consistency, isolation, durability) to maintain the integrity of stored data. Neo4j also supports deep graph analysis and querying using a query language named “Cypher” [37]. It is a reliable graph database system for large and rich

graph modelling of crowds. Consequently, Neo4j is considered an appropriate implementation platform for the proposed weighted graph model presented in this paper.

Noe4j's declarative query language, Cypher, is similar to SQL but specialized for graphs. The Cypher language is considered a simple and easy to learn language that allows users to store, retrieve, and manipulate data stored in a graph database. The Cypher syntax is declarative and is expressed in a logical way for matching patterns of nodes and relationships in graphs. For example, the keywords "MATCH/ RETURN" work similarly to the "SELECT" statement in SQL to project nodes or relationships that satisfy given constraints. Additionally, the keywords "CREATE/ MERGE" are used to create new nodes in the graph, similarly to the "INSERT" statement in SQL. The following figure (Fig. 2) provides an example of a complete directed weighted graph consisting of 11 nodes and 54 relationships.

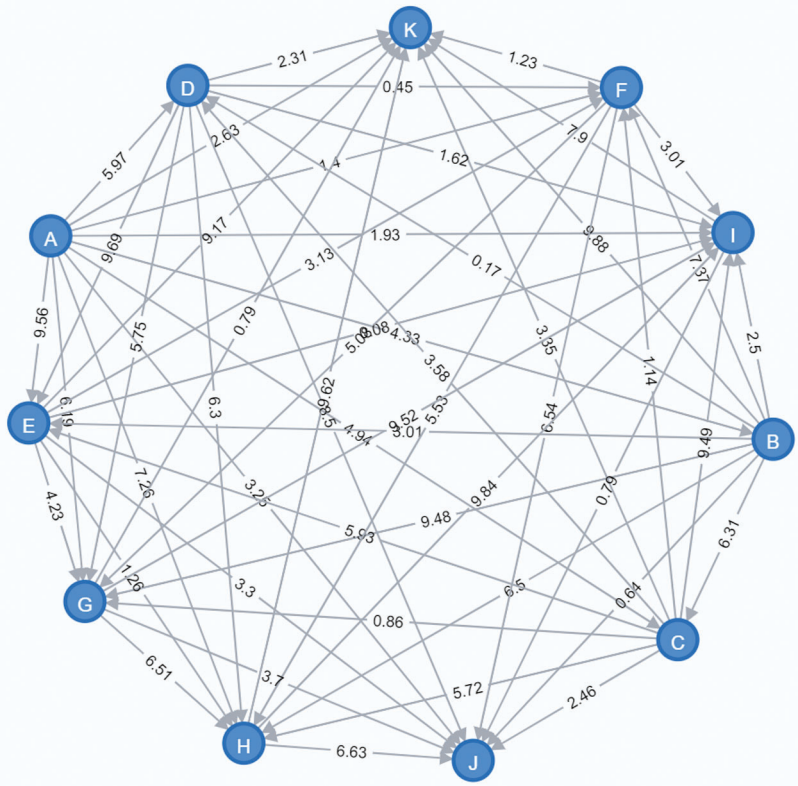

Figure 2: A complete directed weighted graph used to represent a group of individuals and the physical distances between them

Because the output of the proposed transformational system in the middleware layer, discussed in the following subsections, is an executable Neo4j Cypher query script file, the Noe4j database engine can run Cypher script files to construct, or update, the graph representation of the real-world group in the system. Then, several predefined queries that support decision makers might be executed on the stored graph to retrieve insight results. The following snapshot of the Cypher script (Listing 1) illustrates the executable code used to generate three graph nodes, A, B, and C, using CREATE statements, in addition to three undirected and weighted relationships with a randomly computed distance weight between them, using MERGE statements.

Listing 1: A snapshot of the Neo4j Cypher script used to create three nodes including their interrelations

```
CREATE(v0:person { vId: 'A' })
CREATE(v1:person { vId: 'B' })
CREATE(v2:person { vId: 'C' })
MERGE (v0)-[e0:DISTANCE {weight: apoc.number.format(rand()*10, '#.##;(#.##)')}]-(v1)
MERGE (v0)-[e1:DISTANCE {weight: apoc.number.format(rand()*10, '#.##;(#.##)')}]-(v2)
MERGE (v1)-[e11:DISTANCE {weight: apoc.number.format(rand()*10, '#.##;(#.##)')}]-(v2)
```

Nodes of the Neo4j crowd graph represent people (individuals) in a crowd, whereas undirected and weighted edges represent the actual distances between individuals at a corresponding time point. Thus, the graph created at a specific time interval represents the status of the social distancing rule application at that given point in time. In the real world, the person node holds properties to represent the location of an individual who wears a sensor or is attached to a communication device.

Each person node in the crowd graph is located near another person node within a particular distance. The distance between them is represented in the graph by a relationship between two person nodes. Each distance relationship has a calculated weight property to represent the measure between each pair of individuals. The weight of each distance is computed using Formula 7, as mentioned previously in Section 3.1 and Section 3.2.

4.2 Technology and Connectivity of the Sensing Layer

This layer consists of wearable devices connected to a wireless network. In this layer, a WSN (e.g., Wi-Fi and Bluetooth, Zonal Intercommunication Global-standard [ZigBee], and fifth generation cellular wireless [5G]) and a suitable tag-based detection technology were combined to collect required information about physical distancing between individuals during a corresponding time interval. Thus, the physical distancing was tracked using wearable devices or tracker badges and transmitted to the nearest edge point for analysis and processing.

As a design choice, Bluetooth smart badges were considered IoT wearable devices. These cost-effective badges are based on BLE technology, which facilitates short-range communication and offers an adequate level of indoor tracking accuracy. Additionally, beacon devices were used to expand the Bluetooth coverage range of the closed area because the standard beacons have an approximate range of 70 m. Thus, based on the dimensions of the closed area, the number of required beacons could be determined. Fig. 3 shows the Bluetooth network infrastructure using multiple BLE beacons and cloud beacons that were connected to the proposed IoT edge-based cloud system.

Because one of the objectives of the proposed design was to introduce a social distancing tracking solution, cloud beacons, which have the capability of data backup, configuration, and application of updated settings to all other connected BLE beacons, were employed to monitor other BLE beacons and send their collected data over the Internet to the cloud backend data storage. They were also used to increase the collection of distancing data for more than one closed area. Owing to the Wi-Fi connectivity feature available in cloud beacons, they can upload data to the edge server and then to the centralized cloud storage over Wi-Fi. It is worth mentioning that the range of Wi-Fi networks can be up to 200 m greater than the standard Bluetooth range.

