Building Rhythms

Reopening the workspace with indoor localisation

Group 4

Synthesis Project
GEO1101
June 2021

TU Delft

Building Rhythms

Reopening the workspace with indoor localisation

by

Group 4

Guilherme Spinoza Andreo:	5383994
Ioannis Dardavesis:	5372666
Pratyush Kumar:	5359252
Michiel de Jong:	4376978
Maundri Prihanggo:	5151279
Georgios Triantafyllou :	5381738

Project supervisor:	Edward Verbree
Campus Real Estate:	Bart Valks, Adriaan Jung
Esri Nederland:	Niels van der Vaart
CGI Nederland :	Bart Staats, Martijn Koopman & Robert Voûte

TU Delft

Preface

The Geomatics Synthesis Project is a two month group project for the students in the fourth quarter of the Master of Science Geomatics for the Built Environment at the Delft University of Technology. The project gives students the opportunity to apply their knowledge obtained from all of the core courses throughout the master track to a real-world project, while also obtaining valuable experience with project management, data management, stakeholder and client involvement. This final report has the purpose to describe the entire process of the project, with the objective to provide a general overview for the students, stakeholders, clients, supervisors and any other party involved.

Acknowledgements

Before we begin, we would like to express our gratitude towards all of the people that were involved with our project: Building Rhythms - Reopening the work-space.

Firstly, we would like to thank our Synthesis Project coordinator Edward Verbree, who provided excellent supervision during the entire course of this Building Rhythms project. Additionally, we would like to thank Niels van der Vaart, the representative of Esri Nederland and guest lecturer of TU Delft,for his help and support with all of the issues that we had encountered with our project.

We would also like to acknowledge our clients, Bart Valks and Adriaan Jung, of the department of Real Estate of TU Delft for providing us with the opportunity to work on this project, as well as their contributions, feedback and cooperation, which were essential for this project's success.

We also want to express our gratitude to our stakeholder, CGI Nederland, particularly, Bart Staats, Martijn Koopman and Robert Vôute, for their guidance and feedback on the various different aspects of this project.

We would also like to give a special thanks to Abdullah Alattas, Martijn Meijers, Suzanne Hessels and Aytaç Balci. Their cooperation, interest and participation was crucial for development of our project. Last but not least, we extend our thanks to Bastiaan van Loenen, who as programme director of the Geomatics programme at TU delft, gave us the opportunity to participate in this synthesis project.

Contents

BUILDING RHYTHMS: REOPENING THE WORKSPACE WITH INDOOR LOCALISATION

Guilherme Spinoza Andreo, Ioannis Dardavesis, Michiel de Jong, Pratyush Kumar, Maundri Prihanggo, Georgios Triantafyllou

Supervisors: ir. Edward Verbree, Niels van der Vaart

KEY WORDS: Indoor localisation system, Wi-Fi fingerprinting, ArcGIS, Indoor model, Machine Learning

ABSTRACT

Indoor localisation methods are an essential part for the management of COVID-19 restrictions, social distancing, and the flow of people in the indoor environment. Moving towards an open work space in this scenario requires effective real-time localisation services and tools, along with a comprehensive understanding of the 3D indoor space. This project's main objective is to analyse how ArcGIS Indoors can be used with location awareness methods to elaborate and develop space management tools for COVID-19 restrictions in order to reopen the workspace for TU Delft Campus. This was accomplished by using six Arduino micro controllers, which were programmed in C++ to scan all available Wi-Fi fingerprints in the east wing of the Faculty of Architecture and the Built Environment of TU Delft and send over the data to an ArcGIS Indoor Information Model (AIIM). The data stored on the AIIM is then accessed using the app on the user's Android device using REST Application Programming Interface (API) where a kNN based matching algorithm then identifies the location of the user. The results show that the localisation is not consistent for rooms that are directly above each other or share common access points. However, when functioning to locate different tables inside a room, the system proved to uniquely distinguish between the specific tables. As a result, we can conclude that based on the size of the rooms, more Arduino devices should be installed to achieve an ideal accuracy. Finally, recommendations are made for the continuation of this research.

1. INTRODUCTION

Recently, the success of outdoor positioning methods has provided the means for high accuracy positioning technologies research to slowly shift towards the indoor environment. This new wave of indoor localisation technologies has been growing at a rapid pace, however, most of them are considered to still be in the early phases of development (Xia et al.,2017). The core aspect of Wi-Fi fingerprint localisation systems is established on the set of measurements of the signal strength from different Wireless Access Points (WAPs). They can be used as a radio-map, or reference points, that can be applied to estimate and match a location of a given device according to its signal strength (Torres et al.,2016).

Indoor models, in broad terms, are a hierarchy of unique classes of different building elements, spaces, topological and semantic information (Liebich 2009 cited in Tran, H., et al, 2019). Specifically, the topological information model of a building, which can be obtained from either CAD or BIM data, is the first component of creating an indoor localisation and navigation system. These 3D models are vital for the development of navigation assistance applications and emergency responses (Tran, H., et al, 2019).

Indoor localisation methods are an essential part for the management of COVID-19 restrictions, social distancing, and the flow of people in the indoor environment. To move towards an open workspace in a restricted occupancy scenario requires effective real-time localisation services and tools, along with a comprehensive understanding of the 3D indoor space.

This project consists of three parts which have their own goals and expectations. Accordingly, all of them are combined and each of them subdivided in more specific sections in order to form the final product. The initial one is the main goal of the "Synthesis Project" course from the MSc Geomatics which aims to combine and apply all the knowledge, skills and insights gained during the core programme.

The second part relates to the wider TU Delft project "Building Rhythms" which aims to investigate the impact of COVID-19 on campus life through public data visualisations with high respect to privacy preservation. Although the general project targets all of the campus, in this project, the focus is only about the indoor environments.

The third and last part is set by the client and stakeholders, mainly by Esri Nederland. Therefore, one of the main goals of the project is to explore the possibilities and capabilities of ArcGIS and ArcGIS Indoors with the combination of a real-time indoor location system with Fingerprinting produced by open source microcontrollers (Arduino MKRWiFi) and a native mobile application.
To sum up, the combination of these three parts leads to the main research question of the project:

"How can ArcGIS Indoors be combined with indoor Wi-Fi fingerprinting localisation awareness techniques to create a real-time application for understanding COVID-19's impact in the indoor environment of TU Delft with high respect for privacy of the users?"

2. METHODOLOGY

The methodology of the project can be summarized into five key stages: Data Acquisition, Indoor modelling, Indoor localisation with Wi-Fi fingerprinting, Testing and Visualisation. The indoor model was based on CAD files with minor adjustments, in order to represent the current floor-plans of the Faculty. The CAD files, combined with an excel file that contains information about the layers, were imported into

ArcGIS Pro, so that the CAD layers could be transformed into GIS layers.

To accomplish the scope of this project, only the ground and first floor of the Faculty's east wing were investigated. Furthermore, the database, which contains the information about the rooms of the building, was enriched with additional information, such as capacity, occupancy and room types, provided by TU Delft Campus Real Estate and Abdullah Alattas (Alattas et al., 2018). Points of Interest were created for rooms of high importance, such as offices and lecture rooms, based on which the interior network of the model was created. The created model was published in ArcGIS online portal, as a web scene, which will work as the main view of the android application and the part of the end product of this project.

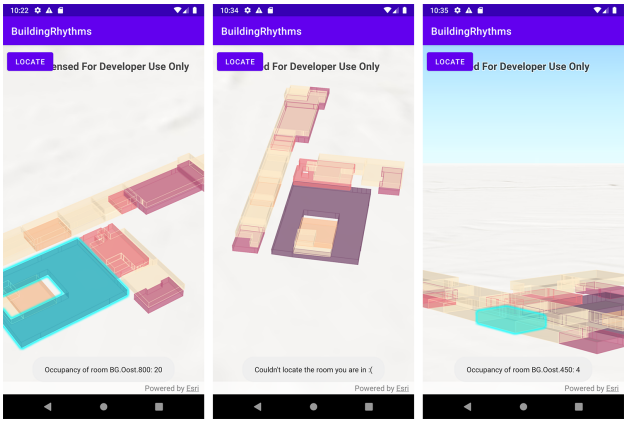

Figure 1: Android Application

To find out if live radio maps combined with Wi-Fi fingerprinting can accurately be used as an indoor localisation system, we use a setup where the rooms we want to test are equipped with a Wi-Fi enabled microcontroller (Arduino MKR 1010 Wi-Fi) which creates a live radio map of the static Wi-Fi networks at the Campus, and sends this to the server.

This data is then used as input for a simple user app we developed that uses a local user Wi-Fi scan and a kNN machine learning algorithm to match the user to a fingerprint location which is semantically linked to the indoor model. This matching is done locally on the user device, ensuring no user Wi-Fi data is being used anywhere except on the users phone itself. Using this, we can then update the indoor model with live occupancy data on a room level of detail, ensuring that no privacy sensitive data is sent to the server.

Because access to the Wi-Fi chip was paramount for the development of the user app, and with Apple not opening this up for developers in iOS, this led us to start the development of an Android user app. The home screen of the user app is a 3D Scene of the Indoor Model of the study area which is directly added from ArcGIS Online. The main functionality of the app is to provide a visual representation of the real time occupancy of the study area, and to allow the user to locate themselves in this study area (given that they are in a room that is covered by the system).

The ArcGIS Online REST API was used to load the table which contains the updated Wi-Fi fingerprints from the Arduino sensors located at different nodes, being hosted on the ArcGIS Online server. The HTTP request returns a JSON string which is then parsed to be loaded into a table, with the columns for MAC address, RSSI values, and Room ID. The Wi-Fi signals received from the Android device are parsed as a JSON string, and loaded directly to a table, which is then cleaned to contain two columns, one for the MAC addresses and one for the signal. Once this is done, the x and y variables for the kNN algorithm are extracted. This is done by the following means: Firstly, in order to take the MAC addresses into account when running a matching algorithm, all the MAC IDs are aggregated and each unique MACaddress is given a unique number. The x variables for training data set then are the Unique MAC numbers and their corresponding RSSI values. The y-variable is the label containing the rooms name from which a particular MAC number and a RSSI was obtained. For the test data-set, only the x-variables need to be created, which should be in the same format as the training data-set, so a unique MAC identifier and RSSI values are needed. To prepare the x-variables for the test data, the MAC addresses are matched to the fingerprint MAC addresses and the unique number assigned to a MAC in fingerprint is given to matching MAC addresses from the test data-set. If a MAC in test data cannot be found in fingerprint data, it is discarded along with its RSSI values, as that particular MAC address would not help us in matching. The y-variable in the case of the test data-set is supposed to be our final label for the room to locate the user's position.

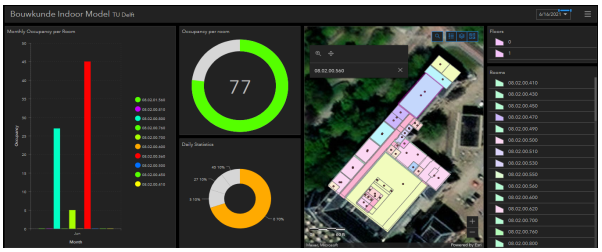

Figure 2: ArcGIS Online Dashboard

On the back-end, to actually implement the updating of the occupancy without storing any more data then needs be, a simple updating function was developed. This function uses the output of the matching algorithm, which will either be a string with the NAME attribute of a feature, or it will output the error string *"Couldn't locate the room you are in :("*. If the output is the error string, nothing happens, as no new occupancy data has not been generated. If the output of the matching algorithm is not the error string, the updating function will be called once or twice, depending on if there is a previous room known to the app or not. If the previous room is known, the app will decrement the occupancy in the previous room, and increment the occupancy in the current room.

One of the main goals of this project was to investigate and analyse the impact of COVID-19 in the indoor environment. The manager of a building, the Faculty of Architecture of TU Delft in this case, should be able to have access to statistics, including movement patterns inside a building, as well as occupancy data, so that the peak hours of movement and occupancy could be identified. This data aids towards optimizing space planning and avoiding overpopulation of rooms. To accomplish that, a dashboard was created, additionally to the main application, in order to provide hourly, daily and monthly statistics, based on occupancy data provided by the Arduino sensors. The dashboard uses the created indoor model of the map and provides several capabilities, such as monthly and daily occupancy per room, as well as information on which rooms are the most populated at the current time. The dashboard is interactive and the included graphs are updated real-time with a small refresh rate.

3. RESULTS

A fully functional hosted indoor model has been implemented using ArcGIS Indoors, with the initial testing of the Android app has proved to successfully obtain the Wi-Fi fingerprints from the selected research area and match them statically to a fingerprint in the database. After expanding to real-time fingerprinting we found that while for some rooms, the accuracy is very high, for other rooms the accuracy is far lower, due to a number of reasons. The accuracy of the system of both testing setups is presented in table 1 and 2 below.

Room ID	Accuracy in center [%]	Accuracy along sides [%]
BG.Oost.560	100.00	100.00
01.Oost.560	10.00	0.00
BG.Oost.700	30.00	40.00
BG.Oost.800	100.00	100.00

Table 1: The accuracy of our indoor localisation system with n=10

Table ID	Accuracy in center[%]
2.031.O	100.00
2.032.O	100.00
2.033.O	100.00
2.034.O	100.00

Table 2: The accuracy of our indoor localisation system within a room with n=3

After looking at table 1 it would seem that the localisation system we designed is not functioning well, because there is low accuracy on at least two out of four rooms upon first glance. However, when diving into the testing deeper we do see that there are some reasons for this outcome, and then these results are not unexpected.

The first thing to consider is the placement of the rooms, namely the pairs BG.Oost.560 and 01.Oost.560 and BG.Oost.700 and BG.Oost.800. For the two .560 rooms we know that these two rooms are directly above each other, which leads to lots of shared access points, and could lead to a less unique fingerprint for each room, leading the fingerprint matching algorithm to wrongfully assign the BG.Oost.560 room when in the 01.Oost.560 room and vice versa.

In the case of 700 and 800, the fact that often when the user was in 700, the algorithm would match the user to 800 erroneously is also due to their proximity, with room 700 sharing a wall with windows with room 800. This could also be the reason that the accuracy for room 700 is higher near the walls of the room, the 40% accuracy are the 4 measurements taken nearer to 800 than to the location of the Arduino.

Another thing to consider is the fact that the ratio of different rooms varies quite a bit, with an overrepresentation of room BG.Oost.560, and an under representation of room 01.Oost.560, leading us to theorise that, this is another reason that 01.Oost.560 is has a very low accuracy. To solve this, in the second testing setup we tried to watch the ratio of the Arduinos feeding their data, and the high accuracy presented in table 2 shows this does have an effect.

4. DISCUSSION AND CONCLUSIONS

With this project, we demonstrate the development of an ArcGIS Indoor Model created from CAD data in ArcGIS, which is later applied to an external Android app to display real-time occupancy of the Faculty for COVID-19 scenarios. The process of creating the ArcGIS Indoor Model was easy to follow, given the information provided by Esri, however, there were some issues with the initial structure of the CAD data, therefore, for it to function properly within the ArcGIS Indoors software required some manual adjustments. We have yet to conclude our project, however we believe that there is potential to expand and implement it as a privacy-preserving indoor localisation system, with location information available on a room scale. We think that the system could be expanded to include routing and navigation.

When defining the product, it became quite clear, early on, that we would need to design, develop and test a system that would try to answer the need created in the research question, namely: the combination of ArcGIS Indoors with a privacy proof Wi-Fi fingerprinting localisation technology. To this extent, we have achieved what we wanted, with different levels of success on individual aspects of the system design.

On a more specific level, the system isn't exactly production ready. But we did make a big initial step to show that a reasonably accurate, versatile, cheap and scalable system can be created to provide an answer for the need stated in the research question. In terms of privacy, which was a very important value and key reason the entire system is set up this way, as opposed to using something like Indoors (the native ArcGIS IPS), we have achieved what we wanted. There is no personal data required to display and store occupancy information for an indoor environment like the Faculty of Architecture. Furthermore, we managed to completely separate the client and server side in the sense that the user is not required to share any data with the server concerning Wi-Fi or location as of yet.

Furthermore, we have developed a robust pipeline for converting often readily available CAD data into a semantically rich indoor model suitable for indoor localisation on a room level, using ArcGIS Pro/Indoors. This pipeline can be applied to any indoor environment for which this data is available, which is essential for the scalable nature of the system in its entirety.

