2.5 Develop the proposed technique for zone digital signature generation as a mobile client service.

The start would be with the metrics that we use then how our final ML models of the system are created and trained then after that is the discussion of the deploying these models on the developed mobile client service that we would call it by WMS (Workplace Monitoring System) mobile application.

1- Metrics for Geofence Digital Signature (GDS) Generation

- ***Metrics for Signature Generation***

The system idea needed to be robust and ideal for any environment without the effort to make any sort of infrastructure. Different metrics have been studied and collected that can be accessed from any traditional commercial smartphone. Various smartphone sensor-based metrics were used to indicate the IO state as in [1], but these sensors-based metrics such as the light sensor and GPS have many disadvantages. The light sensor lacks the generalization on the same environment, the GPS is not an optimal indication for indoor environments with high battery consumption and both provide low accuracy of detection in general [2]. Then, we decided to work on the metrics that would be specific to each workplace and will not change with time and have the optimal battery consumption.
In this subsection, different metrics are studied to be used for generating the digital signature for any geofence in the workplace environment. First, Wi-Fi wireless metrics such as Wi-Fi basic service set identifier and Wi-Fi received signal strength indication parameters are used for GDS generation. This is due to the availability of them, where there can be collected in the smartphone without any requirement for additional hardware installation. Wi-Fi BSSID can be considered as a distinctive identifier for each Wi-Fi AP [3] that characterizes the medium access control (MAC) address for the Wi-Fi AP. However, Wi-Fi BSSID can be collected even if the person is far from the AP with an appropriate distance. Therefore, Wi-Fi RSSI parameters are essential to be used along with the Wi-Fi BSSID to indicate the distance from the Wi-Fi AP for any person. The used Wi-Fi wireless metrics of all neighboring Wi-Fi APs can be collected by scanning the smartphone's Wi-Fi to be used in digital signature generation for any geofence in the workplaces.
To increase the GDS robustness, the magnetic field can be used along with the Wi-Fi wireless metrics to uniquely identify each geofence and detect the crossing of its virtual borders. The magnetic field can be sensed by the magnetometer equipped in smartphones. The magnets affect the Earth's magnetic field and construct a unique signature that can be detected by the smartphone's magnetometer and used to track the geofences violation. The magnetic field of the magnets depends on the magnetic

moment of the magnet, relative position to the smartphone's magnetometer, and the magnetic permeability. Therefore, the three-axis parameters of the magnetic field S_x, S_y and S_z sensed by the magnetometer can be given by [4].

$$\begin{pmatrix} S_x \\ S_y \\ S_z \end{pmatrix} = \frac{\mu_0}{4\pi r^5} \begin{vmatrix} 3x^2 - r^2 & 3xy & 3xz \\ 3yx & 3y^2 - r^2 & 3yz \\ 3zx & 3zy & 3z^2 - r^2 \end{vmatrix} \begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix}, \quad (1)$$

where x, y and z are the distance between the magnet and the magnetometer in the x-axis, y-axis, and z-axis, respectively. Meanwhile, r denotes the norm distance from the magnet to the smartphone's magnetometer and μ_0 represents the magnetic permeability. Moreover, M_x, M_y and M_z denote the magnetic moment of the magnet.

- ***Spatial Features for Magnetic Field***

The smartphone's magnetometer can collect the three-axis parameters of the magnetic field S_x, S_y and S_z. In addition, the pitch P, roll R, and azimuth ψ parameters can be sensed by the smartphone's accelerometer. These last three parameters are the output from the mobile orientation sensors or the position sensors which are used in determining the smartphone's orientation [5]. The magnetometer field strength S can be given by

$$S = \sqrt{S_x^2 + S_y^2 + S_z^2}. \quad (2)$$

According to [6], the horizontal and vertical parameters of the magnetic field S_H and S_V can be calculated as follows

$$S_V = -S_x \sin(P) + S_y \sin(R) + S_z \cos(P)\cos(R), \quad (3)$$

$$S_H = \sqrt{S^2 - S_V^2}. \quad (4)$$

Using the parameters S_H and S_V in the GDS generation can help in reducing the magnetometer orientation effect. Therefore, the 9 features $S_x, S_y, S_z, P, R, \psi, S, S_V, S_H$ are used to represent the spatial features for the magnetic field. Each feature is represented by the mean and the interquartile range (IQR) statistics, which can be given as follows,

$$\mu = \frac{1}{c}\sum_{i=1}^{c} x_i, \quad (5)$$

$$IQR = Q_3 - Q_1, \quad (6)$$

where μ is a central value of the data. Meanwhile, IQR is the difference between the upper quartile Q_3 corresponds with the 75th percentile of the data and the lower quartile Q_1 corresponds with the 25th percentile of the data.