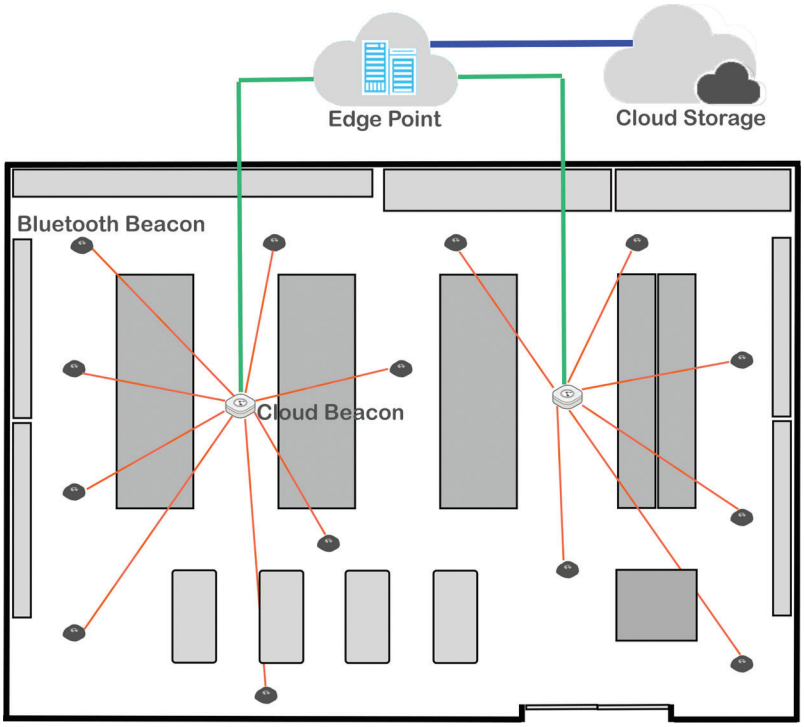

Figure 3: Bluetooth-based (BLE) network infrastructure

As a result, each person approaching the area covered by the proposed physical distancing monitoring system is asked to wear the Bluetooth smart badge assigned by a previously configured unique ID to sense and report distance changes within the group to the decision maker via the wireless network infrastructure. The assigned ID of each badge represents a specific location and direction (if required) of an individual within the group inside the closed area. The approximate distances within the group of individuals and a timestamp are detected and transferred via the Wi-Fi connection to the edge point and then stored in the backend data storage.

This approach also considers information security issues and individual privacy since the distributed Bluetooth badges and the cloud beacons only monitor and record anonymous data at each point in the area, excluding any information that can identify an individual. It should be noted that detailed specification regarding algorithms and techniques used for detecting distances between individuals is outside the scope of this paper.

4.3 Defined Structure and Queries of the Data Storage Layer

The collected and processed information in the middleware layer is stored in the system using a centralized cloud database that supports big data storage, future processing, and retrieval. The stored data might be used for real-time or post processing, archiving, querying, analysis, or prediction.

The graph-oriented database system was selected as a backend data model to gain the benefits of the graph data structure. This is because of the nature of the sensed data stored in the database and the capability of graph databases to perform efficient query processing and deep graph analytics. Importantly, the data storage is physically located in the cloud storage as a centralized cloud database.

To enable the system to calculate the degree to which the existing group is violating the social distancing regulation, the number of edges, with distance values < = 2 m, is counted and compared with the total number of edges between group individuals (with distance > 2 m and < = 4 m). Additionally, the number of individuals involved in the graph is considered to examine the size of the graph. In the proposed proof of concept, this can be achieved using several simple Neo4j Cypher query statements:

Q1: What is the number of attached nodes in the graph?

```
MATCH (n:person)-[r:DISTANCE]->(m:person) WHERE toFloat(r.weight) < 4.0
RETURN COUNT(n)
```

Q2: What is the number of the considered edges in the graph?

```
MATCH (n:person)-[r:DISTANCE]->(m:person) WHERE toFloat(r.weight) < 4.0
RETURN COUNT(r)
```

Q3: What is the number of edges with no violation of distancing measure in the graph?

```
MATCH (n:person)-[r:DISTANCE]->(m:person) WHERE toFloat(r.weight) >= 2
AND toFloat(r.weight) < 4.0
RETURN COUNT(r)
```

Q4: What is the number of edges with mild violation of distancing measure in the graph?

```
MATCH (n:person)-[r:DISTANCE]->(m:person) WHERE toFloat(r.weight) > 1.40
AND toFloat(r.weight) < 2.0
RETURN COUNT(r)
```

Q5: What is the number of edges with high violation of distancing measure in the graph?

```
MATCH (n:person)-[r:DISTANCE]->(m:person) WHERE toFloat(r.weight) > 0.01
AND toFloat(r.weight) <= 1.39
RETURN COUNT(r)
```

Based on the results returned from these queries, simple calculations are required to allow the decision support system to generate an appropriate message to send to the decision maker:

1. The percentage of individuals who are involved in the group:
   `% involved = (total nodes - no of nodes out of the group) / total nodes)*100`
2. The percentage of individuals who violated the social distancing measures:
   `% violation = (no of edges with distance less than 2 meters / total nodes)*100`

5 Overview of Transformation Approach

A major contribution of this study is the design of a model transformation layer, which is considered a middleware layer of the proposed architecture. Three main operations are performed in this layer, namely, constructing a source model that conforms to a defined source metamodel, translating the source model into a target model that conforms to a defined target metamodel, and executing the target model and generating insight reports for decision makers. It is worth mentioning that the mapping rules were implemented using Atlas Transformation Language (ATL). Further details about ATL are presented in Subsection 5.2. The following figure (Fig. 4) illustrates the model transformation approach adopted in the proposed system design.

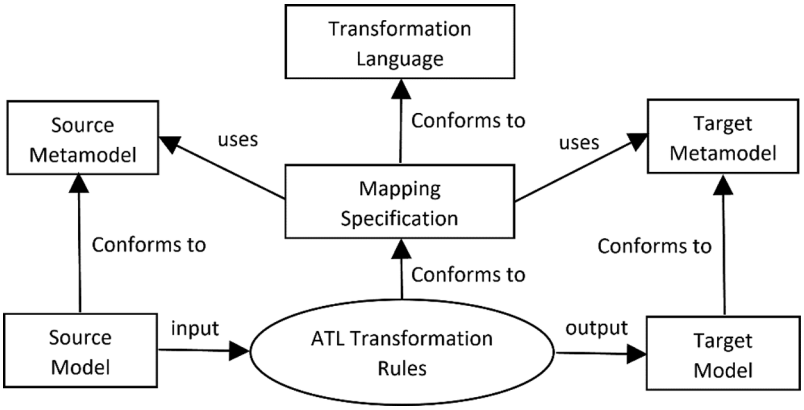

Figure 4: The ATL approach of model transformation

5.1 Constructing the Source Model

Two main steps are considered for creating the source model. The first step is to construct a complete graph structure on the fly by gathering all the sensed data, captured at a specific time from IoT sensors, and manipulated as a density matrix in memory based on edges. The second step is to create an eXtensible Markup Language (XML) memory model of the captured complete graph. During the process of building the XML memory model, all nodes that are outside the range of distancing violations are eliminated. This XML memory model implementation [37] represents the simplified matrix and is considered the source of the translation operation in the form of a graph structure.

A common method to represent and parse XML memory models is to use a Document Object Model (DOM) or Java Document Object Model (JDOM) package. Moreover, options other than XML might be used for data formatting and serialization standards, namely JavaScript Object Notation (JSON) and YAML Ain't Markup Language (YAML). XML has the advantage of describing the structure of data using a human-readable and friendly format and it has a flat learning curve. In addition, the XML format relies on a standard means of representing models using the XML Metadata Interchange (XMI) format in the model transformation strategy adopted in this work, as explained in the following subsection. The following algorithm (Listing 2) describes the rules of translating the complete in-memory graph model into the simplified intermediate source graph model of the core transformation step.