In a technical sense, we have shown that using a single scanning point inside a room to create live radio maps can lead to a reasonable accuracy level, and deploying this setup can yield similar levels of accuracy as when constructing a very time consuming radio map at regular intervals as in the traditional Wi-Fi fingerprinting setup. With our system having lower initial time investing, as the scanning devices operate independently and autonomously, this project serves as a proof of concept.

5. REFERENCES

Alattas, A.; Oosterom, P. V., Zlatanova, S.; Diakité, A. A., & Yan, J. (2018). Developing a database for the LADMIndoorGML model (tech.rep.) . APA. https://repository.tudelft.nl/islandora/object/uuid:31a20fb8dabc4f1982c8432f410a3ece?collection=research

Torres, J., Belmonte, Ó., Montoliu, R., Trilles, S., & Calia, A. (2016). How feasible is wifi fingerprintbased indoor positioning for inhome monitoring?2016 12th International Conference on Intelligent Environments (IE), 68–75. https://doi.org/10.1109/IE.2016.19

Tran, H., Khoshelham, K., Kealy, A., & DíazVilariño, L. (2019). Shape grammar approach to 3d modeling of indoor environments using point clouds.Journal of Computing in Civil Engineering,33(1),04018055.https://doi.org/10.1061/(ASCE)CP.19435487.0000800

Xia, S., Liu, Y., Yuan, G., Zhu, M., & Wang, Z. (2017). Indoor fingerprint positioning based on wifi: An overview. ISPRS *International Journal of GeoInformation*, 6 (5), 135. https://doi.org/10.3390/ijgi6050135

2

Introduction

2.1. Problem Definition

Recently, the success of outdoor positioning methods have provided the means for high accuracy positioning technologies research to slowly shift towards the indoor environment. This new wave of indoor localisation technologies has been growing at a rapid pace, however, most of them are considered to still be in the early phases of development (Xia et al., 2017).

While the integration of Global Navigation Satellite System (GNSS) in positioning devices have made outdoor positioning and localisation a reality, precise indoor localisation remains an issue to be resolved for general-purpose applications (Torres-Sospedra et al., 2019). Furthermore GNSS are not effective in relation to indoor and dense urban environments, since the signal paths can be disrupted, the amount of signals are greatly decreased and have an increase in non-line-of-sight (NLOS) propagation, which results in a decreased availability for Indoor Positioning System (IPS)(GEO1003, Verbree, 2021).

The two most common methods used to implement IPS are either based on applying and installing a particular hardware in the desired location or to utilise an existing sensory system that is already in place within the location device to try to determine the current position. For both cases, the use of fingerprinting has been generally applied for indoor positioning (Torres-Sospedra et al., 2019). The core aspect of Wi-Fi fingerprint positioning systems is established on the set of measurements of the signal strength from different Wireless Access Points (WAPs). They can be used as a radio-map, or reference points, that can be applied to estimate and match a location of a given device according to its signal strength (Torres et al., 2016).

Indoor models, in broad terms, are a hierarchy of unique classes of different building elements, spaces, topological and semantic information (Liebich 2009 cited in Tran et al., 2019). Specifically, the topological information model of a building, which can be obtained from either CAD or BIM data, is the first component of creating an indoor positioning and navigation system. These 3D models are vital for the development of navigation assistance applications and emergency responses (Tran, H., et al, 2019). Indoor positioning and localisation methods are an essential part for the management of COVID-19 restrictions, social distancing, and the flow of people in the indoor environment. To move towards an open workspace in a restricted occupancy scenario requires effective real-time location-based positioning services and tools, along with a comprehensive understanding of the 3D indoor space. The challenge with providing a real-time indoor location-based positioning system with Wi-Fi fingerprinting is still great, since they require accurate up-to-date radio-maps.

2.2. Objectives and Research questions

The project consists of three parts, each having their own goals and expectations. Accordingly, all of them are combined and each of them subdivided in more specific sections in order to form the final product.

The initial one is the main goal of the ”Synthesis Project” course from the MSc Geomatics which aims to combine and apply all the knowledge, skills and insights gained during the core programme.

The second part relates to the wider TU Delft project ”Building Rhythms” which aims to investigate the impact of COVID-19 on campus life through public data visualisations with high importance given to preserving the privacy of the individuals in the building. Although the general project targets the entire campus, in this project, the focus is only about the indoor environments.

The third and last part is set by the client and stakeholders, mainly by Esri Nederland. Therefore, one of the main goals of the project is to explore the possibilities and capabilities of ArcGIS and ArcGIS Indoors in combination with a real-time indoor location system using Fingerprints produced by open-source micro-controllers (Arduino MKR-WiFi) and a native smartphone application.

To sum up, the combination of these three parts leads to the main research question of the project:

”How can ArcGIS Indoors be combined with indoor Wi-Fi fingerprinting localisation awareness techniques to create a real-time application for understanding COVID-19’s impact in the indoor environment of TU Delft with high respect for privacy of the users?”

2.3. Reading Guide

This report document is composed of 8 chapters and concludes all the theoretical background of the project as well the methodology that had been followed and all the critical decisions were taken through the process with the appropriate reasoning. Furthermore, the final results and analysis is presented as well as the ideas and proposals for future work and research.

Following this Chapter 2, where the project’s general concepts, objectives and research questions are introduced, Chapter 3 presents key topics that are a part of our project and provides a theoretical background, while also discussing the previously developed projects of similar study and compares them to our own.

Thereafter, Chapter 4 elaborates upon the methodology of the project. Specifically, this chapter is further divided into sections and subsections according to the different parts of the project’s implementation. Firstly, a short overview of all components are described and then continues with the ”Indoor Model” creation using ArcGIS Indoors, the ”Live radio maps: Arduino”, the ”Server side: ArcGIS Online”, the ”Fingerprinting matching: kNN Algorithm”, the ”User side: Android User Application” as well as the ”User side: ArcGIS Dashboard” and lastly the ”Testing the App” . Consequently, all these sections follow the same logic, which is stating all the key parts of each of them, as well as the analytic process of what occurred. Likewise, all the limitations and issues faced through the process and the critical decisions that were taken and solutions were implemented are presented in this chapter.

In Chapter 5, the final results of the project are presented and more specific a demonstration of the final application is shown. The results that were obtained are then examined and evaluated in a way to reach conclusions about the accuracy of the App in different patterns of testing.

During Chapter 6 discusses the final observations and thoughts of the research and its results. Additionally, a further analysis whether the initial goals of the project were achieved is done.

Finally, Chapter 7 analyses and proposes all the possible future work that could be done for the improvement of the project in general. Specifically, is divided into several subsections in order to cover every aspect of the project

3

Theory and Context

This chapter provides the context of the theoretical concepts for this research and shows the relationship between our project and the previous research related to the development of Indoor Localisation Systems using Wi-Fi fingerprint data. This section aims to distinguish our project in comparison to the previously developed work, while also providing the necessary information about the concepts that were applied to this research.

3.1. GIS - Indoor GIS

According to Esri (ESRI, 2021c):

> ”A Geographic Information System (GIS) is a framework for gathering, managing, and analysing data. Rooted in the science of geography, GIS integrates many types of data. It analyses spatial location and organises layers of information into visualisations using maps and 3D scenes. With this unique capability, GIS reveals deeper insights into data, such as patterns, relationships, and situations, helping users make smarter decisions.”

The first application used in this project was ArcGIS Pro (ESRI, 2021b), which is Esri's desktop GIS application for users. It enables the users to perform all of the required data visualisation, analysis and tools to produce powerful 2D or 3D spatial information. ArcGIS Indoors (ESRI, 2021a), on the other hand, is used to develop indoor mapping systems from Computer Aided Design (CAD) and Building Information Model (BIM). The resulting product is a floor-aware indoor map that can be used inside ArcGIS Online,which is an online cloud-based mapping and analysis platform.

In the context of this research and the parties involved, all of the aforementioned software tools will be used to generate the necessary ArcGIS Indoor Information Model (AIIM), that will be integrated with the end product API. The function of the AIIM model is to provide an interactive model structure inside of ArcGIS online and to also serve as the main database component in which the matched Wi-Fi fingerprints of the rooms will be identified to.

3.2. Indoor Localisation Systems & Wi-Fi Fingerprinting

The extensive distribution of Wi-Fi has considerably advanced the potential of indoor localisation technologies available to the market, as it provides a low-expense, high-precision and low-energy solution for indoor localisation (Xia et al., 2017) and, moreover, are openly available everywhere.

However, with a variety of different Indoor Positioning Methods, like Bluetooth Low Energy (BLE) positioning, Wi-Fi monitoring and Ultra-wide band Positioning, among others, for the purpose of this research, Wi-Fi fingerprinting was evaluated and selected to fulfil the research objectives and scope.

According to Mautz, 2012:

> ”one of the crucial elements for any initiative to design an indoor positioning system is a thorough study of the user requirements and specific application descriptions in order to justify the research and development in this field.”

For this research, specific user requirements were taken into consideration. Firstly, the project had to be developed with the existing Wi-Fi infrastructure of the TU Delft Faculty of Architecture and the Built Environment, also, given of the limited amount of time, resources and experience with other indoor positioning systems, the IPS should have been low cost, accurate and complementary to this infrastructure. Secondly, the efficiency and effectiveness of the IPS in preserving users' privacy and information without additional encrypting, data storage or anonymisation, was a decisive factor to determine the best IPS. These aspects, in combination with other secondary factors, such as availability and accuracy, led us to select Wi-Fi fingerprinting as the framework for this project's IPS.

Figure 3.1: User requirements

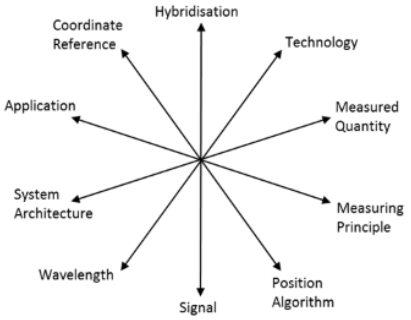

Figure 3.2: Important technical parameters (less significant to the user)

Wi-Fi fingerprinting is a widely known method for indoor positioning based on Wi-Fi Recceived Signal Strength Indicator (RSSI) signals. This method depends on the signal strengths of different access points in order to create a unique signature map of different Wi-Fi fingerprints of a location. Practically, the measured RSSI/MAC combinations are compared to a data-set of identifiable and unique RSSI/-MAC combinations of known locations. When a match between the measured and known observations is found, it can be estimated if the user is at a certain location. In this project's case, to create such a data-set, micro-controllers with Wi-Fi sensors are used to gather real time Wi-Fi fingerprints (Verbree, 2021).

3.3. Machine Learning

According to Mohri et al., 2012/2018, machine learning can be broadly defined as computational methods that use experience and past information available to the learner. This information is collected and made available for analysis, so that accurate predictions can be achieved.

Jordan and Mitchell, 2015, introduces several common methods that are widely known in machine learning approaches, which can be primarily subdivided into supervised and unsupervised methods. Regarding supervised learning methods, training data takes the form of a collection of (x, y) pairs. The ultimate goal is to achieve a prediction $y*$ in response to a query $x*$, which in other words refers to data labelling. Other efforts have also been made to develop deep learning algorithms that discover useful representations of the input without the need for labelled training data. This is referred to as unsupervised learning methods. Generally, unsupervised learning involves the analysis of unlabelled data under assumptions about the structure of the data (e.g. algebraic, combinatorial or probabilistic). Another key machine learning method is reinforcement learning, in which, the information available in the training data is intermediate between supervised and unsupervised learning. Instead of training examples that indicate the correct output for a given input, the training data in reinforcement learning, are assumed to solely provide an indication as to whether an action is correct or not.

Several techniques are implemented by machine learning approaches, such as classification, regression, and clustering. Classification and regression are two common tasks that are accomplished by the supervised learning approach. Mohri et al., 2012/2018, explains that classification is used for assigning a category to each item. For example, document classification consists of assigning a category such as politics, business, sports, or weather to each document, while image classification consists of assigning each image to a category such as car, train, or plane. Regression is used for assigning the problem of predicting a real value for each item, examples of regression include prediction of stock values or that of variations of economic variables.

Both classification and regression are accomplished by using k-Nearest Neighbour (kNN) algorithms. A kNN algorithm is used to classify unlabelled observations by assigning them to the class of the most similarly labelled examples (Zhang, 2016). Several studies show that the kNN algorithm outputs an adequate result in terms of accuracy.

Brownlee, 2016, mentions that kNN makes predictions using the training dataset directly. Specifically, predictions are made for a new data point, by searching through the entire training set for the k most similar instances (the neighbours) and summarising the output variable for those k instances. Distance measurement between the test dataset with the k instances is measured to determine the similarity. While the most popular distance equation is the Euclidean, several other distance measurement equations can be used, such as the Manhattan and the Minkowski distance, etc. Besides the simplicity of kNN algorithm, there are some drawbacks. The algorithm is dependent on selecting an optimal k value, which can be implemented solely by computationally expensive techniques, such as cross validation. Last but not least, the algorithm is also affected adversely by noise and is sensitive to irrelevant features, while the performance also varies according to size, since all data must be revisited. (Singh et al., 2016).

The kNN algorithm is one of the most common ways to obtain the location of a user's device in Indoor Localisation Systems (Bahl and Padmanabhan, 2000). Especially when using Wi-Fi fingerprinting methods that require the use of radio maps, the kNN algorithm will be powerful enough to check the similarity between the RSSI values and MAC addresses obtained from user's device at a location and RSSI values and MAC addresses from the radio map of the building or rooms in question. As with similarity measures, machine learning approaches are often able to show the highest accuracy, at the cost of training data volume and robustness (Burgess, 2019)

3.4. Privacy And Data Protection

The right to privacy is mentioned on the European Commission on Human Rights (ECHR) article 8 where "everyone has the right to respect his private and family life, his home". The ECHR specifically elaborates the meaning of private life which is closely related to personal data. As per General Data Protection Regulation (GDPR) article 4: "personal data means any information relating to an identified or identifiable natural person ('data subject'). An identifiable natural person is one who can be identified, directly or indirectly, in particular, from an identifier that could be a name, an identification number, location data or an online identifier. This identification could also be performed based on other factors related to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person". A person can be directly identified, without any additional information from different sources. If the latter happens then a user is considered to be identified in an indirect manner. (Corte et al., 2020). The identification of the user's location is possible based on a a unique Media Access Control (MAC) address that could be referred to as an identifier according to the GDPR document and is part of a user's device. According to the Wi-Fi fingerprinting method, the user's device measures the RSSI of different access points that are in range. The device scans the area and waits for the Wi-Fi signal to be captured, so that the aforementioned access points can be discovered. (Schauer, 2019).

3.5. Related Research

3.5.1. In_sight: Using Existing Wi-Fi networks to Provide Information on Occupancy and Exploitation of Educational Facilities using at Delft University of Technology

Elaborated by Van Der Spek et al., 2016, The main objective of In_sight was to calculate the occupation of TU Delft campus in order to determine the exploitation of educational facilities using Wi-Fi monitoring methods. The occupation and exploitation parameters were analysed on five different scales or Spatial Levels (SLs) and then visualised through an online dashboard application. The method used in this project was Wi-Fi monitoring, where the Wi-Fi data of the users are stored into a Wi-Fi database table. User's ID stored on the database were being hashed, therefore, no one can directly identify the users. However, there were several limitations on this project. The number of Access Points were insufficient to determine a more accurate location of the users, since the Wi-Fi coverage inside TU Delft was set up only for the optimal communication purpose at that time. Also, when testing on the building, which had multiple floors, the Wi-Fi signals failed to differentiate the users accurately because the Wi-Fi signals

tend to propagate in a vertical direction rather than a horizontal direction. This resulted in overlapping signals and misinterpretation of the analysis on the data stored. Because of these limitations, the occupation of TU Delft campus could not be determined for all SLs. The results show that the maximum SLs in which the users could be identified accurately was limited to SL1, moreover, if the user was on the SL2, they could have been wrongly determined to be on a different floor. For SL3 and SL4, there were insufficient APs at that time.