2- ML-Based Models for GDS Generation and Matching

In this section, we would discuss the proposed ML-based techniques for generating the GDS for our geofenced workplaces. The idea of applying this ML approach is for performing a binary classification task to identify the inside/outside (IO) status of a user to determine whether the user is inside his pre-defined cluster of geofences. This is applied by fetching the user's current geofence metrics by a developed mobile application. The output or the prediction of the ML model would be the indication of the presence of the users inside is associated cluster or not. If not, an alert would be sent to the admins of the web management system for taking the proper action. As we would be exposed to more than one geofence for each user, we still used the binary classification approach instead of a multi-class one. This is done to have the ability to add more geofences later as separate models to the system, creating new clusters or modifying them.

- ***Features Collection and Extraction***

The features collection would be the first step for building our ML models for the GDS generation. The chosen metrics were the Wi-Fi BSSID and Wi-Fi RSSI data along with the magnetometer different metrics Sx, Sy, Sz, S, S_H, S_V ,P, R and ψ. So, after some tests, we decided to go with the first top-five BSSIDs and RSSIs from the Wi-Fi metrics which are scanned by the smartphone and those would make our first ten features in our dataset. Then, they would be followed by the nine features of the magnetometer and accelerometer sensors. The last thing would be our label which indicated the IO status in a certain geofence dataset.
The availability of the Wi-Fi metrics inside different workplaces and the changing nature of the magnetic field inside of buildings from the outdoor environment make them ideally good for our tests. Also, the addition of the orientation parameters of the user's smartphone supported the ML model accuracy as an indication of the real movement and placement of the smartphone in the workplace environment. This was done during the collection of the dataset features, as we tested the user's smartphone usage by putting it on tables, holding it to browse some feed and messages, and walking with it in the trouser pocket.
The datasets for each testing location were collected by a developed Android application on Android Studio Arctic Fox (2020.3.1) edition running on a Xiaomi Redmi 9T. It is equipped with an Octa-core Qualcomm SM6115 Snapdragon 662 chip and Adreno 610 GPU with 4 GB random access memory (RAM) on Android 11 (API 30). The datasets were collected at 20Hz (20 samples in 1 second) by the Android application in which the magnetometer and the accelerometer vary in each sample, while the Wi-Fi metrics are captured differently as we could get one new sample every 2.5-3 seconds from the Wi-Fi sensor.
These features were collected as a time-series data to make some statistical calculations on them. This approach was applied to increase the dimensionality of the

magnetic data further for more features extraction. These datasets were in the range of 15000 samples for each geofence dataset we tested on. This size of samples was acceptable in our tests. We could have increased the number of samples more, but due to the relatively small size of the geofences of different rooms or halls in real life, the variety of the wireless metrics and magnetic data will not be much. The readings would be repeated which causes a degradation in the performance of our ML techniques. On the other hand, less redundant data will lead to less possibility of making decisions based on redundant data/noise, which reduces any future overfitting.
Also, the features would be used interchangeably with different preprocessing steps according to the used model type in each state or situation, this would be discussed later in **appendix (2/T2.7).**