This output source model is considered an intermediate textual domain specific language (DSL) for describing the structure of a graph using XML and conforming to a defined metamodel of a source model (Fig. 5). Listing 3 provides a snapshot of the graph model produced. It is important to note that constructing an intermediate representation, using a language similar to the structure of graph databases, is considered a critical step in the design because it leads to simplifying the process of generating the executable Neo4j Cypher in the transformation chain.

5.2 The Core Transformation Strategy

To present transformation rules applied to the final executable Neo4j Cypher queries for the intermediate XML graph model, the ATL language was used to express each matching rule of this transformation component in the middleware layer [38]. ATL is a hybrid model transformation language that contains a mix of declarative and imperative constructs to describe rules of transformation in a simple manner. One of the main reasons for adopting the hybrid approach of model transformation is to simplify mapping rules to be easier to read and understand, which is crucial in larger transformation modules from a software engineering perspective.

Listing 2: Algorithm of building the intermediate graph model for the complete graph density matrix

```
STEP 1:  Use A[v][u]
         Define enum[A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q]
STEP 2:  Initialize a new XML Graph element
STEP 3:  for each row i in the A
           Create a new XML Vertex element as a child of the Graph element
           Set a Vertex attribute id=enum[i]
         End for
STEP 4:   for each row i in the A
            for each column j in A
              if (A[i][j] > 0) AND (A[i][j] ≤ 4)
                Create a new XML Edge element as a child of Graph element
                Set an Edge attribute id=”auto" and weight= A[i][j]
                Create a new XML Endpoint element as a child of Edge element
                Set an Endpoint attributes ref=vertex(i).id
                Set an Endpoint attributes ref=vertex(j).id
STEP 5:  for each row node r in Graph element g
           for each cell node c in r
           if the values of all cell elements in r are zeros
           then
             Eliminate r
```

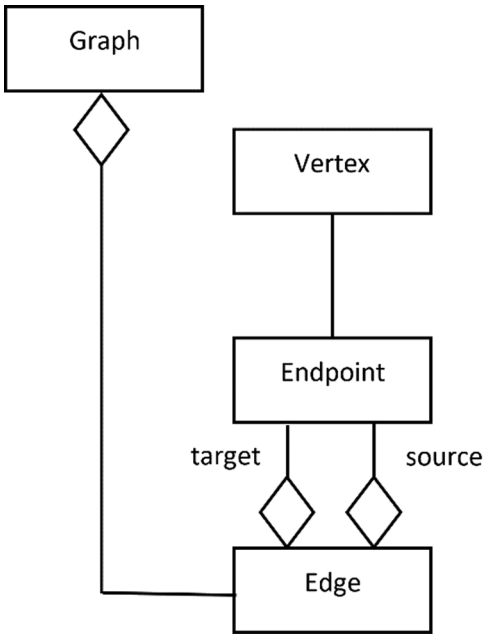

Figure 5: The source graph metamodel

Listing 3: A snapshot of the generated XML graph model

```
<Graph>
  <Vertex id="v0" label="A" />
  <Vertex id="v1" label="B" />
  <Vertex id="v2" label="C" />
  <Edge id="e0" weight="0.36">
    <Endpoint ref="v0" />
    <Endpoint ref="v1" />
  </Edge>
  <Edge id="e1" weight="2.92">
    <Endpoint ref="v0" />
    <Endpoint ref="v2" />
  </Edge>
  <Edge id="e2" weight="1.07" label="DISTANCE">
    <Endpoint ref="v1" />
    <Endpoint ref="v2" />
  </Edge>
</Graph>
```

The transformational approach of ATL is based on pattern matching, in which a pattern of a source model, expressed using XMI standards, is evaluated and transformed using a set of declarative ATL matching rules into a pattern of a target model, also expressed using XMI format. The approach maintains traceability links between elements in a source model, rules, and the created elements in the target model. The imperative side of ATL can be seen when the developer needs to encode or customize control flows explicitly for complex parts of transformational algorithms. These imperative definitions of mapping rules are used to support the declarative mapping rules in the transformation module. This is implemented in ATL using helper functions, as shown in Listing 4, which demonstrates the definition of two ATL declarative matching rules, Vertex2CreateNodeStmt and Edge2MergeStmt, used to transform the intermediate graph XML model into an executable Neo4j Cypher script. In each rule, one-to-one mapping statements are declared. For instance, each vertex v is mapped into a generated CREATE Cypher script statement for creating a new Neo4j graph node with the same properties of v. Additionally, to specify the concrete syntax of the generated script, helper functions are utilized to express the detailed syntax of the target script. A detailed description of the syntax and semantics of the ATL language are outside the scope of this work but can be found in Jouault et al. [38].

Listing 4: ATL matching rules for mapping the intermediate model into an executable Cypher script

```
MATCH (n:person)-[r:DISTANCE]->(m:person)
WHERE toFloat(r.weight) >= 4.0
RETURN n.vId, r.weight, m.vId
```

6 Case Study

In this work, the case study presented was based on a real-world example of actions taken by various global supermarket and grocery store chains around the world, such as Tesco, Waitrose, and Dunnes, to

slow the transmission of the COVID-19 virus. As an example of common implemented physical distancing measures, the governments of the United Kingdom and the Republic of Ireland advise maintaining a gap of at least 2 m (6 feet) from others in public places, specifically those who are coughing, sneezing or have a fever [16–19]. In more detail, these rules restrict the number of customers allowed in medium-sized stores at a given time to between 10 and 25 persons, which represents 40% of the stores' normal capacity. A supermarket consists of one main entrance, several aisles, shelves, fridges, and four cashier desks.

Consequently, initial criteria were defined for categorizing distance properties in the weighted graph model. It can be concluded that an individual is considered out of a group when he or she is located at a distance of more than 4 m from all other individuals in the same group. The distancing rule for reducing the transmission of COVID-19 is based on maintaining a space between individuals of over 2.5 m as much as possible. Thus, to represent the group of people within the crowd that are closer than 2.5 m to each other, an undirected weighted subgraph G is extracted from the original complete graph K_n. The weight of each edge $w(e)$ represents the actual distance between each pair of individuals that violate the social distancing measure to a particular degree (high, low, or safe). Similarly, another adjacency matrix M is extracted from A in which $M \subseteq A$ implements the undirected weighted subgraph G. The following formula describes the definition of M:

$$\begin{aligned}
&\forall G : Graph \cdot O(G) = n,\\
&\forall x \in \mathbb{R} \cdot (0 < x) \wedge (x < 2.5)\\
&\forall e : Edge \cdot (e \in G) \wedge (w(e) = x),\\
&\forall i,j \in \{1,2,\ldots,n\} \wedge i \neq j \qquad (11)\\
&[M]_{i,j} \begin{cases} x, & 0 < x < 2.5 \\ 0, & \neg(e_{vu} \in E) \vee x \geq 2.5 \end{cases}
\end{aligned}$$

From the constructed matrix M, all individuals in the place who are violating the distancing rule for that place, at a specific time, are captured. The matrix M is considered a submatrix of A after eliminating all vertex entries that have a weight equal to zero or greater than 4.