3.5.2. Rhythm of the Campus

Similar projects, like the one mentioned above, were developed with the focus being to seek what kind of patterns can be recognised by using Wi-Fi monitoring methods on the area of Library and Aula within TU Delft Campus and how reliable these results will be (Van der Ham et al., 2014). This project was executed by using 20 Wi-Fi sensors which were developed by BlueMark Innovations (spin off company of the University of Twente). Instead of acting as access points, these sensors were actively scanning the probe request packets sent by the devices then captured the MAC address of the devices and in order to maintain privacy, it was being hashed directly in the sensor. The sensors are located at various places within TU Delft Campus, and they were continuously active 24 hours. The raw data was reduced to simplify the data processing, then sampled from the original detection and classified to groups which contain all the devices in the vicinity of a specific sensor. After sampling the raw data, it was being filtered to differentiate between static devices (Wi-Fi routers, desktop computers, printers, scanners, etc.) and user's devices. To execute the data validation, a ground truth survey was being held by using an online questionnaire and routing app. To have a full insight of the project, an online dashboard was developed where it has two types of information, the real time "sensor health" and display of collected and analysed data. This project concludes that with enough sensors, Wi-Fi monitoring method is suitable to find the building's occupancy rates and the pattern of building relation can be derived from the data.

3.5.3. Identifying movement patterns from large scale Wi-Fi-based location data

To have a better understanding of human motion behaviour, data collection of people's position and location are needed. This data will give knowledge for numerous activities, including facility management on campus. Developed by Bon et al., 2016, this project's objective was to identify people's movement patterns from the eduroam network of TU Delft by using a 1730 Wi-Fi access point distributed over more than 30 buildings. People were being tracked by using Wi-Fi monitoring method when they activate their Wi-Fi mode and get connected to a certain access point their NET-ID and MAC address were logged in the Wi-Fi log, however, because of the privacy reasons, this data was hashed. The data acquisition process took 2 months and collected data for more than 30.000 students and staff members. The movement patterns that were analysed was between building or outdoor movement patterns, and the second one is between parts inside the building or indoor movement patterns. This project leads to the conclusion that on the outdoor level, people's movement are closely related to the start and end of lecture hours while on the indoor movement was had movement patterns that deviated from the outdoor trend. There are several limitations on this project such as the reliability of the data and the limited Wi-Fi access points for determining outside movement patterns, since as most of the Wi-Fi access points are located inside the building.

3.5.4. Difference between this project and the related research

To summarise, our research is different from the previous projects mentioned above. These projects used Wi-Fi monitoring methods, where privacy and data security can be the main issue, since people can identify the user using the devices, even though the MAC address or ID is being hashed or anonymised. Therefore, when we use Wi-Fi fingerprinting method, the user's devices are able to find their location based on the matching algorithm between RSSI on the user's device and RSSI that has been stored on the server with the help of radio-map. This ensures that the user's privacy is preserved, without requiring additional anonymising processes. Furthermore, tracking and monitoring people's location are the main objective for these projects while our project has the goal to locate where the user is, the occupation of the room of the area and determine if the location is considered safe under COVID-19 guidelines.

4

Methodology

4.1. System Overview

To get users' location inside the building, a system was designed combining ArcGIS Indoors and Online, along with Wi-Fi recording devices and an user side app to tie it all together. These different components all work together to, in tandem, create a live radio map of the building, use fingerprint matching on the user side, and update the occupancy of an indoor model which can be visualised for both users and TU Delft Campus Real Estate. In this section, all the components of the methodology will be discussed, with an overview available in figure 4.2. First, in section 4.2, the creation and publication of the indoor model will be discussed. Next, the generation of the live radio maps will be discussed in section 4.3. Thirdly, in section 4.4, the data hosting with ArcGIS Online will be covered, in section 4.5, the procedure for the fingerprint matching will be explained, and lastly the user app will be presented in section 4.6.

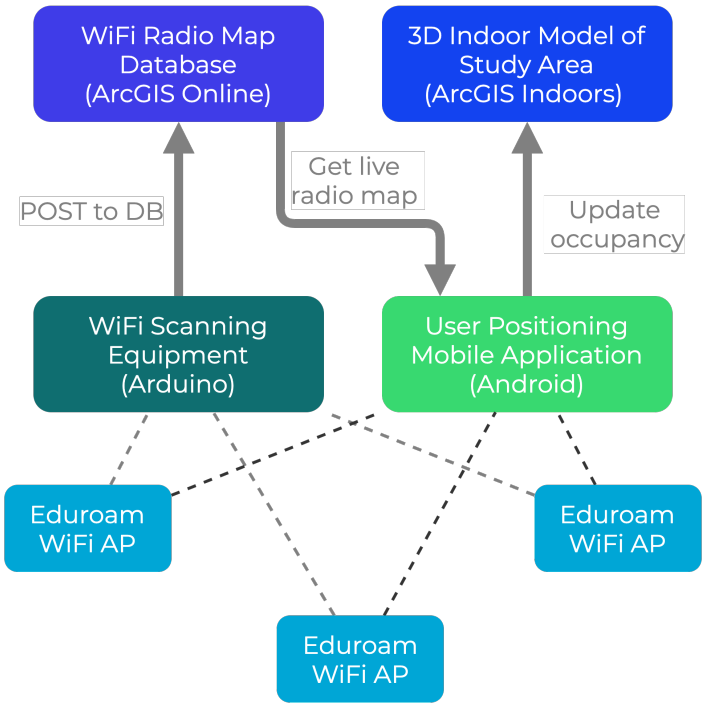

Figure 4.1: Overview of all components of the system

4.2. Indoor model: ArcGIS Indoors

The Indoor model was generated based on different steps that will be thoroughly explained in this chapter. CAD files provided to the team by TU Delft Real Estate, were used as data. Initial manipulation was implemented, as the CAD files needed to be at the exact shape in order to be valid for use in ArcGIS Indoors. Moreover, a vital part towards the creation of the indoor model, was to check that the floor plans were up to date. More specifically, some rooms were incomplete due to the absence of doors, or some other rooms were changed, as the floor plans were created a few years ago. Last but not least, the team decided to make an indoor model, based only on a part of the Faculty of Architecture of TU Delft. Specifically, the East side of the Faculty, as well as the ground and first floor were used, as they were enough to implement the scope of this project. This decision, was also useful in terms of time management.

4.2.1. Indoor Model and Database

After the initial manipulation of the data, the floor-plans were loaded in ArcGIS pro. This operation was accomplished by loading the CAD files, accompanied by an excel file that describes the details of those floor plans. The users can find this excel file on their computer, as it is installed within ArcGIS pro, with an ArcGIS Indoors additional license. Characteristically, this excel file provides information for the facilities, levels and rooms of the floor plans, as well as specific details, as the walls, doors and stairs of the Architecture Faculty. The user has also the option, to add sections or zones of a facility, but this information is not mandatory. It is significant that the CAD layers are distinguishable, in order to express the above characteristics of the model, in a way that will be recognised by ArcGIS pro.

The CAD files of the 2 floors were loaded in ArcGIS Pro. Their coordinate system was set, with the ”Set projection” tool, and they were also georeferenced, based on the ”Georeference” tool of the platform. The world imagery background map was used for the georeference, and by rotating, scaling and using control points, the CAD files were assigned to their correct position. These steps had to be done, since the CAD drawings did not contain this information.

The ”Create File Geodatabase” tool was firstly used to create an empty database. Afterwards, the ”Create Indoor Database” tool was executed, keeping the existing geodatabase as target. After implementing the latter, three new indoor layers are created in the geodatabase file. Those files are named ”Indoors”, ”Network” and ”PrelimNetwork” and include GIS layers, such as the facilities, units, details, network and others, that will be populated with information in the next stages of this procedure.

Furthermore, the ”Import Floorplans To Indoors Geodatabase” tool was implemented, by establishing the following parameters: The existing geodatabase was set as target, the excel file with the floor-plan information and other information as the area unit of measure were added. After this operation, the Facilities, Levels, Units and Details files of the indoor geodatabase were filled based on the information of the excel files. At this point, the CAD layers were transformed into GIS layers, so that the capabilities of ArcGIS Indoors could be exploited. The user can visualise those layers by adding them in the platform. The Units layer that contains information about each room of the case study area, was populated with some additional attributes that were provided by Abdullah Allatas (Alattas et al., 2018), as well as some information about the rooms’ capacity and occupancy, provided by TU Delft Real Estate. This was implemented, by joining the existing excel file with the ones containing the additional information, based on the unique identifier of each room.

4.2.2. Additional Indoor Data

The next step after the creation of the initial indoor model and the corresponding data set, is to create some additional features, in order to enhance the indoor model functionality. For selected attributes of the Units layer, such as offices and lecture rooms, points of interest were created, by using the relevant tool in ArcGIS pro. This operation was created by first selecting specific units , with the ”Select Layer By Attribute” tool and then by executing three different ArcGIS Pro tools. The ”Feature To Point Geoprocessing” tool was used, so that the selected units will be converted into points in a temporary class. Then, the ”Feature To 3D By Attribute” geoprocessing tool, based on the temporary layer previously created and setting the Relative Elevation value in the Height Field text box. The final tool that was used for the creation of the points of interest was the ”Append” tool, which adds the temporary 3D Points Of Interest feature class to the Points Of Interest feature class of the database. The previously created temporary layer is used as input, the target is the Points Of Interest feature Class and finally

the option ”Use the Field Map to reconcile schema differences” in the Schema Type box was chosen. That way, the user can populate additional fields to the Points of Interest feature class, that do not exist in the Units attribute table. Those points of interest will become the nodes of the indoor network that will be explained in the next steps.

4.2.3. Indoor Routable Network

A significant part of the indoor model, is the establishment of its network, so that the connectivity between rooms of the facility is clear. First, the preliminary pathways were generated, in each facility level. This is achieved by using the ”Generate Indoor Pathways” tool, where the user is capable of defining boundaries, such as walls or windows that are part of the details layer, which the generated pathways will not traverse. The user can check the validity of the outcome, by making sure that each room is connected to the rest of the network, through an opening or a door.

Furthermore, in case the facility consists of more than one levels, the floor transitions which can be stairs, elevators or escalators should be generated, so that different floors are connected. In this project, the ground and first floor are connected through a stairway. The ”Generate floor transitions” tool creates network features that connect levels, using vertical 3D lines. As a result the end product of this tool was a vertical line connecting the two floors at the location of the stairway. The user can manually edit the line to create a poly line that will resemble a stairway.

Afterwards, the preliminary network that was previously created, was thinned. Routes can be calculated between selected routable location points or polygons. In this case, the created points of interest, were used, consisting the nodes, of the thinned indoor model that was produced.

In case the Indoor Model consists of different facilities, it would be wise to additionally run the ”Generate Facility Entryways” tool, which sets an entrance for each facility, so that the routable network can connect different facilities through those entrances.

4.2.4. Visualisation enhancement

The symbology of the layers are a vital part that leads to an optimal visualisation of the product. As an example, different colours and textures can be used in the Units layer, where each room can be distinguished, based on his unique identifier, use type, elevation and others. The user has different capabilities to improve the visualisation of the model. First of all, in case there are different floors, the user can set a range, based on the Vertical Order value of the floors. This value which in this project was 0 for ground and 1 for the first floor, can be assigned through the Units layer properties. In this manner a slider will appear on the map, where the user can traverse between the two floors. Last but not least, the user can navigate to the Appearance menu and choose extrusion based on different height options. In this way the user, can extrude walls,windows, doors or other preferable information, in order to give a third dimension to the model.

4.2.5. Publish Indoor Model

After completing the necessary operations towards the creation of the indoor model, it can be published into ArcGIS Online Portal.The user has different options, depending on the use of the model. Specifically, the model can be exported as a web map, a web scene or a mobile package, that can be used in order to create a mobile application. In this project, the indoor model with all of the created indoor layers, was published into ArcGIS Online Portal, as a Hosted Feature Layer. This layer, was used as a basis for the application, which is the end product of this project. In order to publish the model, a name and a summary of the model have to be written, as well as hashtags that are associated with the model and can help in case someone needs to search this project in the online portal. An important step, is to add the ArcGIS Indoors hashtag before uploading, as it will unlock some online indoor capabilities, as the Space Planner and the Indoor Viewer. However, we will go into the server side in more detail in section 4.4.

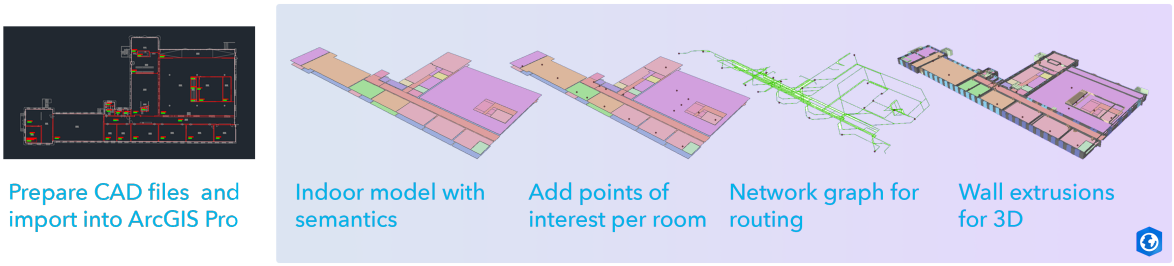

Figure 4.2: Steps towards creating the indoor model

4.2.6. Issues and Limitations

The procedure of creating the indoor model in ArcGIS was considerably smooth and straightforwards, however some issues could not be absent.Data concerning some rooms were absent or invalid, especially concerning the capacity and occupancy. Those values were approximately filled based on other rooms with similar size, for which the data was complete.Furthermore, as it was previously mentioned CAD files were used as input, however, some parts of the floor-plans were not up-to-date. As a result, some manual changes had to be implemented in the drawing, so that the current situation of the Faculty is captured. Last but not least, BIM data could be used as input, which could probably be a better solution in terms of appearance and information. However, this was not possible, as BIM data was provided to the team after a significant time period has passed, so it was not used due to time constraints.

4.3. Live radio maps: Arduino

Generating up to date radio map is an essential part of obtaining accurate positioning information, due to the fact that many factors, like humans physically distorting the radio signals, can change a radio map, and thus introduce inaccuracies and lead to incorrect positioning trough Wi-Fi. To obtain this live radio map, Wi-Fi scanners will have to be present in the rooms that the positioning system needs to cover, with a higher density of scanners, leading to a higher density radio map, leading to a higher level of accuracy. In this project, a density level of 1 scanner per room was chosen.

4.3.1. Hardware & Firmware

For the scanning itself, six Arduino MKR Wi-Fi 1010 micro-controllers were chosen, due to their integrated `u-blox NINA-W102` Wi-Fi radio chip. The Arduino micro-controllers are often used for prototyping, with this specific Arduino micro-controller being suited for IoT applications such as this project.

To power the Arduinos, six Li-Ion battery backs were ordered, with one for each Arduino, so the Arduinos can be placed inside the study area autonomously and keep collecting data even when the study area is not accessible for the public. An schematic overview of the circuit diagram is available in figure 4.3.

Due to the memory limitations of the Arduinos, certain edits to the Nina firmware had to be made, to ensure that enough data could be collected at once. Specifically, the Nina firmware file `~/arduino/libraries/WiFi/src/WiFi.h` needed to be edited, with the following line being changed, with the parameter increasing from 10 to 64. With the line of code displayed below.

```
1 #define MAX_SCAN_RESULTS 64
```

Lastly, SSL certificates for services3.arcgis.com needed to be loaded on to the Arduinos to enable connection through the HTTPS protocol.

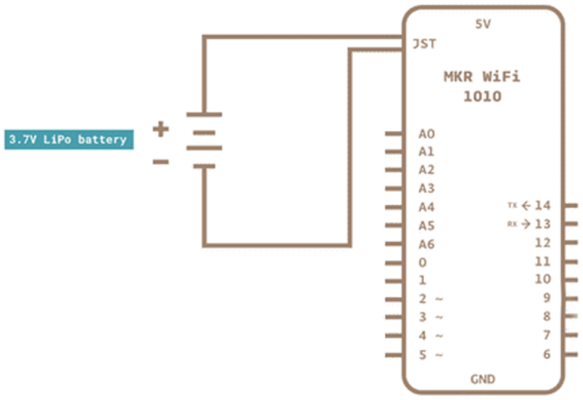

Figure 4.3: A circuit diagram of the hardware used for Wi-Fi scanning

4.3.2. Software

Arduinos are to be coded with C++ programs. Arduino C++ programs need to have the following structure:

```
void setup() {
  // put the setup code here, to run once:

}
void loop() {
  // put the main code here, to run repeatedly:
}
```

While Arduino does not support the C++ Standard Library, it does have libraries of its own, and the capability to extend with 3rd party (open-source) libraries. The full Arduino code is available in appendix B.