- ***TSMF Data Preprocessing***

The datasets for the tested geofences are collected as time-series data at 20Hz from the magnetometer and accelerometer sensors, and the Wi-Fi metrics were nearly at 0.3Hz. Before going into details, it is worth mentioning that the reason for increasing and expanding our features from just the traditional magnetic field strength to having nine features of magnetic data was for achieving the possibility of using them on their own. Due to the low dimensionality of magnetic field strength alone. This is done mainly for covering the case of the absence of Wi-Fi signals in any location.
Different statistical operations were applied differently to our metrics in what is called the Time-Series Metrics Fusion (TSMF). There are three preprocessing TSMF situations in which the datasets of our system architecture are tested: Magnetic data only; Wi-Fi metrics only; and Wi-Fi with magnetic data, in each time we tested a different portion of the features.
First, the magnetic field and orientation TSMF. This is done by processing our 20 samples into one reading using two statistical operations, the Mean and Interquartile range. So, 20 samples of the nine features are taken each second and their mean is calculated to diminish the 20 samples into one single reading. Also, the IQR is calculated in the same manner. Then, the 9 means and 9 IQRs are concatenated together to get 18 features with the addition of the IO indication label. These two operations were chosen specifically after some conducted tests between some other statistical operations such as variance, kurtosis, etc. Now, the final preprocessing steps are done by splitting the data into 80% training dataset and the rest as a testing dataset. The traditional encoding of the IO state into 0 and 1 as inside and outside labels, respectively. The last step before training the model would be to standardize the data or use the StandardScaler, which standardizes the features by subtracting the mean and then scaling it to unit variance [7]. Unit variance means dividing all the values by the standard deviation, this method is applied to all our features of magnetic data.

Secondly, the Wi-Fi metrics TSMF. The features would the first five BSSIDs and their RSSIs which forms 10 features then the IO label is added to each sample, the number of samples were in the range 600 – 900 sample. As the RSSI varies from 0 to -100 dBm, so that the absence of the measurement of any of the top five Wi-Fi's RSSI would be marked as -100 dBm to make our model converge to higher results [8]. The corresponding missing BSSID of an RSSI would also get a value of -100 dBm to indicate its absence. During GDS matching, if an unseen Wi-Fi network BSSID is detected as a new input to the trained model, it is treated as the missing one with a value of -100 to prevent the confusion of the model and the tests showed the expected results indeed. To prevent and overcome the problem of unseen Wi-Fi networks after training on certain data and the addition of new permanent Wi-Fi signals to a geofence location, a technique would be applied by retraining the model at certain time points (e.g., during idle state, every night) using a new batch of collected wireless metrics so the model can identify them in future inferences. The last part is the encoding and labeling of the BSSIDs features in each row in the dataset. Followed by the same steps of encoding the IO state, splitting the dataset, and the standardization of it.
Thirdly, the usage of the two types of features after the preprocessing steps were done. So, we get a total of 18 magnetic data features in addition to 10 Wi-Fi metrics features for a total of 28 features with the IO label in the end.

- ***ML-Based Classification Models***

Different ML techniques [9] can be used to provide effective solutions for various real-life problems. Before studying the ML techniques, it is better to understand the difference between supervised and unsupervised categories of machine learning [10]. Unsupervised learning is when the model is trained with unlabeled data. This means that the model will have to find its own features and make predictions based on how it classifies the data. Supervised learning is when the machine learning model is trained using labeled data. It means that you have data that already have the right classification associated with them. One common use of supervised learning is to help you to predict and classify a class from a set of classes based on the input data. With supervised learning, the models would need to be rebuilt after acquiring new data to make sure that the predictions returned are still accurate. The ML-based supervised techniques that used in our experiments are Support Vector Machine (SVM) [11], k-Nearest Neighbor (kNN) [12], Random Forest [13], Artificial Neural Network (ANN) [14], Convolutional Neural Network (CNN) [15].

- ***Training and Testing Processes***

During the training and testing process of the models on our geofences datasets, After various testing, the suitable ML models parameters were discovered as would be briefly discussed in this section. Firstly, the classical machine learning models. In SVM model, we have C equals 1.0, kernel: rbf, and gamma: scale. The kNN with n_neighbors is equal to 5. Then random forest model with n_estimators equal to 100, max_depth: None, and max_features: auto.