6.1 The Simulation Strategy

As proof of concept, a discrete-time simulation model is adopted. It is assumed that the changes in distances between individuals occur only at each discrete time tick. Consequently, a simple algorithm for generating multiple weighted graphs with random distances between their nodes is implemented using Noe4j Cypher script. The Neo4j built-in random function *rand()* is used to generate random weights for all edges in each complete graph to represent distances between individuals rather than to detect real-world locations of individuals inside the store at a specific time. In the simulation, 21 weighted graphs are generated randomly. After inspecting the weights on their edges, all edges with distances of > 4 m are eliminated from the graph and their end-point nodes are considered unattached. Listing 4 illustrates the Cypher script used to return the updated graph after eliminating edges with weights > 4 m.

The simulation then answers the five queries, presented in Subsection 4.3.1, for each of the generated graphs (Tab. 3). The Neo4j 3.5.14 database system is used to run the script, thus producing 21 graphs with a total of 176 nodes, 1040 properties, and 864 relationships, completed after 103 ms. Fig. 6 provides a snapshot of the creation of the weighted graphs used in the simulation.

Table 3: Simulation results of the 21 generated graphs and their classifications

Graph no.	# Connected nodes	# Edges (distances < 4)	# No violation cases	# Mild violation cases	# High violation cases	Category
1	11	19	11	5	3	Mild
2	11	21	14	4	3	Mild
3	10	25	17	2	6	High
4	9	7	7	0	0	No violation
5	11	29	12	5	12	High
6	8	10	5	1	4	High
7	14	25	15	7	3	Mild
8	7	6	6	0	0	No violation
9	11	20	12	5	3	Mild
10	11	14	6	2	6	High
11	8	5	1	3	1	Mild
12	11	23	17	5	1	Mild
13	8	22	13	0	9	High
14	9	19	10	3	6	High
15	11	21	11	3	7	High
16	3	2	2	0	0	No violation
17	11	19	12	7	0	Mild
18	11	18	9	1	8	High
19	11	26	15	9	2	Mild
20	11	21	9	2	10	High
21	8	13	3	1	9	High

A second round of inspection is applied, and it is noted that the simulation results showed three different cases. These cases can be categorized based on the number of nodes falling in the distancing range as no violation, mild violation, and high violation cases. Generated graphs belonging to the no violation case represent the possible potential ideal situation, captured by the system at a specific time tick, when there is no detected violation of social distancing measures inside the store; that is, the majority of distances between individuals range between 2 m and 4 m.

Graphs belonging to the mild violation cases demonstrate a situation where some minor or moderate violations of the distancing measures in the store are detected. The group of persons has a small number of nodes falling in the range of 1.4 m and 2 m, whereas the remainder fall in the greater range. However, graphs belonging to the high violation cases show major violations of distancing measures captured by the system. A larger number of nodes violates the social distancing rules inside the store, with distancing ranges starting from 0 m to about 1.39 m.

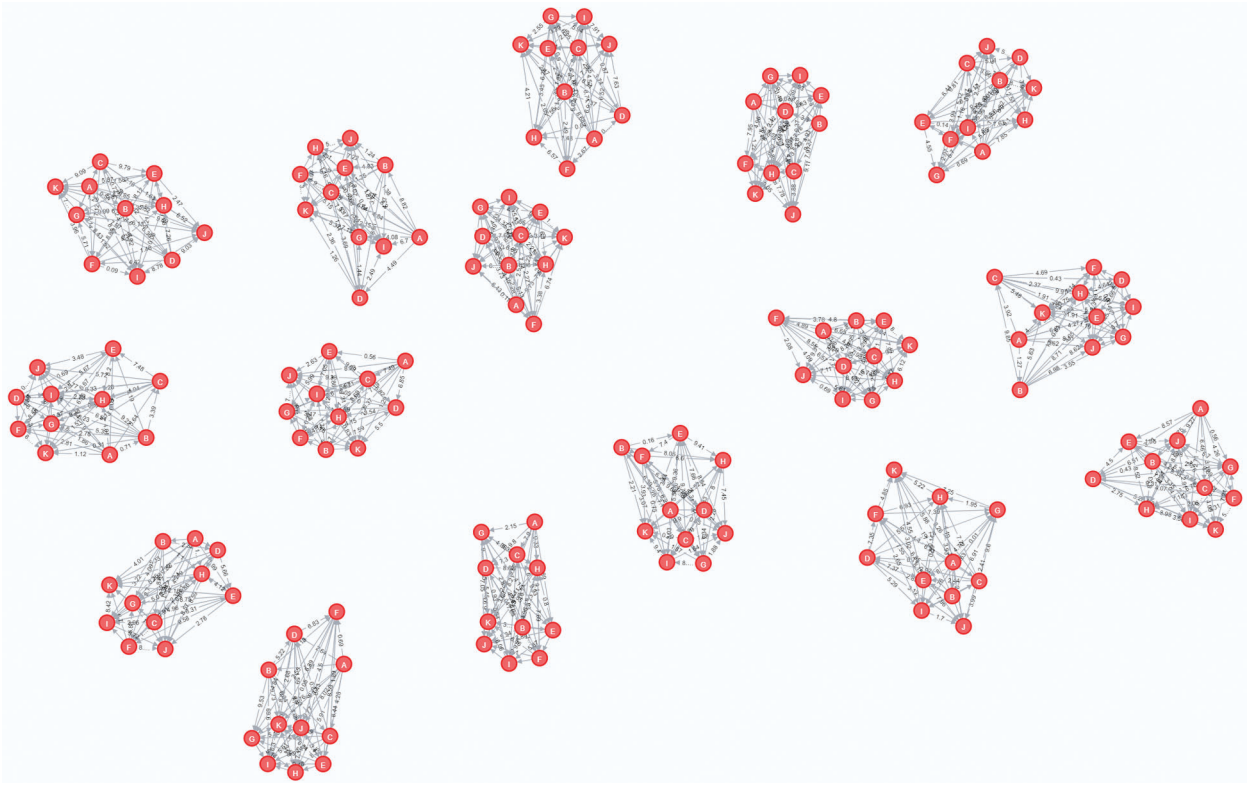

Figure 6: A snapshot of the generated graph networks used in the simulation

6.2 Technical Discussion

According to the simulation results demonstrated in Tab. 3, it is possible at this stage of system development to detect the distancing measure violations for a group of individuals using simple graph database queries, rather than machine learning and computer vision techniques. From the 21 randomly generated graphs, the system detects the class of violation and its degree, as demonstrated in the following chart. Fig. 7 summarizes the results showing the difference in violation of social distancing measures for each case. The simulation produces results of 10 high violation, 8 mild violation, and 3 no violation cases. For simplicity, a selected graph from each category, represented in Tab. 3, is considered in the following subsections to demonstrate in more detail the simulation results of Graphs 4, 11, and 21.

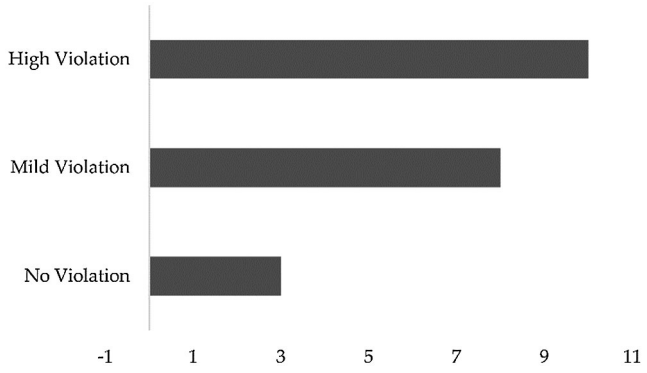

Figure 7: A summary of the experiment results

6.2.1 Detecting No Violation of Social Distancing Measures

The graph generated in case 4 is considered for demonstrating the detection of no violation of distancing measures. A subgraph representation of individuals inside the store and their distances that range between 2 m and 4 m only are considered.