For this project, the software needed to scan for networks, store this scan, send this scan to the server, and repeat.

4.3.3. Scanning for networks

This was done using the `WiFiNINA` library, which is the standard library for this type of Arduino. This library has a function `WiFi.scanNetworks()`, which returns an integer over which can be looped to access the specifics of the scanned networks. In this for-loop, the SSIDs of the networks will be checked, and only if the SSIDs are equal to one of the four pre-defined accepted static public networks (i.e. eduroam, tudelft-dastud, Delft Free Wi-Fi or TUvisitor), will the network proceed to storage.

4.3.4. Storing and preparing scan data

After a scan is completed, the scan data needs to be prepared for sending it to the server. The ArcGIS Online REST API requires data to be in a list of JSON objects, thus, the data needs to be transformed as such. To do this, the ArduinoJson library is used. The data is stored as a JSON with the following structure:

```
features=[
    {"attributes": {
            "Time_Stamp": "1623246089",
            "MAC": "84:3D:C6:C7:EF:20",
            "BSSID": "eduroam",
            "RSSI": -81,
            "Room_ID": "08.02.00.560"
        }
    },
    ...
```

```
11 ]
```

The list of features can hold at maximum 32 entries, due to memory efficiency, even though the firmware has been altered to find 64 networks per scan, there is not enough memory to create a list of that size. This list of features is then converted to type string and used as input to post the data to the ArcGIS online server.

4.3.5. Posting to server

To get the data from the Arduinos to the ArcGIS Online server, an HTTPS POST request needs to be made through the ArcGIS REST API. This API has an operational endpoint for adding features to a table, namely `AddFeatures`.

Linking this up to the dedicated feature server on the ArcGIS server leads to a well known endpoint with the following base URL:

```
services3.arcgis.com/jR9a3QtlDyTstZiO/ArcGIS/rest/services/Arduino_Table/
   FeatureServer/0/addFeatures
```

This URL is then used to instantiate an HTTPS POST request, with the list described in section 4.3.4 as data, and `Content-type: "application/x-www-form-urlencoded"`. If the data is structured exactly as the table prescribes, with the appropriate field names, the REST API appends the list to the table and assigns all new features with a unique ObjectID. The full schema of the table is found in appendix E.

4.3.6. Limitations and possible solutions

While the Arduinos provide flexibility and high control, there are some limitations which have presented during the scanning part of the project, which will be presented in this section, with a possible solution.

Hardware

As stated, the Arduinos are a versatile tool, however they do have some limitations, concerning memory and the `u-blox NINA-W102` Wi-Fi radio chip. The memory limitations make it so that one cannot store more than 248kb on the micro-controller, and with the OS, the libraries etc. this does not leave much room for variables, especially String variables, which cost much more in terms of memory than `integer` or `char`. Thus, great care had to be taken to keep memory consumption as low as possible, which was achieved by finding a sensible balance between the number of networks that could be stored, and the update frequency of the table. This balance was set to 32 scan results per iteration, with a pause of 1 second between cycles.

Software

The WifiNINA library is the dedicated library that has to be used when connecting the Arduino MKR Wi-Fi 1010 to the internet. However, this library has some limits, be it because of limitations with the `u-blox NINA-W102` Wi-Fi radio chip or because of other *under the hood* limitations, like that it is not possible for the Arduino to operate a HTTPS Client at the same time as scanning for networks. This is a severe limitation, which cause problems, because the HTTP Client is used to post the data to the server. This led to the current set up, where:

- the Arduino scans for networks
- the Arduino connects to Wi-Fi
- the Arduino creates a post request to send the data to server
- the Arduino disconnects from Wi-Fi
- the Arduino waits 1 second, and then repeats

Eduroam

The setup as presented in the section above however, leads to the the members of the team being banned from the eduroam network. Because, while testing at home, there were no issues, but when testing at Bouwkunde and using the eduroam network to send the data, the DDoS protection of the network kicks in and blocks the Arduino from reconnecting to the network. An increase in the waiting (to a maximum of 5 minutes) was tested, but was not deemed feasibly because of the low data density. Thus, we had to connect the Arduinos to our own smartphones mobile data hot-spots to send the data, which limits the Arduinos flexibility, because we cannot leave them overnight to keep collecting data, plus while testing we (or at least our mobile phones) have to remain in the vicinity of the Arduinos.

4.4. Server side: ArcGIS Online

Early on in the process it was decided that because of ESRI Netherlands' involvement in the process and the data hosting capabilities of the ArcGIS Online platform, ArcGIS Online would be used as the main data hosting space for the project. All components would be connected with each other through this online portal, and all data will be available through the ArcGIS REST API as public data. This decision was made both due to practical reasons, because it removes the need for authentication on the both the Arduinos and the user application that will be developed, but also because it enables everyone to see our collected data. Because only the MAC addresses of the four static Wi-Fi networks present in the building are collected (see section 4.3.4, there is no risk for personal data to be exposed.

4.4.1. REST API

The ArcGIS REST API is a standardisation of HTTPS requests linked to a specific Feature Server, with predefined methods to add, update, query and delete features in that table. Using this as basis, POST and GET requests can be instantiated to access, add to or edit the data from a remote device over the internet, which is exactly what is required. Two `FeatureLayers` were created: one to host the live radio maps, and one to host the indoor model.

4.4.2. Indoor Model Feature Layer

While the creation of the Indoor Model itself has been extensively covered in section 4.2, the entity that is being published on ArcGIS Online is, in fact, a hosted Feature Layer, with 6 tables. One of the layers, namely `Units_new`, a spatial feature layer consisting of the geometry and lots of attributes per feature (every feature is a unit, i.e. room, or hallway), is being used to store and update occupancy data, specifically in the field `OCCUPANCY`. This layer has also been time-enable in ArcGIS online, so ArcGIS Online keeps track of edits, so we can use this data later on the get the rhythm of the building.

4.4.3. Arduino Feature Table

The Arduino table is a much more straightforward hosted table, which is similar to a regular database. There are five fields, identical to the attributes as displayed in the data JSON object in section 4.3.4, namely: `MAC`, `RSSI`, `BSSID`, `Room_ID`, `Time_Stamp`, and a generated primary key `ObjectID`.

4.5. Fingerprint matching: kNN Algorithm

4.5.1. Pilot study using laptop Wi-Fi receivers

To implement indoor positioning using Wi-Fi fingerprinting, we first needed to test if any of the algorithms would actually work for positioning in our context. For this purpose, the team decided to reuse a kNN based algorithm which was devised in an assignment for location and position awareness course/-GEO1003. This data-set was outdated and the algorithm was in a crude phase of development. To build over this the team had to collect data specific to the nodes that we had decided to work on in the building. A pilot study was done in which we collected 15-minute Wi-Fi fingerprint data at various nodes identified within the building using `Intel Dual Band Wireless-AC 3165` chip-sets which are available in windows based computers (laptops). Furthermore, 30 second test data-sets were also collected around these nodes for purpose of testing the algorithm.

Results from pilot study
The pilot study resulted in good results using a kNN based algorithm coded in Python. We were able to predict the position of the 30 second datasets using the trained model correctly. As the project progressed, we realised the need to make an app for locating the user in real-time, and also to gather the users 30 second test data-set. For this purpose we had the options of making an app using Java for Android or using C++ based Swift for iOS devices, however, IOS does not allow apps to access the Wi-Fi chipset on the device and thus would not have worked for our use case, thus the team was left with the option to make an app on Android.

4.5.2. Shift to Java

Coding the app on java which would take a users Wi-Fi scan and collect data from a live table hosted on ArcGIS online server and then would run a matching algorithm using kNN came with its own challenges. Most kNN implementation on java were done by users by hard coding a method to calculate the distances, and do a majority voting on the calculated distances. These tasks in case of the pilot study were being handled by the `sci-kit learn and scipy` libraries on Python. The team decided to make our own implementation of kNN on java after trying to use tensorflow lite for training the model for some time. Tensorflow models could be downloaded from a server after training the model on a ArcGIS notebook which would run every few minutes and update the model for the latest fingerprints. However, this required for us to configure a webserver which would host the ArcGIS Notebook server and the notebook itself to be run using a scheduler, this would have meant that we needed to have a server up and running and would have cost the project a computer device. Thus it was decided to implement our own version of kNN on Java.

4.5.3. kNN on Java

To implement kNN on java we needed to find a way to load the JSON data-set from both the users device and the table storing the live fingerprint data.This was done by using Googles' `gson` library on Java. Classes had to be made to store the JSON objects and then to parse the JSON object to make `kNN_methods` objects which would store the MAC ID, RSSI values and the label of the room from which that data is from.

Furthermore a kNN method was to be implemented which would calculate the distances, find the minimum distances and then conduct a majority vote on the labelled data-set to find the final prediction.

The pseudo-codes 1 and 2 elucidates the process of implementing the Euclidean distance and finding the k minimum distances in the Java code.

Algorithm 1: Calculate distance of (x1, x2) from (tx1, tx2)

Function `getEuclideanDistance`$(training\_object\_list, test\_object)$**:**
  $tx_1, tx_2 \leftarrow test\_object.x_1, test\_object.x_2$;
  $sum \leftarrow 0$;
  foreach x_1, x_2 *in* $training\_object\_list$ **do**
    $sum = sum + (tx_1 - x_1)^2 + (tx_2 - x_2)^2$;
    $distance = \sqrt{sum}$;
  return $distance$

However, it must be noted that the Java implementation was a very rudimentary form of kNN as compared to the python implementation, which was being handled by libraries made specifically for machine learning purposes.

4.5.4. Porting Python to Java and current solution

After implementing the kNN algorithm on Java we tried testing the code with the live fingerprint data and the live user fed data from the android device, however, the results were dishearteningly bad, we were unable to correctly locate the rooms that the users were in, thus it was decided that we re write the java kNN implementation and fix the bug in the algorithm, however due to a paucity of time, it was later decided that since finding a bug would cost the project more time than we had at hand, it would be wiser to find a way to port the python code which worked will even with live fingerprints and users android

Algorithm 2: Find minimum distance in distance list

```
1  Function getMinDistances(object_list, distanceList, k):
2      min[k] , indices[k] = new Array;
3      min[k] ⟵ populate with maximum possible integer value;
4      labelList[] ⟵ new Array;
5      for i=0 →objList.size() do
6          distance = distanceList[i] max ⟵ populate with minimum possible value;
7          maxIdx = 0;
8          for j=0 →k do
9              if max < min[j] then
10                 max = min[j];
11                 maxIdx = j;
12         if min[maxIdx] > distance then
13             min[maxIdx] = distance;
14             indices[maxIdx] = i;
15     foreach index in indices[k] do
16         labelList ⟵ place value of label of object at given index;
17     return labelList[];
```

data, to Android. This was accomplished by using a Python SDK for Android, known as Chaquopy, it allows for python scripts to be imported and used in a Java or Kotlin based Android apps, and has support for libraries such as `SciPy, SciKit Learn, Numpy and Pandas`. To run this, necessary changes were made in the dependencies of the app and the python script to make it compatible to Chaquopy. The following subsections explain in detail how the present approach to kNN works.

Creation of a training model
The ArcGIS Online REST API was used to load the table which contains the updated Wi-Fi fingerprints from the Arduino sensors located at different nodes, being hosted on the ArcGIS Online server. This is done using the `JSON` and `URLLib3` libraries on Python. The HTTP request returns a JSON string which is then parsed to be loaded into a Pandas DataFrame table, with the columns for MAC address, RSSI values, and Room_ID.

Preparing the test data-set
The Wi-Fi signals received from the Android device are parsed as a JSON string, and loaded directly to a Pandas DataFrame. This DataFrame is then cleaned to contain two columns, one for the MAC addresses and one for the signal.

Matching algorithm on kNN
Once our test DataFrame and the training DataFrame are prepared, the x and y variables for the kNN algorithm is extracted. This is done by the following means: Firstly, in order to take the MAC addresses into account when running a matching algorithm, all the MAC IDs are collected and each unique MAC address is assigned a unique number. The x-variable for a given training data-set is the set of unique MAC numbers and their corresponding RSSI values. The y-variable is the label containing the rooms' name from which a particular MAC number and a RSSI was obtained. For the test data-set, only the x-variables need to be created, which should be in the same format as the training data-set, so a unique MAC identifier and RSSI values are needed. To prepare the x-variables for the test data, the MAC addresses are matched to the fingerprint MAC addresses and the unique number assigned to a MAC in fingerprinting is given to matching MAC addresses from the test data-set. If a MAC in test data cannot be found in fingerprint data, it is discarded along with its RSSI values, as that particular MAC address would not help us in matching. The y-variable in the case of the test data-set is supposed to be our final label for the room to locate the user.

The kNN algorithm uses Euclidean distances to compute the prediction, additionally after some trial and error, the best k value was chosen as 7 with a weight being added to the distances computed.

Pseudo-code at 3 explains the general working of the algorithm, and the full Python code is available in appendix C.

Algorithm 3: Pseudo-code explaining the general working of the python script running the matching algorithm

input : A JSON string $test\_json\_string$ gathered from users device Wi-Fi scan
output: A label for the predicted room the user is in

Function `python_knn_function(`$test\_json\_string$`)`**:**
    import pandas, scipy, sklearn, numpy, urllib3, json;
    url ← URL path for HTTP get request to ArcGIS online table storing live fingerprint;
    json_table ← read url with urllib3, decode as a dictionary using json;
    training_df←Pandas DataFrame made using the json_table;
    foreach *MAC_ID in training_df* **do**
        `// assign a unique number to unique MAC addresses`
        uinque_list ← dictionary of MAC Address and unique number assigned;
        assign the unique mac number to all available MAC address in training_df;
    x_train ← mac number and RSSI corresponding values in training_df ;
    y_train ← room labels for corresponding x_train in training_df ;
    test_df←Pandas dataframe made form test_json_string;
    test_df⟵test_df with MAC addresses assigned the same number from unique_list
    x_test←mac number and RSSI from test_df;
    `// run the kNN training and prediction`
    use sklearn kNN training model using k=7;
    labels_list←run the sklearn predict() function to obtain list of labels;
    final_label←mode of labels from labels_list using scipy.stats;
    return *final_label*

4.5.5. Limitations and possible solutions

Present approach of using kNN on python algorithm and porting it to Java has accurate results but also has huge scope being erroneous if a good number of matching MAC addresses of the access points are not found. This can be fixed by training a deep learning model on a data-set which is gathered over the period of a few weeks. Such a trained model would take into account the general fluctuations that occur in the Wi-Fi RSSI values as people come and go in the building, however our implementation using live fingerprints from Arduino Wi-Fi sensors serves the same purpose of capturing the fluctuations in RSSI values as they occur in real time. Apart from that, kNN could be adapted to be adaptive in nature such that the best-suited k value is chosen every time a user needs to be located.

4.6. User side: Android User Application

As stated in section 4.5.1, access to the Wi-Fi chip was paramount for the development of the user app, and with Apple not opening this up for developers in iOS, this led us to start the development of an Android user app. However, due to the fact that Android development is something that was not in either of our skill-sets, we did make the app as basic as we could, while remaining in scope and fulfilling our project goals. In this section we will outline the basic functionality and user interface of the app, as well as the back end. The entire app is coded in Java, and relies heavily on the ArcGIS Runtime API for Android, as well as the main Android predefined structure (`MainActivity` etc.).

4.6.1. User Interface: 3D Scene

The home screen of the user app is a 3D Scene of the Indoor Model of the study area which is directly added from ArcGIS Online. The main functionality of the app is to provide a visual representation of the real time occupancy of the study area, and to allow the user to locate themselves in this study area (given that they are in a room that is covered by the system).