This range of distances is considered a safe and acceptable measure for coronavirus and the need to maintain separation between the customers in the store as much as possible. For illustration only, the graph nodes are positioned manually in possible locations inside the store sketch, maintaining the distances generated between them, to present a possible visualization in this proof of concept. In Fig. 8, it is evident that the original completed graph has become two subgraphs. The first graph consists of four vertices and three edges, whereas the second graph contains five nodes and four relationships only.

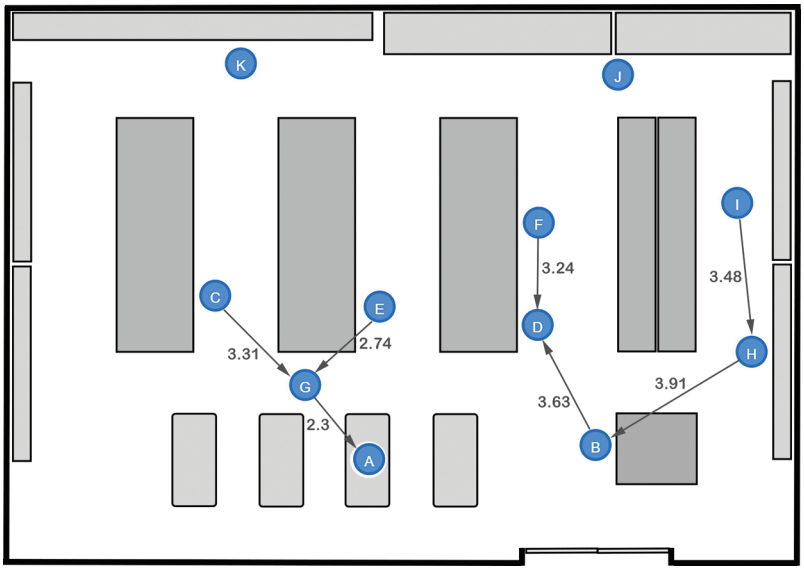

Figure 8: A case representing no violation of social distancing measures inside a supermarket

Based on the simulation results illustrated in Tab. 3, which provide the answers to the predefined queries Q1–Q5, the system should output the message shown in Tab. 4 to the decision maker.

Table 4: Results of generated Graph 4 and the output message to decision support

Q1	Q2	Q3	Q4	Q5	% Node Involved	% Violation	Message to User
9	7	7	0	0	82	0	There are no violations of physical distancing measures in the store.

6.2.2 Detecting a Moderate (Mild) Violation of Social Distancing Measures

Like the steps applied to the first case, the generated graph in case 11 is considered to demonstrate the detection of a mild violation of distancing measures. Subgraphs of persons in the store in which some nodes fall in the range of 1.4 m and 2.0 m are considered, while the remainder exceed that range.

The chosen distance range in this case is considered to represent a low degree of transmission of the COVID-19 virus in public. Like the first case, the graph vertices are located manually inside the store sketch, maintaining the distances generated between them, to provide a visualization for decision makers.

Fig. 9 shows three small subgraphs derived from the original graph. The first graph consists of two nodes and an edge, and each of the second and third graphs has three nodes and two relationships.

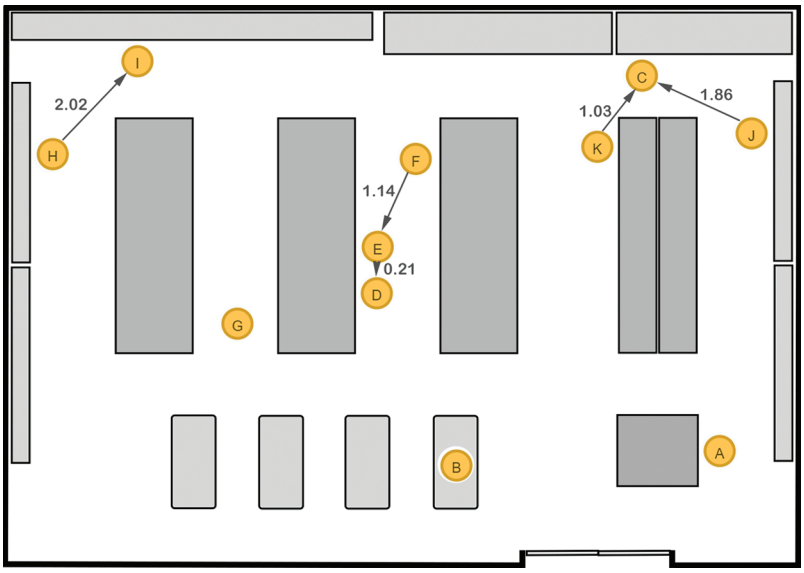

Figure 9: A case representing mild/moderate (limited) violation of social distancing measures inside a supermarket

Like the previous case, based on the simulation results and answers to the predefined queries Q1–Q5 in Tab. 3, the system should output the message shown in Tab. 5 to the decision maker.

Table 5: Results of generated Graph 11 and the output message to decision support

Q1	Q2	Q3	Q4	Q5	% Node involved	% Violation	Message to user
8	5	1	3	1	73	60	There are moderate violations in the store at the time of this message. Advice to customers might be required.

6.2.3 Detecting a High Violation of Social Distancing Measures

Like the steps applied in the previous cases, the generated graph in case 21 is considered for demonstrating the detection of a high violation of social distancing rules that leads to a high degree of transmission of the virus. Subgraphs for this situation with a distancing range between 0.01 m and 1.39 m are considered.

As for the previous cases, Fig. 10 demonstrates that there are two subgraphs that violate the distancing measures inside the virtual store. The first graph is considered a large graph and accounts for about 64% of the total number of the nodes in the original graph (7 nodes and 12 relationships). The second subgraph is a smaller graph with two vertices and one relationship. Like the past cases, based on the simulation results and the answers to the predefined queries Q1–Q5 in Tab. 3, the system should output the message shown in Tab. 6 to the decision maker.

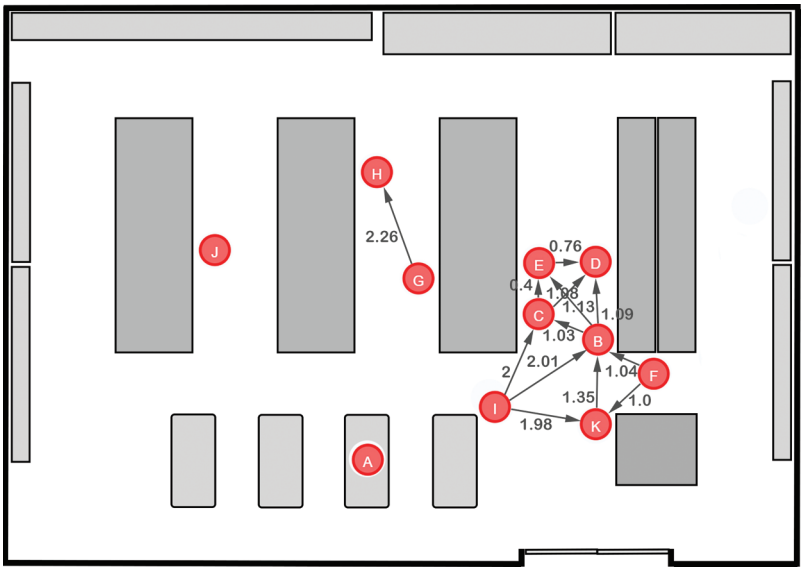

Figure 10: A case representing high violation of social distancing measures inside a supermarket

Table 6: Results of generated graph 21 and the output message to decision support

Q1	Q2	Q3	Q4	Q5	% Node involved	% Violation	Message to user
8	13	3	1	9	72	77	There are high violations in the store at the time of this message. Urgent action must be taken.