On startup, the app shows the scene, and a button to locate themselves. Due to the fact that this

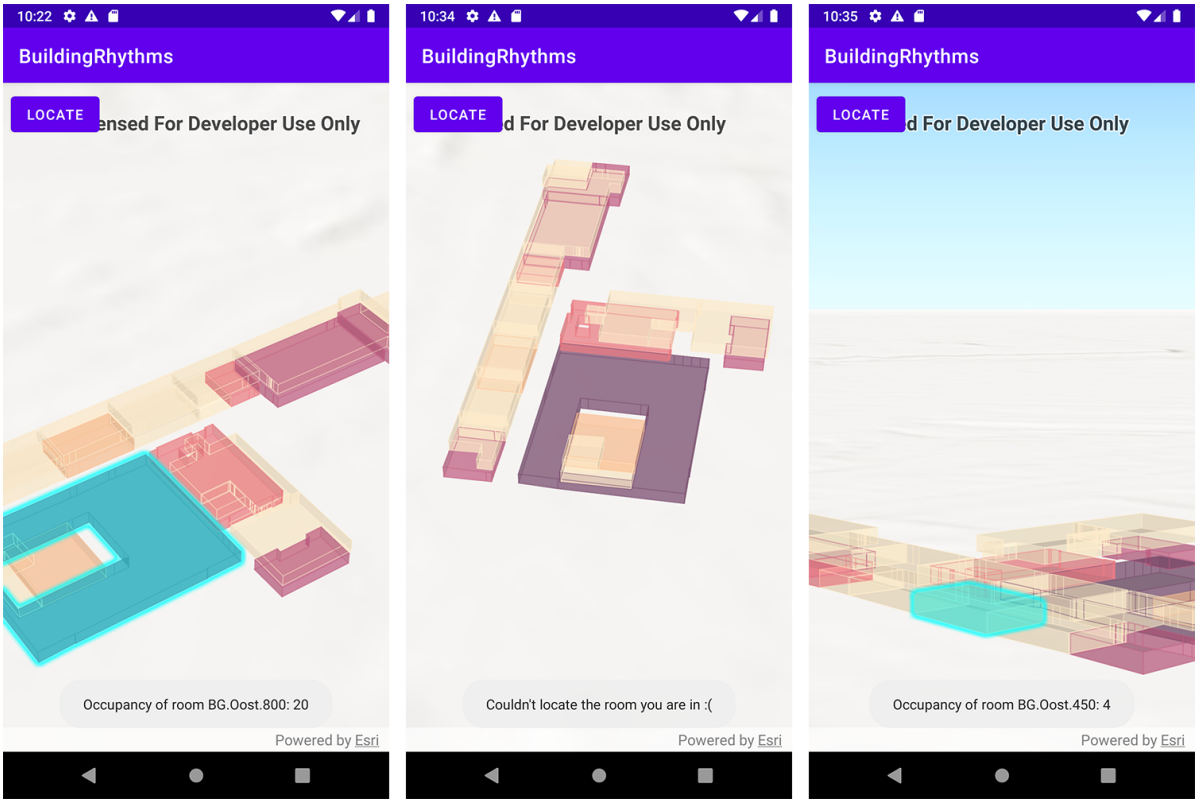

Figure 4.4: The main screen of the app, with the Oost Serre Room selected

app was designed with an *opt-in* philosophy, the user has to press this button for the locating algorithm (as described in section 4.5.4) to start, instead of the location algorithm to start on launch. For the purpose of displaying the model itself, the ArcGIS Runtime API for Android is used to instantiate a scene layer, namely the `Units_new` layer from the indoor model, and extrude this layer according to the elevation and height parameter as specified in the model. A class based colouring function is used to give the layers colours according to the occupancy values. The assigned colour gradient is visible in figure 4.5. Currently the colour groups are solely based on the occupancy values themselves, but ideally the colour groups would be based on a ratio of

$$\frac{occupancy_{current}}{capacity_{covid}}$$

However, this was not possible to implement at the moment.

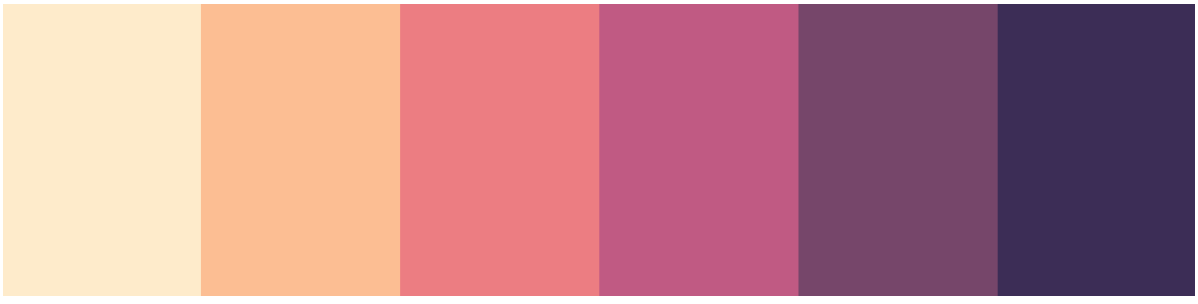

Figure 4.5: The colour scheme for the occupancy values, with from left to right: 0, 1-3, 4-7 8-11, 12-15, 16+

Furthermore, apart from providing the user with an intuitive colour based indication of the occupancy of a room, it's also possible to tap on a room and see it's occupancy, with the selected room highlighted.

4.6.2. Linking positioning to occupancy

On the back-end, to actually implement the updating of the occupancy without storing any more data then needs be, a simple updating function was developed. This function uses the output of the matching algorithm, which will either be a string with the `NAME` attribute of a feature, or it will output the error string (`Couldn't locate the room you are in :(`). If the output is the error string, nothing happens, as no new occupancy data has not been generated. If the output of the matching algorithm is not the error string, the updating function will be called once or twice, depending on if there is a previous room known to the app or not. If the previous room is known, the app will decrement the occupancy in the previous room, and increment the occupancy in the current room. The entire logic is visible in the pseudocode below.

```
onCreate() {
    previousRoom = null;
    currentRoom = null;
    clickLocate() {
        locationResult = matchingResult();
        previousRoom = currentRoom;
        currentRoom = locationResult;
        if(previousRoom != null){
            updateOccupancy(previousRoom, -1);
        }
        updateOccupancy(currentRoom, +1);

    }
}
```

The actual `updateOccupancy()` function, as it is called above, uses the ArcGIS REST API to GET the current update of a room, and POST an edit to this feature, either incrementing or decrementing the occupancy for that feature, the full function can be seen in appendix D.

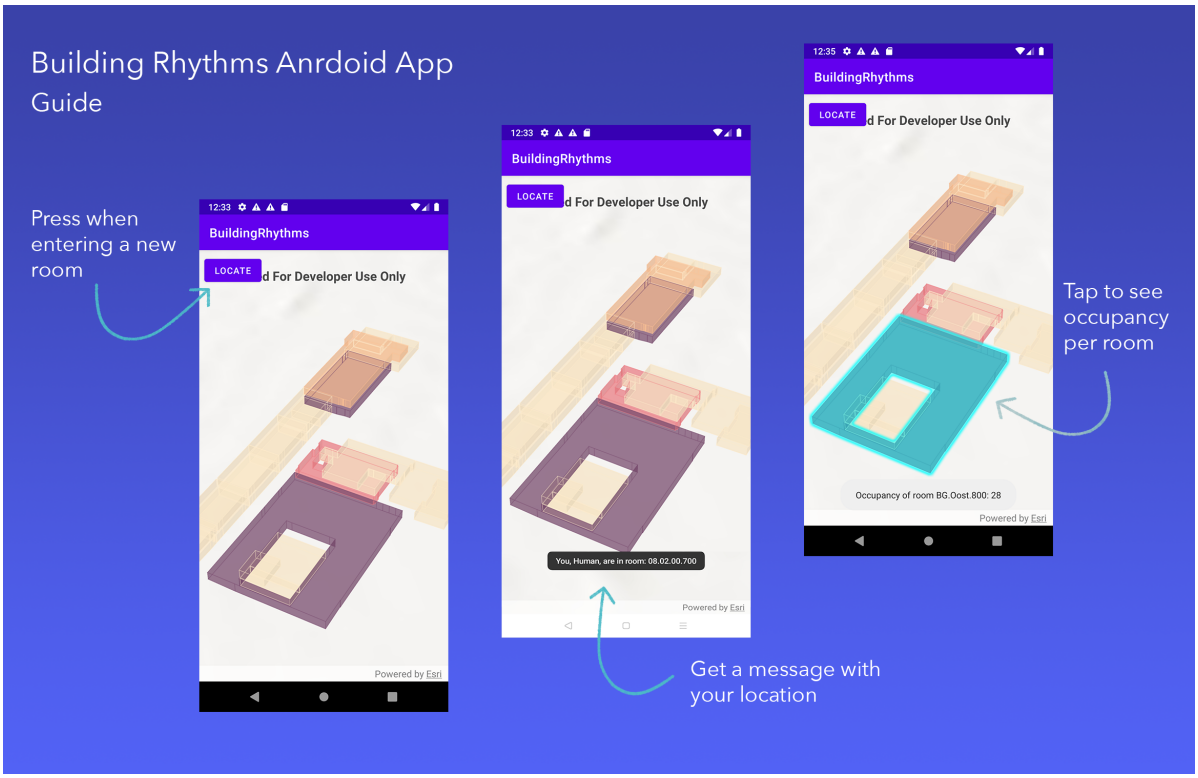

Figure 4.6: A user manual for the developed app

4.6.3. Limitations and Solutions

While we always envisioned a user app as one of the end products of our project, as per the PID, the scope of the app did get trimmed down somewhat. For instance, we were not able implement routing, or significantly expand the user interface, or implement sound logic for entering and exiting the study area. In this section, these limitations will be discussed.

General bugs

Despite great care being taken to make the app run as smoothly as possible, it is not finished and is still a work in progress. One thing that is still rather buggy is the Wi-Fi scanning protocol. Per Android APK level 28, Google has implemented more restrictions on the Wi-Fi Manager protocols, leading to developers having less direct control of the output of the scan results. This, coupled with the fact that the Android operating system can be quite complex, lead to quite some crashes when testing started due to the list of scan results growing exponentially. From the GEO1003 course, we were aiming for scanning for about 30 seconds, and using that as input for the matching algorithm, but it was not possible to implement this without significant (crashes, hangups etc.) bugs.
Another general bug is that when the app start, sometimes it renders the scene upside down for a short period of time.

Routing

On of the goals stated in the PID was to also provide a routing option, which we were not able to implement in the app due to time constraints, even though the back end (that is, the Indoor Model) is already set up to enable this. The way to implement this would be to have a separate button for routing, which can be tapped whenever a user has a room selected on the map, and the cheapest (in terms of occupancy) route will be calculated using the connectivity model stated in section 4.2.
This would however, require further deployment of the system, namely, all the rooms, units, hallways etc. need an Arduino point to locate to, otherwise there will be significant gaps in the route.

UI Limits

While the 3D is somewhat clear at the moment, we would have liked to spend more time making the 3D even more intuitive, perhaps by creating a layer selector element with which individual floors could be selected. This would enhance the visualisation significantly.
Another element that has been omitted in the visibility is the building details that *are* included in the Indoor Model (i.e. walls, doors, windows etc.). These were omitted for both visualisation reasons as for performance reasons. It would be nice to fully flesh out these details and for instance display them only at a certain zoom level to prevent the app from using too much memory.

Entering/leaving the study area

While the app is able to update the occupancy for a user inside the study area, the testing set up is currently rather small scale, and thus no logic has been implemented to handle when users leave the study area and stop actively locating. A possible implementation would be to use either a *sign in/sign out* method, where the user is required to sign in (their first locate) upon entry, and sign out (just decrement their previous room) upon leaving the building, or use a more sophisticated method like geofencing, where the app will only be active when the user is within a certain radius of the study area, and if the user has not been locate within the study area for an arbitrary but sensible amount of time, the app will automatically decrement the users last room.
While the former option is less sophisticated and less user friendly (in an obvious way), it does not require any extra data like GPS, which could be a benefit.

4.7. User side: ArcGIS Dashboard

As it was previously mentioned, one of the main goals of our project is to eventually investigate and analyse the impact of COVID-19 in the indoor environment. In order to do that, the manager of a faculty or even the whole campus should be able to have access into statistics regarding the movement patterns in the buildings, and information such as the peak hours when rooms are full. This information could aid towards designing an optimal space planning, so that overpopulation of rooms could be

avoided. To accomplish that, a dashboard was created, through the ArcGIS Online Portal. The dashboard offers various ways to express hourly, daily, or even monthly statistics, while its graphs can be updated real-time with a small refresh rate.

The dashboard consists of the indoor model, a list of the rooms per floor, a series chart, a pie chart and a gauge. The user, has several capabilities concerning the map, such as enabling or disabling layers, as well as changing the basemap type. Moreover, when a user chooses a floor, the corresponding rooms are visualised on the map and the rooms' list is updated. When a room is chosen from the list, it is automatically selected on the map, while also room statistics appear in the charts. A pop-up with room information also appears on the map. A series chart was also created in order to express, the monthly occupancy per room. Furthermore, a gauge that shows the occupancy number per day was created. In case a room is selected only the occupancy of this room is visible. The pie-chart shows, daily statistics, so that questions such as the number of rooms that have daily occupancy of 10 can be answered. In case a part of the pie chart is selected, then the user can see the corresponding rooms on the map. Last but not least, there is a date selection option, which affects the pie chart and the gauge, by showing different results per day.

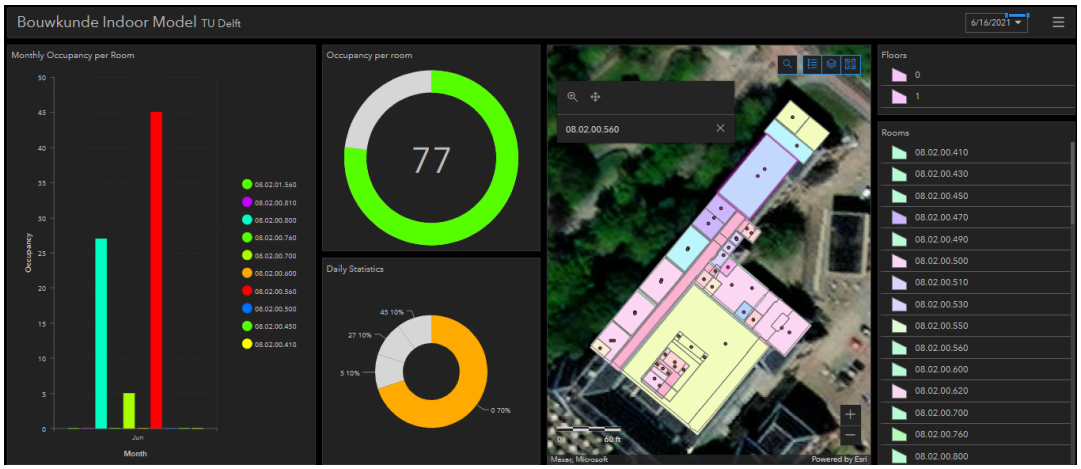

Figure 4.7: Dashboard Preview in ArcGIS Online Portal

4.8. Testing the App

One of the most crucial points of the project is the testing of the app. At that phase everything we were working on theoretically would be able to appear in reality. Therefore, we decided to divide the testing into two parts so we could present wider results regarding the accuracy of the app. These two parts were about four rooms firstly and then about different tables inside one room. The final results of the testing will be presented in the Chapter 5.

4.8.1. Four Room Setup

For that part of the testing we had to place the Arduinos in four different rooms as it is visible in the Figure 4.8. Afterwards, with the use of few Android mobiles which had installed our mobile App we walked around the rooms in order to "Locate" ourselves in the specific rooms and test if the mobile app working in the way that should be working. To demonstrate the level of success and the accuracy of the App we tested different patterns for example being in the center of the room and close to the Arduino but also being in a corner meaning closer to another Arduino location. At the same time the ArcGIS Dashboard was being checked for the real-time updating of the statistics there. The final results and their analysis are presenting in the section 5.1.

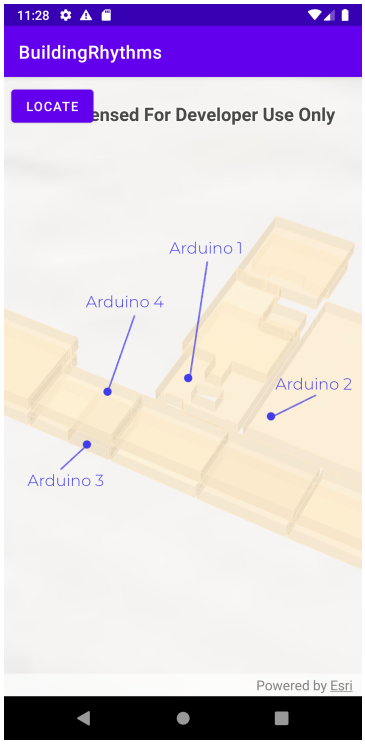

Figure 4.8: The testing setup with four rooms actively being scanned

4.8.2. One Room with Four Tables

To test the versatility and accuracy of the system we decided to also perform a test on one room and test the results regarding four tables close to each other. Therefore the Arduinos were placed on each of the four tables as it is shown in the figure 4.9. Finally the same procedure like in the first part was followed in order to check the functionality of the app and the dashboard as well. This time we tried to ”Locate” into different spots on the tables and check if the app will give the expected result. The final results and further analysis is provided in the section 5.2.

4.8.3. Limitations and solutions

The limitations and issues we faced during the testing was mainly regarding the Arduinos. As already mentioned in the section 4.3.6, since we could not connect them to Eduroam network, we were forced to have them connected all the time in our laptops in order to provide them power and at the same time use our mobile hot-spots for the internet connection they need in order to do the scanning for networks. This of course was not letting us to do the testing for too long period of time since all of us had to stay next to Arduinos. For that reason and because our team consists of six members, we decide to use only four Arduinos while the remaining two members were testing around with the Android mobiles.

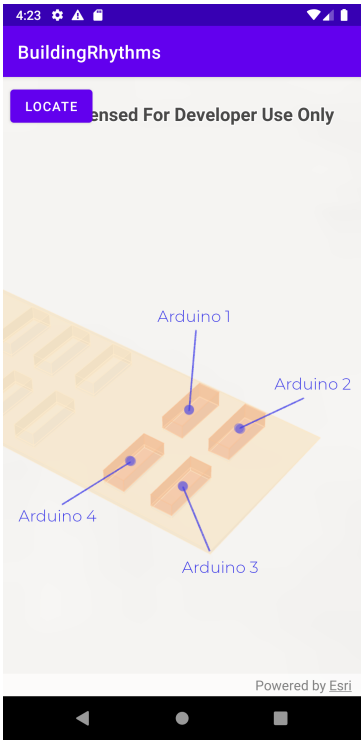

Figure 4.9: The testing setup with four tables inside one room actively being scanned

5

Results & Analysis

In this section we will discuss the results of testing the system and we will analyse these results. First we will present the results of our two testing setups presented in section 4.8 in section 5.1 and 5.2. In sections 5.1.1 and 5.2.1 we will analyse these results.

5.1. Four Room Testing Setup

Room ID	Centre of room accuracy [%]	Corner of room accuracy [%]
BG.Oost.560	100.00	100.00
01.Oost.560	10.00	0.00
BG.Oost.700	30.00	40.00
BG.Oost.800	100.00	100.00

Table 5.1: The accuracy of our indoor localisation system with n=10

Table ID	Number of features	Percentage
BG.Oost.560	5193	48,1
01.Oost.560	743	6,90
BG.Oost.700	2167	20,1
BG.Oost.800	2692	24,9
Total	10804	100

Table 5.2: The amount of features per room at time of testing.

5.1.1. Analysis

After looking at table 5.1 it would seem that the localisation system we designed is not functioning well, because there is low accuracy on at least 2/4 rooms upon first glance. However, when diving into the testing deeper we do see that there are some reasons for this outcome, and then these results are not unexpected.

Room proximity

The first thing to consider is the placement of the rooms, namely the pairs BG.Oost.560 and 01.Oost.560 and BG.Oost.700 and BG.Oost.800. For the .560 rooms we know that these two rooms are directly above each other, which leads to lots of shared access points, and could lead to a less unique fingerprint for each room, leading the fingerprint matching algorithm to wrongfully assign the BG.Oost.560 room when in the 01.Oost.560 room and vice-versa.

In the case of 700 and 800, the fact that often when the user was in 700, the algorithm would match the user to 800 erroneously is also due to their proximity, with room 700 sharing a wall with windows with 800. This could also be the reason that the accuracy for room 700 is higher near the walls of the room, the 40% accuracy would then be the 4 measurements taken nearer to 800 than to the location of the Arduino.

Table ratios
Another thing to consider is the fact that, as visible in table 5.2, the ratio of different rooms varies quite a bit, with an over-representation of room BG.Oost.560, and an under representation of room 01.Oost.560, leading us to theorise that, this is another reason that 01.Oost.560 is has a very low accuracy. To solve this, in the second testing setup we tried to watch the ratio of the Arduinos.

5.2. One Room Testing Setup

Table ID	Accuracy [%]
2.031.O	100.00
2.032.O	100.00
2.033.O	100.00
2.034.O	100.00

Table 5.3: The accuracy of our indoor localisation system within a room with n=3

Table ID	Number of features	Percentage
2.031.O	892	23,8
2.032.O	998	26,7
2.033.O	845	22,6
2.034.O	1008	26,9
Total	3743	100

Table 5.4: The amount of features per table at time of testing.

5.2.1. Analysis
In the above tables we can see that the accuracy for this setup is 100%, although the sample size is lower. This does however validate our theory that the division of features in the table is of large importance for the accuracy of the results of the system and the matching algorithm in particular. As a result, based on both testing methods we can understand that based on the size of the rooms, different number of Arduino devices should be placed, in order to achieve an optimal accuracy.

6

Conclusions

In chapter 2 we defined our research question as:

> *"How can ArcGIS Indoors be combined with indoor Wi-Fi fingerprinting localisation awareness techniques to create a real-time application for understanding COVID-19's impact in the indoor environment of TU Delft with high respect for privacy of the users?"*

In this section we will look at whether our goals have been realised, and to what extent.

When defining the product, it became clear quite, early on, that we would need to design, develop and test a system that would try to answer the need created in the research question, namely: the combination of ArcGIS Indoors with a privacy preserving Wi-Fi fingerprinting positioning technology. To this extent, we have achieved what we wanted, with different levels of success on individual aspects of the system we design, as discussed in chapter 5.

On a more specific level, the system isn't exactly production ready. But we did make a big initial step to show that a reasonably accurate, versatile, cheap and scalable system can be created to provide an answer for the need stated in the research question. In terms of privacy, which was a very important value and key reason the entire system is set up this way, as opposed to using something like Indoors (the native ArcGIS IPS), we have achieved what we wanted. There is no personal data required to display and store occupancy information for an indoor environment like the Faculty of Architecture. Furthermore, we managed to separate the client side and server side in the sense that the user is not required to share any data with the server concerning Wi-Fi or location as of yet.

Furthermore, we have developed a robust pipeline for converting often readily available CAD data into a semantically rich indoor model suitable for indoor positioning on a room level, using ArcGIS Pro/Indoors. This pipeline can be applied to any indoor environment for which this data is available, which is essential for the scalable nature of the system in its entirety.

In a technical sense, we have shown that using a single scanning point inside a room to create live radio maps can lead to a reasonable accuracy level, and deploying this setup can yield similar levels of accuracy as when constructing a very time consuming radio map at regular intervals as in the traditional WiFi fingerprinting setup. With our system having lower initial time investing, as the scanning devices operate independently and autonomously, this project serves as a proof of concept.

However, as previously stated, the system developed in this project is not yet production ready, and many steps could be undertaken to improve it. These possible improvements are discussed in the next section.

7

Future Work

As the conclusions show, this project is an initial effort to use fingerprinting techniques, in order to discover the effects of COVID-19 in University life. However, there are various aspects that could lead to the project improvement, that were not covered mostly because of time limitations.

7.1. Fingerprint Matching Algorithm

First of all, kNN algorithm was used, with its advantages and disadvantages that were described in previous chapters. The results of the final product could be improved, if adaptive kNN was to be used instead of the simple kNN. As its name implies, the adaptive kNN would change the k value on the go and chose the best k possible which gives a prediction with the highest probability, this would have slight chances of increasing the prediction accuracy. Apart from the kNN algorithm, a random-forest algorithm could also be used to predict the room in which the user is present. To further increase the accuracy, we could also use a different non-machine learning based approach such as using algorithms which work with signal propagation, though this would mean that we would need a grid of Wi-Fi sensors laid out across the area of interest where the user needs to be located. Additionally, the project could also benefit from exploring the usage of BLE (Bluetooth Low Energy) in combination with Wi-Fi fingerprinting, so the Bluetooth would have assisted the kNN algorithm to already bring down the possible number of rooms in which the user is located.

7.2. Scale of Project

This project was initiated to cover a part of the Faculty of Architecture. As the results have shown, the project can also be scaled up to be implemented at a larger scale, such as the whole Faculty, or even the entire campus. In order to achieve this, an adequate number of Arduinos should be strategically placed in each room, based on a grid, so that high accuracy even for large rooms could be achieved. Another approach could be the division of large rooms into smaller segments.

7.3. Navigation

This project mostly focuses on rooms' occupancy, which is a measure for expressing the room's population. The capabilities of this project could be significantly expanded by providing the user the chance to navigate in this indoor environment. This could not be implemented, due to a paucity of time, however, this option could highly improve the user experience and help the manager of the indoor space to answer problems, such as finding patterns between specific rooms, at peak hours as well as in determining the pattern of movement or trips within the building.

7.4. App implementation

The project resulted in the team making a native Android app based on Java which locates the user within the building, tells the user of their position, updates the occupancy on the ArcGIS online portal and also tells the user of the occupancy through a map displayed on the user's device. This app however,

was developed by students of Geomatics who don't necessarily have expertise on app development; thus, the app at present, is buggy and crashes at times. The apps implementation in itself has a lot of scope of improvement where the app's interface could be made better, the matching algorithm could be made to run in the background at all times therefore removing the need to have a "Locate" button on screen. The users could also be notified of the occupancy of the place they are in without the need to touch the Locate button. Notifications could also be sent to the user about the change of occupancy of the room to alarming levels or possible rooms they could move into.

7.5. Server-side implementation of Python scripts

At present the app implements the kNN algorithm on a python script which is run using Chaquopy, an SDK for Python on Android, this comes with a demerit that the app becomes too large in size as it needs to load python and required python libraries and its port to java. The projects final app is about half a gigabyte in size and this could be curbed. To do so, the python script could be made to run on a hosted notebook on a web-server which could be accessed by the app through HTTP GET and POST requests, however this would require WiFi fingerprints to be sent server-side, which could have repercussions for privacy.

7.6. Big data from Wi-Fi fingerprints

The idea of the project to place sensors in the building to collect live fingerprint data could be scaled temporally such that all the fingerprinting made is stored for a longer period such as a week, a month or even months. This big-data collected over such a large period of time would enable new research possibilities, an example of which could be to find a pattern in the fluctuation of RSSI values within the building and to visualise how they change seasonally depending on the time of the day and the day of the week. This data-set could also be used to model a deep-learning based fingerprint matching algorithm which could work for more accurate values. These are some cases which are possible even with the present state of the project.

Glossary

AIIM ArcGIS Indoor Information Model. 1, 7

Android A Mobile Operating System. 1

API Application Programming Interface. 1

ArcGIS GIS commercial software. 1, 6, 7, 11, 13

Arduino An Open-Source Brand of Micro Controllers for Prototyping. 1, 6, 14

BIM Building Information Model. 5, 7, 14

BLE Bluetooth Low Energy. 7

Bouwkunde Faculty of Architecture. 1, 17

C++ A Cross-Platform Language that Can be used to Create High-Performance Applications. 1, 15, 18

CAD Computer Aided Design. 5, 7, 12, 14

COVID-19 Corona Virus Disease-19. 10

ECHR European Commission on Human Rights. 9

eduroam International Roaming Service for Education. 17

GDPR General Data Protection Regulation. 9

Geodatabase Georeferenced Database. 12

GIS Geographic Information System. 7

GNSS Global Navigation Satellite System. 5

HTTP Hypetext Transfer Protocol. 16

HTTPS Hypetext Transfer Protocol Secure. 16

iOS An Operating System Used for Mobile Devices Manufactured by Apple Inc. 18

IPS Indoor Positioning System. 5, 8

JSON Javascript Object Notation. 19

kNN k-Nearest Neighbour. 1, 6, 9, 17–19

MAC Media Access Control. 9, 10, 19

python An Object-Oriented Programming Language. 18

REST Representational State Transfer API. 1, 16, 17

RSSI Recceived Signal Strength Indicator. 8, 10, 19

tensorflow A Free And Open-Source Software Library for Machine Learning. 18

WAPs Wireless Access Points. 5

Wi-Fi Wireless Networking Technology. 1, 5–7

Bibliography

Alattas, A. ; Oosterom, P. V., Zlatanova, S. ; Diakité, A. A., & Yan, J. (2018). *Developing a database for the LADM-IndoorGML model* (tech. rep.). APA. https://repository.tudelft.nl/islandora/object/uuid:31a20fb8-dabc-4f19-82c8-432f410a3ece?collection=research

Bahl, P., & Padmanabhan, V. (2000). Radar: An in-building rf-based user location and tracking system. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, *2*, 775–784 vol.2. https://doi.org/10.1109/INFCOM.2000.832252

Bon, M. P., Den, X. A., Dukai, D. B., Griffioen, S. J., Kang, Y., & Vermeer, M. (2016). *Identifying movement patterns from large scale Wifi-based location data A case study of the TU Delft Campus*. TU Delft. Delft. https://repository.tudelft.nl/islandora/object/uuid%5C%3Abd426086-5ccb-4e3e-9472-b6f4af6a9956

Brownlee, J. (2016). *Master machine learning algorithms: Discover how they work and implement them from scratch*. Jason Brownlee. https://books.google.nl/books?id=PCJnAQAACAAJ

Burgess, T. (2019). 6 - radio fingerprinting-based indoor localization: Overcoming practical challenges. In J. Conesa, A. Pérez-Navarro, J. Torres-Sospedra, & R. Montoliu (Eds.), *Geographical and fingerprinting data to create systems for indoor positioning and indoor/outdoor navigation* (pp. 115–127). Academic Press. https://doi.org/10.1016/B978-0-12-813189-3.00008-3

Corte, L., Tilburg University (Tilburg, N., & School, T. L. (2020). *Safeguarding data protection in an open data world: On the idea of balancing open data and data protection in the development of the smart city environment*. Tilburg University. https://books.google.nl/books?id=Y-WnzQEACAAJ

ESRI. (2021a). ArcGIS Indoors | Indoor GIS for smart building management. Retrieved June 15, 2021, from https://www.esri.com/en-us/arcgis/products/arcgis-indoors/overview

ESRI. (2021b). What is ArcGIS Pro? Retrieved June 15, 2021, from https://arcgis.pro/what-is-arcgis-pro/

ESRI. (2021c). What is GIS? | Geographic Information System Mapping Technology. Retrieved June 15, 2021, from https://www.esri.com/en-us/what-is-gis/overview

Jordan, M., & Mitchell, T. (2015). Machine learning: Trends, perspective, and prospects. *Science Magazine*, *349*, 255–260. https://doi.org/10.1126/science.aaa8415

Mautz, R. (2012). Indoor positioning technologies. https://doi.org/10.3929/ethz-a-007313554

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning* (2nd ed.). MIT Press. (Original work published 2012)

Schauer, L. (2019). 2 - wi-fi tracking threatens users' privacy in fingerprinting techniques. In J. Conesa, A. Pérez-Navarro, J. Torres-Sospedra, & R. Montoliu (Eds.), *Geographical and fingerprinting data to create systems for indoor positioning and indoor/outdoor navigation* (pp. 21–43). Academic Press. https://doi.org/10.1016/B978-0-12-813189-3.00002-2

Singh, A., Thakur, N., & Sharma, A. (2016). A review of supervised machine learning algorithms. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 1310–1315.