6.3 Scalability and Limitation of the Proposed Approach

Before discussing the scalability details of the proposed design, it should be recalled that the main objective of the system is to detect violations of COVID-19 physical distancing measures. To evaluate scalability, a larger test case to detect distancing violations of a crowd of people (121 persons) was designed. The Neo4j 3.5.14 database system was utilized to run the Cypher script, shown in Listing 5, to produce a larger graph with 121 nodes and 2020 relationships (edges) in 176 ms.

Tab. 7 shows the number of generated elements of the graph, including the number of created nodes and the number of created or matched edges. It also lists the time the system took to assess the graph structure and edges with weights > 4 m.

The concept of connected graphs was considered in orchestrating the generation of edges between nodes to make this case more realistic. The 121 nodes were distributed into five groups of nodes, each of which represented a subgraph. Then, edges were defined between nodes in every pair of neighboring graphs. This strategy is illustrated in Fig. 11.

Like the strategy of assessing graphs in the original cases discussed in Section 6.2, the predefined queries Q1–Q5 were considered to determine the level of distancing violation. For simplicity, Tab. 8 shows the triples of Group1 and Group2, including their connected nodes and edges. These triples are presented following the format: (NODE EDGE[weight] NODE).

Listing 5: Neo4j Cypher script used to generate a larger graph with 121 nodes

```
# Creating two groups and edges between them
UNWIND $group1 AS cycle1
UNWIND $group2 AS cycle2
MERGE (x1:per1 { vId: cycle1.vId })
MERGE (y1:per2 { vId: cycle2.vId })
MERGE (x1)-[e0:DISTANCE {weight: apoc.number.format(rand()*10, '#.##;(#.##)')}]->(y1)

# Eliminating edges that have weight > 4.
MATCH (x1)-[e0:DISTANCE]->(y1)
WHERE toFloat(e0.weight) >= 4.0 OR (toFloat(e0.weight) >= 0 AND toFloat(e0.weight) <= 0.1)
DELETE e0
RETURN x.vId, e0.weight, y.vId
```

Table 7: Total number of elements in the generated graph and the time of their creation

	#Nodes	#Edges	#Deleted edges	Time (ms)
Group1	41	420	267	29
Group2	20	400	241	33
Group3	20	400	256	28
Group4	20	400	263	40
Group5	20	400	259	46
Total	121	2020	1286	176

Based on the results shown in Tab. 7 and Tab. 8, no problems are apparent with the horizontal scaling of the proposed system design. As with any expansion of the graph size, the system is able to record all of the triples to be assessed using the predefined queries Q1Q5 and provide the percentage of violations to output an appropriate message to the decision maker, as shown in Tab. 9.

Overall, the proposed approach employed various utilizations of IoT devices, edge/cloud computing, and BLE beacons for indoor localization, detection, and monitoring. Consequently, the main objective of the system design presented is achievable. An advantage of the proposed design is the ability to use the recorded data in the closed area to analyze and understand group behavior inside that area using only simple graph database query processing. For deeper analysis, the Neo4j Graph Data Science library introduced by Neo4j enables data scientists to make valuable predictions.

Another advantage is that the proposed design can be considered vertically scalable by increasing the number of IoT devices, including BLE beacons and Bluetooth badges, to accommodate larger crowds and coverage areas. A potential extension of the proposed system architecture is the adoption of smartphones or smart watches as alternatives to Bluetooth badges because all smart devices are currently equipped

with Bluetooth connectivity. This would enable the design presented to be expanded to include, for instance, a mobile application installed in smart devices to notify individuals about the status of their current location, which could also apply to outdoor areas.

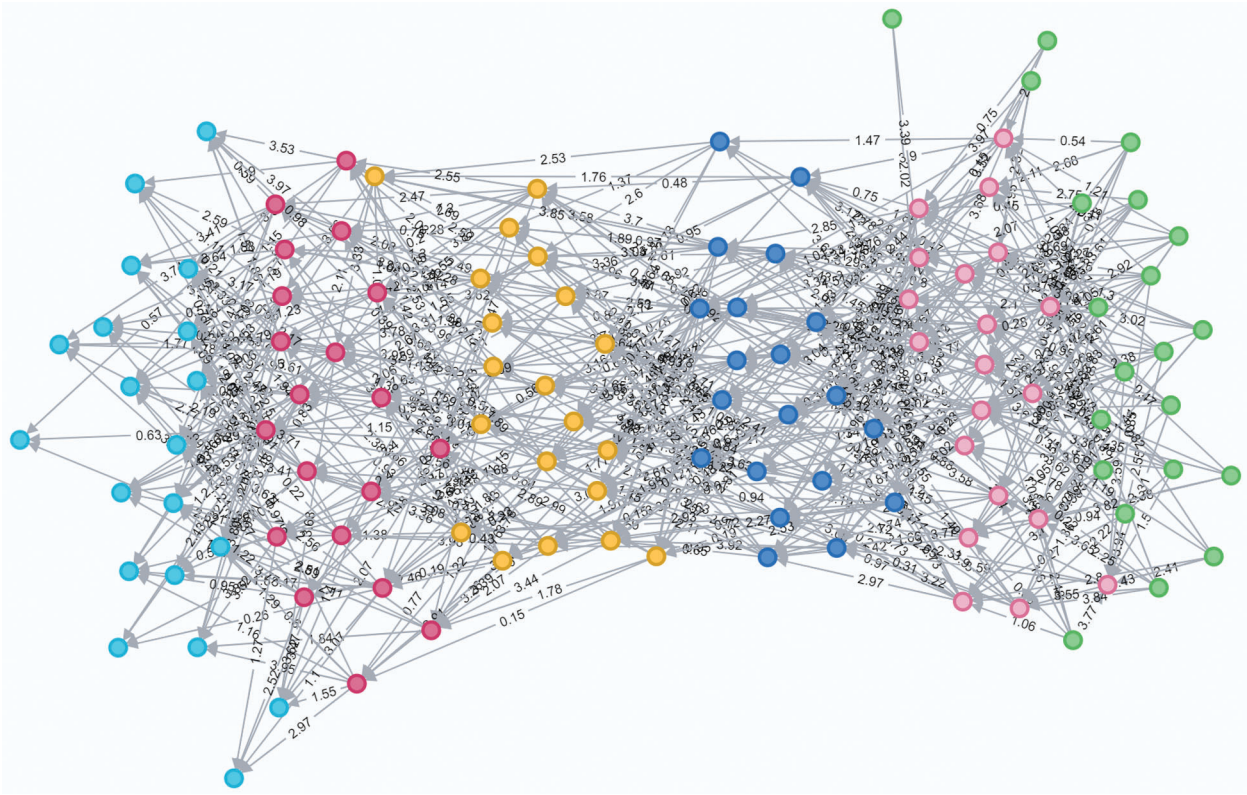

Figure 11: Snapshot of the larger graph for evaluating the scalability of the proposed approach