Torres, J., Belmonte, Ó., Montoliu, R., Trilles, S., & Calia, A. (2016). How feasible is wifi fingerprint-based indoor positioning for in-home monitoring? *2016 12th International Conference on Intelligent Environments (IE)*, 68–75. https://doi.org/10.1109/IE.2016.19

Torres-Sospedra, J., Belmonte-Fernández, Ó., Mendoza-Silva, G. M., Montoliu, R., Puertas-Cabedo, A., Rodríguez-Pupo, L. E., Trilles, S., Calia, A., Benedito-Bordonau, M., & Huerta, J. (2019). 3 - lessons learned in generating ground truth for indoor positioning systems based on wi-fi fingerprinting. In J. Conesa, A. Pérez-Navarro, J. Torres-Sospedra, & R. Montoliu (Eds.), *Geographical and fingerprinting data to create systems for indoor positioning and indoor/outdoor navigation* (pp. 45–67). Academic Press. https://doi.org/10.1016/B978-0-12-813189-3.00003-4

Tran, H., Khoshelham, K., Kealy, A., & Díaz-Vilariño, L. (2019). Shape grammar approach to 3d modeling of indoor environments using point clouds. *Journal of Computing in Civil Engineering*, *33*(1), 04018055. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000800

Van der Ham, M., Kalogianni, E., Lam, M., Sileryte, R., Zhou, K., Van der Spek, S., Verbree, E., & Valks, B. (2014). Rhythm of the campus. *Geomatics Synthesis Project Fall 2014 on Wifi Tracking Technologies*.

Van Der Spek, S., Verbree, E., Meijers, B., Bot, F., Braaksma, H., Braggaar, R., Ligtvoet, B., & Staats, B. (2016). In_sight: Using existing wi-fi networks to provide information on occupancy and exploitation of educational facilities using at delft university of technology. *13 th International Conference on Location-Based Services*, 151. http://resolver.tudelft.nl/uuid:4068d4d0-9497-4bec-9a7c-e45093d68295

Verbree, E. (2021). Lecture Notes of GEO 1003 Positioning and Location Awareness.

Xia, S., Liu, Y., Yuan, G., Zhu, M., & Wang, Z. (2017). Indoor fingerprint positioning based on wi-fi: An overview. *ISPRS International Journal of Geo-Information*, *6*(5), 135. https://doi.org/10.3390/ijgi6050135

Zhang, Z. (2016). Introduction to machine learning: K-nearest neighbors. *Annals of Translation Medicine*, *4*. https://doi.org/10.21037/atm.2016.03.37

A

Project Organisation

This project was not only focused on the development of the product but also the organisational aspect. This section will focus on explaining the project organisation.

A.1. Contributions and Responsibilities

At the beginning of the project, each member was assigned to a specific role what they will be responsible for primarily. These roles were used to manage the structure of the project's organisation. However, the roles did deviate from what was stated in the PID, and the following role division is more accurate to what was actually done:

- Project Manager : Ioannis Dardavesis
- Report Manager : Maundri Prihanggo
- Communication Leader : Guilherme Spinoza Andreo
- Quality Leader : Georgios Triantafyllou
- Technical Manager : Pratyush Kumar and Michiel de Jong

Besides the specific role which is related to the responsibility, there were also contributions where it was spread within the member to be as equal as it could be. Several tasks in the contributions are :

- building indoor model
- creating Wi-Fi fingerprinting algorithm
- creating the radio maps
- building android user application
- testing the application
- writing on the final report

However, each of the group's members come from different backgrounds and have different strengths and weaknesses. The distribution of the contributions was always based on what the members were most interested to do. There were little forces from some members to do one specific activity. However, in the middle of the project, the workload was divided into indoor modelling and Wi-Fi fingerprinting where the groups members were divided equally to do these activities. Nevertheless, if some members were having difficulties, other members were willing to help.

A.2. Communication Plan

During Covid-19 pandemic, students were forced to do academic activities from home. This also affected how the synthesis project of Geomatics students was be conducted. Most of the time, each member of the group had to work remotely to finish the synthesis project while the ideal way is to meet in a group. To tackle this limitation, several days were dedicated to the project for the group to do the discussion and update the progress. Every Wednesday, the progress of the project and plan for the upcoming day were being discussed. The other days were flexible, at least the meeting was being held twice a week. After the Covid-19 measurements restriction was being more relaxed, the online discussion on Wednesday was changed into offline discussion on campus however if there was discussion with the stakeholders and/or supervisors, it was still held in an online way. Some communication tools were being used to during this project :

- **Discord**: mostly for text-based communication and discussion
- **Microsoft OneDrive**: share reading material and other file
- **GitHub**: code sharing, especially for the Wi-Fi fingerprinting algorithm and the building of android application
- **Zoom**: mostly for online-video communication within the group members and/or with the stakeholders
- **Trello**: share progress about what has to be done and what has finished

A.3. Involvement of Stakeholders

There were three stakeholders being involved in this project. TU Delft Campus and Real Estate was the client while ESRI Netherlands and CGI Netherlands were the technology provider. In the PID document, to have the most updated data of the indoor model, a SLAM-based laser scanning method was proposed. This laser scanning method would have been guided and helped by CGI Netherlands. However, after having discussion within the group member and with the supervisor, the CAD data provided by the client is sufficient. And using ArcGIS Indoor with the help of ESRI Netherlands, indoor modelling with the AIIM format was created. Furthermore, with ESRI bi-weekly meetings were held to update them about the progress.

A.4. Time management

This project took 10 weeks in total and during this time, the timeline has been revised two times. The ideal timeline was the one that was proposed on the PID document. Several problems and issues have been measured to create the ideal timeline, however these were only estimations, the real project hasn't been executed. After the start of the project, some measurements took more time than the estimated time, such as the availability of the CAD data as the basis for creating the indoor model, the license of the ArcGIS Indoor which is different from the regular license of ArcGIS Pro given by the campus, changing the hardware from cheap smartphone to Arduino radio and mobile app application including the programming language used. The first revision of the timeline was before the midterm presentation. The midterm presentation was held in week 5 of the project. It was estimated that before the midterm presentation, the indoor model of the study area had already been built and the WiFi fingerprinting application/method had already been finished. However, because of the late availability of the CAD data model and the ArcGIS Indoor license the target of the ideal timeline couldn't be achieved. These problems were understood well by the stakeholders therefore another revision was also being made for the LIDAR indoor acquisition data which is that this method will not be used because may lead to more delay of the project. The second revision was before the final week came. This revision was more to be the realisation of the project timeline.

B

Arduino Script for Live Fingerprinting

The full code of the project can be found at https://github.com/dumigil/Building-Rhythms.

```
#include <WiFiNINA.h>       // use this for MKR1010 and Nano 33 IoT boards
#include <ArduinoHttpClient.h>
#include <RTCZero.h>
#include <ArduinoJson.h>
#include "arduino_secrets_thuis.h"
//#include "arduino_secrets.h"

WiFiSSLClient netSocket;               // network socket to server
const char server[] PROGMEM = "services3.arcgis.com";  // server name
const char route[] PROGMEM= "/jR9a3QtlDyTstZiO/ArcGIS/rest/services/
   Arduino_Table_ROOM/FeatureServer/0/addFeatures";              // API
   route

RTCZero rtc;
const byte seconds = 0;
const byte minutes = 30;
const byte hours = 14;

/* Change these values to set the current initial date */
const byte day = 16;
const byte month = 6;
const byte year = 21;

// request timestamp in ms:
long lastRequest = 0;
// interval between requests:
int interval = 10000;
int status = WL_IDLE_STATUS;     // the WiFi radio's status
HttpClient http(netSocket, server, 443);


void setup() {
  Serial.begin(9600);               // initialize serial communication
  //while (!Serial);        // wait for serial monitor to open
  rtc.begin();
  rtc.setTime(hours, minutes, seconds);
  rtc.setDate(day, month, year);
  // while you're not connected to a WiFi AP,
```

```
  while ( WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to Network named: ");
    Serial.println(SECRET_SSID);

    //at home
    WiFi.begin(SECRET_SSID, SECRET_PASS);
    // at uni
    //WiFi.beginEnterprise(SECRET_SSID,SECRET_USER,SECRET_PASS);
       // try to connect
    delay(1000);
  }

  // When you're connected, print out the device's network status:
  IPAddress ip = WiFi.localIP();
  //Serial.print("IP Address: ");
  //Serial.println(ip);
}
void loop() {
  WiFi.end();
  String dummyData = listNetworks();
  Serial.println("Wifi scan complete, starting eduroam timeout");
  WiFi.end();
  delay(1000);
  while ( WiFi.status() != WL_CONNECTED) {
    Serial.print("Attempting to connect to Network named: ");
    Serial.println(SECRET_SSID);

    //at home
    WiFi.begin(SECRET_SSID, SECRET_PASS);
    // at uni
    //WiFi.beginEnterprise(SECRET_SSID,SECRET_USER,SECRET_PASS);
       // try to connect
    delay(1000);
  }

  if (millis() - lastRequest > interval ) {
    //Serial.println("making request");
    //http.get(getRoute);  // make a GET request
    String contentType = "application/x-www-form-urlencoded";
    http.post(route, contentType, dummyData);

    while (http.connected()) {       // while connected to the server,
      if (http.available()) {        // if there is a response from the
         server,
        String result = http.readString();  // read it
        Serial.print(result);               // and print it
      }
    }
    //  // when there's nothing left to the response,
    http.stop();                     // close the request
    lastRequest = millis();

  }
  Serial.println(freeMemory());

}
```

```
String listNetworks() {
  String dataString = "";
  const char room []PROGMEM= "08.02.00.800";
  int numSsid = WiFi.scanNetworks();
  if (numSsid == -1)
  {
    Serial.println("Couldn't get a WiFi connection");
    while (true);
  }
   String comma = "";

  dataString += "features=[";
  // print the network number and name for each network found:
  for (int thisNet = 0; thisNet < numSsid; thisNet++) {
    if(String(WiFi.SSID(thisNet)) == "eduroam" ||
    String(WiFi.SSID(thisNet)) == "tudelft-dastud" ||
    String(WiFi.SSID(thisNet)) == "Delft Free Wifi" ||
    String(WiFi.SSID(thisNet)) == "TUvisitor"){
      DynamicJsonDocument doc(2048);
      byte bssid[6];
      String address = printMacAddress(WiFi.BSSID(thisNet, bssid));
      String measurement = "";
      dataString += comma;
      doc["attributes"]["RSSI"] = WiFi.RSSI(thisNet);
      doc["attributes"]["MAC"] = String(address);
      doc["attributes"]["Table_ID"] = room;
      doc["attributes"]["Time_Stamp"] = String(rtc.getEpoch());
      doc["attributes"]["BSSID"] = String(WiFi.SSID(thisNet));
      serializeJson(doc, measurement);
      dataString+=measurement;
      comma = ",";
    }
    else{
      continue;
    }

  }
  dataString += "]";

  return dataString;
}

String print2digits(int number) {
  if (number < 10) {
    return "0" + String(number);
  }
  return String(number);
}

String printMacAddress(byte mac[]) {
  String address;
  for (int i = 5; i >= 0; i--) {
    if (mac[i] < 16) {
      address += "0";
    }
```

```
    //Serial.print(mac[i], HEX);
    address += String(mac[i], HEX);
    if (i > 0) {
      address += ":";
    }
  }
  return address;
}

#ifdef __arm__
// should use uinstd.h to define sbrk but Due causes a conflict
extern "C" char* sbrk(int incr);
#else  // __ARM__
extern char *__brkval;
#endif  // __arm__

int freeMemory() {
  char top;
#ifdef __arm__
  return &top - reinterpret_cast<char*>(sbrk(0));
#elif defined(CORE_TEENSY) || (ARDUINO > 103 && ARDUINO != 151)
  return &top - __brkval;
#else  // __arm__
  return __brkval ? &top - __brkval : &top - __malloc_heap_start;
#endif  // __arm__
}
```

C

Python Script for Fingerprint Matching

```
from joblib.parallel import DEFAULT_MP_CONTEXT
import pandas as pd
import numpy as np
from scipy import stats
import urllib3
import json
from sklearn.neighbors import KNeighborsClassifier

#%%
def predict_func(test_json_string, time=0):
    #For the BK model
    # url_path = "https://services3.arcgis.com/jR9a3QtlDyTstZiO/ArcGIS/
       rest/services/Arduino_Table/FeatureServer/0/query?where=ObjectID%3E
       %3D0&outFields=MAC%2C+RSSI%2C+BSSID%2C+Room_ID+%2C+ObjectId+%2C+
       Time_Stamp+&returnIdsOnly=false&returnUniqueIdsOnly=false&
       returnCountOnly=false&returnDistinctValues=false&cacheHint=false&
       sqlFormat=none&f=pjson"
    #For the one room
    url_path = "https://services3.arcgis.com/jR9a3QtlDyTstZiO/arcgis/rest/
       services/Arduino_Table_ROOM/FeatureServer/0/query?where=ObjectID%3E
       %3D0&outFields=MAC%2C+RSSI%2C+BSSID%2C+Table_ID+%2C+ObjectId+%2C+
       Time_Stamp+&returnIdsOnly=false&returnUniqueIdsOnly=false&
       returnCountOnly=false&returnDistinctValues=false&cacheHint=false&
       sqlFormat=none&f=pjson"

    http = urllib3.PoolManager()
    r = http.request('GET', url_path)
    data = json.loads(r.data.decode('utf-8'))

    df_mega = pd.DataFrame.from_dict(data['features'])
    df_mega.loc[:, "MAC"] = df_mega.attributes.apply(lambda x: x["MAC"])
    df_mega.loc[:, "RSSI"] = df_mega.attributes.apply(lambda x: float(x["
       RSSI"]))
    df_mega.loc[:, "Room_ID"] = df_mega.attributes.apply(lambda x: x["
       Table_ID"])
    # df_mega.loc[:, "Room_ID"] = df_mega.attributes.apply(lambda x: x["
       Room_ID"])
    df_mega.drop('attributes', axis=1, inplace=True)

    a = df_mega.MAC.unique()
```

```
    df_mega['macno'] = df_mega.MAC.apply(lambda x: np.where(a==x)[0][0])

    print(f"Max nof unique identifier is: {max(df_mega.macno)}")

    df_mega=df_mega.astype({'RSSI': 'float32'}, copy=False)

    def indexer(a,x):
        x=x.lower()
        ind = np.where(a==x)[0]
        if len(ind)!=0:
            return ind[0]

    #to load the test dataset as a pandas df
    test_df = pd.read_json(test_json_string)
    test_df.loc[:,"MAC"] = test_df.attributes.apply(lambda x: x["MAC"])
    test_df.loc[:,"signal"] = test_df.attributes.apply(lambda x: float(x["
       RSSI"]))
    test_df.drop('attributes', axis=1,  inplace=True)

    # training data ////////////////////////

    x_train = df_mega[['macno' , 'RSSI']]
    y_train = pd.Series(df_mega.Room_ID)
    # y_train = pd.Series(df_mega.Room_ID)

    x_test = test_df[['signal' , 'MAC']].copy()

    x_test['macno'] = x_test.MAC.apply(lambda x: indexer(a,x))
    x_test.drop(['MAC'], axis=1, inplace=True)
    x_test.reset_index(inplace=True, drop=True)
    x_test.dropna(inplace=True)
    x_test = x_test[['macno','signal']]

    try:

        # train kNN model with training data ////////////////
        knn = KNeighborsClassifier(n_neighbors = 7, weights='distance',
           algorithm='auto')
        knn.fit(x_train , y_train)

        # predicting using trained model
        pred = knn.predict(x_test)
        room =str(stats.mode(pred)[0][0])

        return(room)
    except:
        return "Couldn't locate the room you are in :("
```

D

Java Function for Updating Occupancy

```
public void updateOccupancy(String room_id,@NonNull int plusminus ){
        try {
            String sURL = "https://services3.arcgis.com/jR9a3QtlDyTstZiO/
               ArcGIS/rest/services/Map_3D_OOST_700_WFL1/FeatureServer/1/
               query?where=NAME+like+%27%25"+room_id+"%25%27&objectIds=&
               time=&geometry=&geometryType=esriGeometryEnvelope&inSR=&
               spatialRel=esriSpatialRelIntersects&resultType=none&
               distance=0.0&units=esriSRUnit_Meter&returnGeodetic=false&
               outFields=OBJECTID&returnGeometry=true&returnCentroid=false
               &featureEncoding=esriDefault&multipatchOption=xyFootprint&
               maxAllowableOffset=&geometryPrecision=&outSR=&
               datumTransformation=&applyVCSProjection=false&returnIdsOnly
               =false&returnUniqueIdsOnly=false&returnCountOnly=false&
               returnExtentOnly=false&returnQueryGeometry=false&
               returnDistinctValues=false&cacheHint=false&orderByFields=&
               groupByFieldsForStatistics=&outStatistics=&having=&
               resultOffset=&resultRecordCount=&returnZ=false&returnM=
               false&returnExceededLimitFeatures=true&
               quantizationParameters=&sqlFormat=none&f=pjson&token=";
            URL url = new URL(sURL);
            HttpURLConnection request = (HttpURLConnection) url.
               openConnection();

            request.connect();
            //Log.d("REQUEST", "updateOccupancy: "+ request.getContent());

            JsonParser jp = new JsonParser(); //from gson
            JsonElement root = jp.parse(new InputStreamReader((InputStream
               ) request.getInputStream())); //Convert the input stream to
                a json element
            JsonObject rootobj = root.getAsJsonObject();


            Gson gson = new GsonBuilder().create();
            indoorModel value = gson.fromJson(root, indoorModel.class);
            indoorModel.Features[] feat_arr = value.features;

            System.out.println("There are so many features: \t"+ feat_arr.
               length);
```

```
            for (indoorModel.Features iter: feat_arr)
            {

                String mRoom = iter.attributes.NAME;
                int currOcc = iter.attributes.OCCUPANCY;
                String objectID = iter.attributes.OBJECTID;
                currOBJECTID = objectID;
                roomOccupancy = currOcc;
                //System.out.println("There are " + currOcc +" people in
                    room "+mRoom+" with objectID "+objectID);
                //Toast.makeText(MainActivity.this,"You are in room "+
                    mRoom+". There are "+currOcc+" people in the room with
                    you",Toast.LENGTH_SHORT).show();

            }

            request.disconnect();
        }

        catch (Exception e)
        {
            e.printStackTrace();
        }



        if(plusminus > 0){
            JSONObject postRequest = new JSONObject();
            JSONObject mBody = new JSONObject();

            try {
                mBody.put("OBJECTID",currOBJECTID);
                mBody.put("OCCUPANCY", (roomOccupancy+1));
                postRequest.put("attributes", mBody);
                String url = "https://services3.arcgis.com/
                    jR9a3QtlDyTstZiO/arcgis/rest/services/
                    Map_3D_OOST_700_WFL1/FeatureServer/1/updateFeatures";
                RequestQueue requestQueue = Volley.newRequestQueue(this);
                final String requestBody = "features=["+postRequest.
                    toString()+"]";
                System.out.println(requestBody);
                StringRequest stringRequest = new StringRequest(Request.
                    Method.POST, url, new Response.Listener<String>() {
                     @Override
                     public void onResponse(String response) {
                         Log.i("VOLLEY", response);
                     }
                }, new Response.ErrorListener() {
                     @Override
                     public void onErrorResponse(VolleyError error) {
                         Log.e("VOLLEY", error.toString());
                     }
                }) {
                     @Override
                     public String getBodyContentType() {
```

```
                    return "application/x-www-form-urlencoded; charset
                        =UTF-8";
                }
                @Override
                public byte[] getBody() throws AuthFailureError {
                    try {
                        return requestBody == null ? null :
                            requestBody.getBytes("utf-8");
                    } catch (UnsupportedEncodingException uee) {
                        VolleyLog.wtf("Unsupported Encoding while
                            trying to get the bytes of %s using %s",
                            requestBody, "utf-8");
                        return null;
                    }
                }
                @Override
                protected Response<String> parseNetworkResponse(
                    NetworkResponse response) {
                    String responseString = "";
                    if (response != null) {
                        responseString = String.valueOf(response.
                            allHeaders);

                        // can get more details such as response.
                            headers

                    }
                    return Response.success(responseString,
                        HttpHeaderParser.parseCacheHeaders(response));
                }
            };
            requestQueue.add(stringRequest);

        } catch (JSONException e) {
            e.printStackTrace();
        }
    }else{
        JSONObject postRequest = new JSONObject();
        JSONObject mBody = new JSONObject();

        try {
            mBody.put("OBJECTID",currOBJECTID);
            mBody.put("OCCUPANCY", (roomOccupancy-1));
            postRequest.put("attributes", mBody);
            String url = "https://services3.arcgis.com/
                jR9a3QtlDyTstZiO/arcgis/rest/services/
                Map_3D_OOST_700_WFL1/FeatureServer/1/updateFeatures";
            RequestQueue requestQueue = Volley.newRequestQueue(this);
            final String requestBody = "features=["+postRequest.
                toString()+"]";
            System.out.println(requestBody);
            StringRequest stringRequest = new StringRequest(Request.
                Method.POST, url, new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Log.i("VOLLEY", response);
```

```
                    }
                }, new Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError error) {
                        Log.e("VOLLEY", error.toString());
                    }
                }) {
                    @Override
                    public String getBodyContentType() {
                        return "application/x-www-form-urlencoded; charset
                            =UTF-8";
                    }
                    @Override
                    public byte[] getBody() throws AuthFailureError {
                        try {
                            return requestBody == null ? null :
                                requestBody.getBytes("utf-8");
                        } catch (UnsupportedEncodingException uee) {
                            VolleyLog.wtf("Unsupported Encoding while
                                trying to get the bytes of %s using %s",
                                requestBody, "utf-8");
                            return null;
                        }
                    }
                    @Override
                    protected Response<String> parseNetworkResponse(
                        NetworkResponse response) {
                        String responseString = "";
                        if (response != null) {
                            responseString = String.valueOf(response.
                                allHeaders);

                            // can get more details such as response.
                                headers

                        }
                        return Response.success(responseString,
                            HttpHeaderParser.parseCacheHeaders(response));
                    }
                };
                requestQueue.add(stringRequest);

            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
```

E

Schema for Arduino ArcGIS Online Table

```
Layer: Arduino_Table (ID:0)
Name: Arduino_Table
Display Field:
Type: Table
Description:
Copyright Text:
Max Record Count: 20000
Supported query Formats: JSON
Use Standardized Queries: True
HasZ: false
HasM: false
Has Attachments: false
Has Geometry Properties: false
HTML Popup Type: esriServerHTMLPopupTypeNone
Object ID Field: ObjectId
Unique ID Field:
Name : ObjectId
IsSystemMaintained : True
Global ID Field: GlobalID
Type ID Field:
Fields:
MAC (type: esriFieldTypeString, alias: MAC, SQL Type: sqlTypeNVarchar,
   length: 4000, nullable: true, editable: true)
RSSI (type: esriFieldTypeString, alias: RSSI, SQL Type: sqlTypeNVarchar,
   length: 4000, nullable: true, editable: true)
BSSID (type: esriFieldTypeString, alias: BSSID, SQL Type: sqlTypeNVarchar,
    length: 4000, nullable: true, editable: true)
Room_ID (type: esriFieldTypeString, alias: Room ID, SQL Type:
   sqlTypeNVarchar, length: 4000, nullable: true, editable: true)
ObjectId (type: esriFieldTypeOID, alias: ObjectId, SQL Type:
   sqlTypeInteger, length: 0, nullable: false, editable: false)
Time_Stamp (type: esriFieldTypeString, alias: Timestamp, SQL Type:
   sqlTypeOther, length: 256, nullable: true, editable: true)
GlobalID (type: esriFieldTypeGlobalID, alias: GlobalID, SQL Type:
   sqlTypeOther, length: 38, nullable: false, editable: false)
CreationDate (type: esriFieldTypeDate, alias: CreationDate, SQL Type:
   sqlTypeOther, length: 8, nullable: true, editable: false)
Creator (type: esriFieldTypeString, alias: Creator, SQL Type: sqlTypeOther
   , length: 128, nullable: true, editable: false)
```

```
EditDate (type: esriFieldTypeDate, alias: EditDate, SQL Type: sqlTypeOther
    , length: 8, nullable: true, editable: false)
Editor (type: esriFieldTypeString, alias: Editor, SQL Type: sqlTypeOther,
    length: 128, nullable: true, editable: false)
Is Data Versioned: false
Supports Rollback On Failure Parameter: true
```

F

Schema for Indoor Model ArcGIS Online Table


```
Layer: Units_new (ID:4)
View In:   ArcGIS.com Map
Name: Units_new
Display Field: NAME
Type: Feature Layer
Geometry Type: esriGeometryPolygon
Description:
Copyright Text:
Min. Scale: 10009
Max. Scale: 0
Default Visibility: true
Max Record Count: 2000
Supported query Formats: JSON
Use Standardized Queries: True
Extent:
XMin: 486527.944600001
YMin: 6801207.0943
XMax: 486641.156500001
YMax: 6801365.3707
Spatial Reference: 102100 (3857)
Drawing Info:
{"renderer":{"visualVariables":[{"type":"colorInfo","field":"OCCUPANCY","
   valueExpression":null,"stops":[{"value":0,"color":[255,252,212,255],"
   label":"\u003c 0"},{"value":0.75,"color":[177,205,194,255],"label":null
   },{"value":1.5,"color":[98,158,176,255],"label":"1.5"},{"value":2.25,"
   color":[56,98,122,255],"label":null},{"value":3,"color":[13,38,68,255],
   "label":"\u003e 3"}]}],"authoringInfo":{"visualVariables":[{"type":"
   colorInfo","minSliderValue":0,"maxSliderValue":9,"theme":"high-to-low"
   }]},"type":"classBreaks","field":"OCCUPANCY","defaultSymbol":{"color"
   :[255,252,212,106],"outline":{"color":[153,153,153,64],"width":0.75,"
   type":"esriSLS","style":"esriSLSSolid"},"type":"esriSFS","style":"
   esriSFSSolid"},"minValue":-9007199254740991,"classBreakInfos":[{"symbol
   ":{"color":[170,170,170,255],"outline":{"color":[194,194,194,64],"width
   ":0.375,"type":"esriSLS","style":"esriSLSSolid"},"type":"esriSFS","
   style":"esriSFSSolid"},"classMaxValue":9007199254740991}],"defaultLabel
   ":"Other"},"transparency":20}
HasZ: true
```

```
HasM: false
Time Info:
Start Time Field: EditDate
End Time Field: null
Track ID Field: null
Time Extent
[6/3/2021 4:22:48 PM UTC, 6/16/2021 2:17:34 PM UTC]
Time Reference: UTC
Time Interval: 0
Time Interval Units:
Has Live Data: false
Export Options:
Use Time: false
Time Data Cumulative: false
Time Offset: 0
Time Offset Units: esriTimeUnitsCenturies
Has Attachments: false
Has Geometry Properties: true
HTML Popup Type: esriServerHTMLPopupTypeAsHTMLText
Object ID Field: OBJECTID
Unique ID Field:
Name : OBJECTID
IsSystemMaintained : True
Global ID Field: GlobalID
Type ID Field:
Fields:
OBJECTID (type: esriFieldTypeOID, alias: OBJECTID, SQL Type: sqlTypeOther,
    length: 0, nullable: false, editable: false)
UNIT_ID (type: esriFieldTypeString, alias: Unit ID, SQL Type: sqlTypeOther
   , length: 255, nullable: true, editable: true)
ACCESS_TYPE (type: esriFieldTypeString, alias: Access Type, SQL Type:
   sqlTypeOther, length: 50, nullable: true, editable: true)
USE_TYPE (type: esriFieldTypeString, alias: Use Type, SQL Type:
   sqlTypeOther, length: 50, nullable: true, editable: true)
NAME (type: esriFieldTypeString, alias: Name, SQL Type: sqlTypeOther,
   length: 100, nullable: true, editable: true)
Name_Full (type: esriFieldTypeString, alias: Name_Full, SQL Type:
   sqlTypeOther, length: 255, nullable: true, editable: true)
SITE_ID (type: esriFieldTypeString, alias: Site ID, SQL Type: sqlTypeOther
   , length: 255, nullable: true, editable: true)
SITE_NAME (type: esriFieldTypeString, alias: Site Name, SQL Type:
   sqlTypeOther, length: 100, nullable: true, editable: true)
FACILITY_ID (type: esriFieldTypeString, alias: Facility ID, SQL Type:
   sqlTypeOther, length: 255, nullable: true, editable: true)
FACILITY_NAME (type: esriFieldTypeString, alias: Facility Name, SQL Type:
   sqlTypeOther, length: 100, nullable: true, editable: true)
LEVEL_ID (type: esriFieldTypeString, alias: Level ID, SQL Type:
   sqlTypeOther, length: 255, nullable: true, editable: true)
LEVEL_NAME (type: esriFieldTypeString, alias: Level Name, SQL Type:
   sqlTypeOther, length: 100, nullable: true, editable: true)
LEVEL_NUMBER (type: esriFieldTypeInteger, alias: Level Number, SQL Type:
   sqlTypeOther, nullable: true, editable: true)
ASSIGNMENT_TYPE (type: esriFieldTypeString, alias: Assignment Type, SQL
   Type: sqlTypeOther, length: 255, nullable: true, editable: true, Coded
   Values: [hotdesk: Hot Desk], [hotel: Hotel], [none: None], ... 2 more
   ...)
```

```
AREA_GROSS (type: esriFieldTypeDouble, alias: Gross Area, SQL Type:
    sqlTypeOther, nullable: true, editable: true)
AREA_NET (type: esriFieldTypeDouble, alias: Net Area, SQL Type:
    sqlTypeOther, nullable: true, editable: true)
AREA_UM (type: esriFieldTypeInteger, alias: Area Unit of Measure, SQL Type
    : sqlTypeOther, nullable: true, editable: true, Coded Values: [1:
    Square miles], [2: Square kilometers], [3: Acres], ... 11 more ...)
ELEVATION_RELATIVE (type: esriFieldTypeDouble, alias: Relative Elevation,
    SQL Type: sqlTypeOther, nullable: true, editable: true)
HEIGHT_RELATIVE (type: esriFieldTypeDouble, alias: Relative Height, SQL
    Type: sqlTypeOther, nullable: true, editable: true)
VERTICAL_ORDER (type: esriFieldTypeInteger, alias: Vertical Order, SQL
    Type: sqlTypeOther, nullable: true, editable: true)
SOURCE_NAME (type: esriFieldTypeString, alias: Source Name, SQL Type:
    sqlTypeOther, length: 50, nullable: true, editable: true)
SOURCE_PATH (type: esriFieldTypeString, alias: Source Path, SQL Type:
    sqlTypeOther, length: 255, nullable: true, editable: true)
SOURCE_TYPE (type: esriFieldTypeString, alias: Source Type, SQL Type:
    sqlTypeOther, length: 50, nullable: true, editable: true)
SOURCE_METHOD (type: esriFieldTypeString, alias: Source Method, SQL Type:
    sqlTypeOther, length: 50, nullable: true, editable: true)
Capacity (type: esriFieldTypeDouble, alias: Capacity, SQL Type:
    sqlTypeOther, nullable: true, editable: true)
Covid_Occupancy (type: esriFieldTypeDouble, alias: Covid_Occupancy, SQL
    Type: sqlTypeOther, nullable: true, editable: true)
Shape__Area (type: esriFieldTypeDouble, alias: Shape__Area, SQL Type:
    sqlTypeDouble, nullable: true, editable: false)
Shape__Length (type: esriFieldTypeDouble, alias: Shape__Length, SQL Type:
    sqlTypeDouble, nullable: true, editable: false)
GlobalID (type: esriFieldTypeGlobalID, alias: GlobalID, SQL Type:
    sqlTypeOther, length: 38, nullable: false, editable: false)
OCCUPANCY (type: esriFieldTypeInteger, alias: Occupancy, SQL Type:
    sqlTypeOther, nullable: true, editable: true)
CreationDate (type: esriFieldTypeDate, alias: CreationDate, SQL Type:
    sqlTypeOther, length: 8, nullable: true, editable: false)
Creator (type: esriFieldTypeString, alias: Creator, SQL Type: sqlTypeOther
    , length: 128, nullable: true, editable: false)
EditDate (type: esriFieldTypeDate, alias: EditDate, SQL Type: sqlTypeOther
    , length: 8, nullable: true, editable: false)
Editor (type: esriFieldTypeString, alias: Editor, SQL Type: sqlTypeOther,
    length: 128, nullable: true, editable: false)
Templates:
Name: New Feature
Description:
Drawing Tool: esriFeatureEditToolNone
Prototype:
Attributes:
ASSIGNMENT_TYPE: none
Is Data Versioned: false
Supports Rollback On Failure Parameter: true
```