Table 8: Resulting graph triples of the simulation of the large graph

120 Graph triples						
A6 0.86 A36	A5 3.19 A26	A19 2.25 A34	A9 1.35 A26	A14 3.02 A37	A17 1.36 A29	A14 0.18 A21
A11 3.34 A37	A17 0.43 A26	A11 1.91 A34	A0 3.77 A26	A6 0.18 A38	A9 3.67 A29	A10 0.27 A21
A20 2.58 A24	A8 0.29 A26	A3 0.94 A34	A7 3.7 A27	A18 3.02 A38	A11 3.92 A29	A20 0.27 A21
A10 3.55 A24	A11 2.68 A26	A15 1.71 A34	A16 2.8 A27	A7 2.95 A38	A10 0.65 A29	A9 2.7 A21
A12 3.82 A24	A14 1.6 A26	A18 2.96 A35	A5 1.06 A27	A16 0.82 A38	A19 3.65 A30	A17 2.38 A21
A7 3.86 A24	A20 1.05 A26	A15 2.07 A35	A19 3.55 A27	A17 3.61 A38	A15 0.41 A30	A11 3.73 A21
A5 0.53 A24	A0 0.36 A27	A3 0.27 A27	A6 2.41 A29	A19 3.1 A38	A20 3.83 A30	A8 2.83 A22
A13 3.39 A33	A1 2.62 A27	A18 0.91 A27	A15 2.57 A29	A8 0.57 A38	A5 1.6 A30	A0 0.36 A22
A20 0.69 A33	A15 1.86 A24	A15 3.3 A37	A4 0.75 A35	A20 2.92 A38	A8 2.06 A30	A5 1.08 A22
A12 2.38 A34	A3 3.6 A25	A0 2.43 A37	A8 0.45 A35	A12 0.47 A38	A14 0.85 A31	A6 0.24 A22
A5 2.48 A34	A4 3.52 A25	A9 3.36 A37	A2 3.97 A35	A15 1.05 A39	A10 3.59 A31	A10 0.95 A22
A18 1 A34	A7 1.07 A25	A10 0.24 A37	A0 1.29 A35	A10 2.78 A39	A0 1.31 A31	A1 1.81 A22

Table 8 (continued).

120 Graph triples						
A7 0.95 A34	A14 0.37 A25	A16 1.19 A37	A3 2.83 A35	A4 2.53 A39	A5 3.77 A31	A11 0.24 A22
A7 2.6 A33	A15 2.1 A25	A12 1.35 A37	A6 2.11 A35	A20 1.21 A39	A11 2.55 A31	A16 0.48 A22
A15 0.41 A28	A3 3.14 A36	A7 0.31 A37	A19 0.22 A29	A14 2.93 A39	A3 3.94 A31	A14 1.07 A22
A14 0.25 A28	A8 2.87 A36	A1 1.92 A37	A3 2.78 A29	A1 0.83 A39	A9 1.5 A31	A17 1.23 A22
A2 2.32 A28	A18 2.38 A36	A3 3.62 A37	A0 2.77 A28	A6 0.54 A39	A16 2.41 A31	A2 2.95 A22
A18 1.3 A28	A14 0.61 A36	A2 0.15 A33	A11 0.44 A28	A7 1.36 A40	A2 3.68 A32	A9 2.12 A22
A7 3.98 A28	A11 1.27 A23	A15 0.28 A32	A18 3.93 A33	A9 2.22 A40	A12 2.02 A32	A15 3.92 A23
A10 2.85 A28	A10 1.13 A23	A8 0.8 A32	A19 3.81 A33	A19 3.84 A40	A1 2 A32	A2 2.11 A23
A8 2.07 A33	A1 3.05 A24	A16 1.43 A40	A5 2.54 A32	A10 1.23 A40	A0 2.13 A32	A1 2.79 A23
A7 3.98 A28	A11 1.27 A23	A9 1 A36	A17 2.69 A32	A1 0.82 A40	A13 2.02 A32	A6 2.68 A23
						A17 2.75 A23

Table 9: Results of the generated larger graph and the output message to decision support

Q1	Q2	Q3	Q4	Q5	% Node involved	% Violation	Message to user
40	154	83	8	64	100	47	There are high violations in the store at the time of this message. Urgent actions must be taken.

7 Conclusion

This paper presented a conceptual design for an indoor physical distancing monitoring system based on a graph database and edge processing. The IoT edge-based system architecture introduced consists of several components; each of the components is responsible for performing and managing a variety of tasks. These components comprise wearable devices and WSN infrastructure (sensing layer), data processing at edge points (middleware layer), and a cloud graph database (data storage layer). By using the BLE beacon technology in the proposed design for indoor detecting physical distancing, the network infrastructure presented has the advantages of low energy consumption, signal range and accuracy of BLE standards, and cloud beacons. A graph database is used to represent the distancing between individuals in a group, and distancing measures are implemented via graph database queries as an alternative solution to that of the traditional tracking and localization system. A Neo4j graph database system was used to conduct several experiments as proof of concept of the overall design idea. These system layers are glued by a strategy of model transformation aimed at generating the final executable Neo4j Cypher script code from a high-level graph model (intermediate). The mapping rules are implemented using the commonly known hybrid transformation language, ATL.

Experimental results indicate the proposed design would allow decision makers to detect COVID-19 social distancing measures within an enclosed area. The proposed system includes a notification sent to an authorized person informing them about the status of physical distancing measures at a specific time and advising when urgent action is required. Based on the recorded and analyzed data, the number of individuals permitted to be inside the monitored area at a specific time of the day or on a particular day of the week can be reduced. Conversely, the permitted number of individuals can be increased in areas in which rules of social distancing have been followed without violation.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] G. Pal, G. Li and K. Atkinson, "Multi-agent big-data lambda architecture model for e-commerce analytics," *Data*, vol. 3, no. 4, pp. 58, 2018.

[2] B. Leang, S. Ean, G. Ryu and K. H. Yoo, "Improvement of kafka streaming using partition and multi-threading in big data environment," *Sensors*, vol. 19, no. 1, pp. 134, 2019.

[3] G. J. Milne, S. Xie and D. Poklepovich, "A modelling analysis of strategies for relaxing COVID-19 social distancing," *MedRxiv*, vol. 2020, pp. 1–17, 2020.

[4] A. Polenta, P. Rignanese, P. Sernani, N. Falcionelli, D. N. Mekuria *et al.,* "An internet of things approach to contact tracing—the bubbleBox system," *Information*, vol. 11, no. 7, pp. 347, 2020.

[5] Z. Allam, G. Dey and D. S. Jones, "Artificial intelligence (AI) provided early detection of the coronavirus (COVID-19) in china and will influence future urban health policy internationally," *AI*, vol. 1, no. 2, pp. 156–165, 2020.

[6] A. Bekkali and M. Matsumoto, "RFID indoor tracking based on inter-tags distance measurement," *Wireless Telecommunications Symposium*, IEEE, Pomona, CA, pp. 1, 2007.

[7] T. Y. H. Angela, V. Viswanathan, M. Lees and W. Cai, "Analyzing the effectiveness of wearable wireless sensors in controlling crowd disasters," In *Procedia Computer Science,* vol. 29, Elsevier, Netherlands, pp. 1590–1599, 2014.

[8] T. Ahmed, T. B. Pedersen and H. Lu, "Finding dense locations in indoor tracking data," in *IEEE 15th Int. Conf. on Mobile Data Management*, vol. 1, IEEE, Brisbane, Australia, pp. 189–194, 2014.

[9] X. Xu, S. Ding and T. Sun, "A fast density peaks clustering algorithm based on pre-screening," in *IEEE Int. Conf. on Big Data and Smart Computing (BigComp)*, IEEE, Shanghai, China, pp. 513–516, 2018.

[10] C. Martella, M. Van Steen, A. Van Halteren, C. Conrado and J. Li, "Crowd textures as proximity graphs," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 114–121, 2014.

[11] M. Nikodem, M. Słabicki, T. Surmacz, P. Mrówka and C. Dołęga, "Multi-camera vehicle tracking using edge computing and low-power communication," *Sensors*, vol. 20, no. 11, pp. 1–16, 2020.

[12] F. Jamil, S. Ahmad, N. Iqbal and D. H. Kim, "Towards a remote monitoring of patient vital signs based on IoT-based blockchain integrity management platforms in smart hospitals," *Sensors*, vol. 20, no. 8, pp. 1–26, 2020.

[13] J. Kolakowski, V. Djaja-Josko, M. Kolakowski and K. Broczek, "UWB/BLE tracking system for elderly people monitoring," *Sensors*, vol. 20, pp. 6, 2020.

[14] N. Vallez, S. Krauss, J. L. Espinosa-Aranda, A. Pagani, K. Seirafi *et al.,* "Automatic museum audio guide," *Sensors*, vol. 20, no. 3, pp. 779, 2020.

[15] H. Zhang, Z. Zhang, N. Gao, Y. Xiao, Z. Meng *et al.,* "Cost-effective wearable indoor localization and motion analysis via the integration of UWB and IMU," *Sensors*, vol. 20, no. 2, pp. 1–23, 2020.

[16] L. Shen, Q. Zhang, J. Pang, H. Xu, P. Li *et al.,* "ANTspin: Efficient absolute localization method of RFID tags via spinning antenna," *Sensors*, vol. 19, no. 9, pp. 1–20, 2019.

[17] J. H. Huh and K. Seo, "An indoor location-based control system using bluetooth beacons for IoT systems," *Sensors*, vol. 17, no. 12, pp. 1–22, 2017.

[18] F. S. Daniş and A. T. Cemgil, "Model-based localization and tracking using bluetooth low-energy beacons," *Sensors*, vol. 17, no. 11, 2017.

[19] O. Belmonte-Fernández, A. Puertas-Cabedo, J. Torres-Sospedra, R. Montoliu-Colás and S. Trilles-Oliver, "An indoor positioning system based on wearables for ambient-assisted living," *Sensors*, vol. 17, no. 12, pp. 36, 2017.

[20] M. Lewandowski, B. Płaczek, M. Bernas and P. Szymała, "Road traffic monitoring system based on mobile devices and Bluetooth low energy beacons," *Wireless Communications and Mobile Computing*, Egypt, vol. 2018, pp. 12, 2018.

[21] B. Yu, L. Xu and Y. Li, "Bluetooth low energy (BLE) based mobile electrocardiogram monitoring system," in *IEEE Int. Conf. on Information and Automation*, IEEE, Chengdu, China, pp. 763–767, 2012.

[22] D. Zhang, L. T. Yang, M. Chen, S. Zhao, M. Guo *et al.,* "Real-time locating systems using active RFID for internet of things," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1226–1235, 2014.

[23] G. Paolini, D. Masotti, F. Antoniazzi, T. S. Cinotti and A. Costanzo, "Fall detection and 3-D indoor localization by a custom RFID reader embedded in a smart e-health platform," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 12, pp. 5329–5339, 2019.

[24] M. El-Absi, A. A. H. Abbas, A. Abuelhaija, K. Solbach and T. Kaiser, "Distance and tag aware localization in indoor terahertz systems," in *First International Workshop on Mobile Terahertz Systems (IWMTS)*, IEEE, Duisburg, Germany, pp. 1–5, 2018.

[25] P. Kriz, F. Maly and T. Kozel, "Improving indoor localization using Bluetooth low energy beacons," *Mobile Information Systems*, vol. 2016, pp. 1–11, 2016.

[26] P. Spachos and K. N. Plataniotis, "BLE beacons for indoor positioning at an interactive IoT-based smart museum," *IEEE Systems Journal*, vol. 14, no. 3, pp. 3483–3493, 2020.

[27] A. Jabbari, K. J. Almalki, B. Y. Choi and S. Song, "ICE-MoCha: Intelligent crowd engineering using mobility characterization and analytics," *Sensors*, vol. 19, no. 5, pp. 1025, 2019.

[28] F. Zafari, A. Gkelias and K. K. Leung, "A survey of indoor localization systems and technologies," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2568–2599, 2019.

[29] B. Matthias, F. Burkert, F. Schmidt, S. Hinz, D. Hartmann *et al.,* "Integrating pedestrian simulation, tracking and event detection for crowd analysis," in *IEEE Int. Conf. on Computer Vision Workshops (ICCV Workshops)*, IEEE, Barcelona, Spain, pp. 150–157, 2011.

[30] V. Galpin, N. Zoń, P. Wilsdorf and S. Gilmore, "Mesoscopic modelling of pedestrian movement using carma and its tools," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 28, no. 2, pp. 1–26, 2018.

[31] Y. Jia, Y. Qi, H. Shang, R. Jiang and A. Li, "A practical approach to constructing a knowledge graph for cybersecurity," *Engineering*, vol. 4, no. 1, pp. 53–60, 2018.

[32] K. Jiang, D. Yang, C. Liu, T. Zhang and Z. Xiao, "A flexible multi-layer map model designed for lane-level route planning in autonomous vehicles," *Engineering*, vol. 5, no. 2, pp. 305–318, 2018.

[33] A. F. Subahi, "Edge-based IoT medical record system: Requirements, recommendations and conceptual design," in *IEEE Access*, vol. 7, IEEE, USA, pp. 94150–94159, 2019.

[34] M. Younas, "Research challenges of big data," *Service Oriented Computing and Applications*, vol. 13, no. 2, pp. 105–107, 2019.

[35] A. F. Subahi and K. E. Bouazza, "An intelligent IoT-based system design for controlling and monitoring greenhouse temperature," in *IEEE Access*, vol. 8, IEEE, USA, pp. 125488–125500, 2020.

[36] I. Robinson, W. Jim and E. Emil, *Graph Databases: New Opportunities for Connected Data*, O'Reilly Media, USA, 2015.

[37] Y. W. Chiou, "APIs for XML: DOM, SAX, and JDOM," in *Perspectives on Information Systems Development*, Boston, MA: Springer, pp. 299–308, 2002.

[38] F. Jouault and I. Kurtev, "Transforming models with ATL," in *Int. Conf. on Model Driven Engineering Languages and Systems*, Berlin, Heidelberg: Springer, pp. 128–138, 2005.