Secondly, the deep learning models. The ANN model consists of three layers: input, hidden and output classifier dense layer with (10-8-1) neurons with 10 input features. Each layer has a ReLU activation function except the output with a sigmoid one recommended for binary classification. The optimization algorithm and the loss function are the Adam optimizer and the binary cross-entropy, respectively. The training epochs equal 150 with early stopping that applied to prevent overfitting of the training process. In CNN model, the difference in the preprocessing step is changing from the StandardScaler to the MinMaxScaler [16]. For each value in a feature, the MinMaxScaler subtracts the minimum value in the feature and then divides it by the range. The range is the difference between the original maximum and the original minimum. MinMaxScaler preserves the shape of the original distribution. The required range for the feature returned by MinMaxScaler is 0 to 1. To make the data ready as an input to the CNN layers, we reshaped it from an array of 10 by 1 to an array of 5x2x1 in the case of Wi-Fi metrics, 9x2x1 in case of magnetics data and 14x2x1 in case of using all the features. The model has three layers of convolutional 2D layers with (64-128-256) filters, a kernel of 2x2 then 1x1 and the input size has dimensions 5x2x1 in the case of Wi-Fi metrics. Then used Conv 2D instead of the Conv 1D due to reshaping the data as an 2D image in the CNN models after using the TSMF approach on the time series data collected. This adjustment increased the accuracy by 2% to 3% using our dataset. The activation functions are ReLU. Due to the small input size, this CNN model does not have max-pooling layers. Finally, we will flatten the output of the CNN layers, feed it into two fully-connected dense layers with (100-50) neurons, and then to a sigmoid layer for binary classification with training epochs also equal to 150 with early stopping applied. Both ANN and CNN had batch size equals 32. Some other parameters were used is the validation split with a value of 0.1 and model checkpoints.

During the implementation and testing, we used TensorFlow and Keras library [17] for deep neural networks and Scikit-learn [37] for the classical machine learning models implementation and the typical machine learning operations.

3- ML *Experimental* Setup

The operating idea of the system would be illustrated in this section. The ML system would operate in two modes, a training phase and a detection phase as shown in Figure 1. The training phase involves the collection of the dataset of our chosen features in each of the pre-defined geofences for training the ML models by a smartphone admin application. Then, we apply the TSMF preprocessing to generate our required GDS datasets to train our chosen ML models. On the other hand, in the detection phase, each user's associated geofences would have their own ML model deployed locally on his smartphone client application. It would detect the user's location periodically at a defined time interval. So, we will detect if the user is inside his cluster or not. Each of the assigned clusters will be set to cover certain rooms or halls to manage the density of the people for monitoring the movement of employees in the workplaces. The real-time detected locations and the GDS of the users are uploaded to the web management system for tracking them automatically. Each group of users will be assigned by the system admin to a cluster of geofences based on the nature and the requirement of the work and the workplace environment.

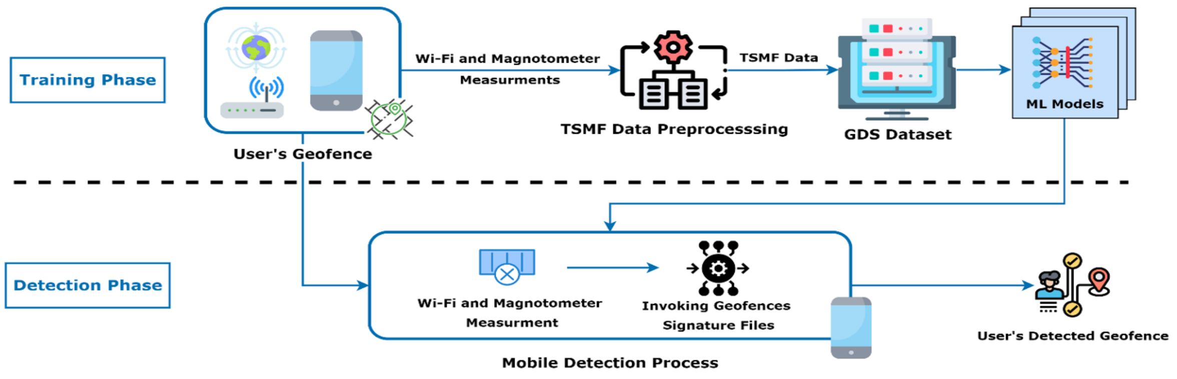


Figure 1. ML system operation diagram.

The detection of the user's location occurs by the ML prediction function on the mobile application. It is invoked periodically at regular time intervals to collect the wireless metrics and outputs a location. If the user is a student in a university as an example, he will be assigned with the cluster of certain lecture halls 1, 2, and 3, where he takes most of his lectures in hall 1. But certainly, he is not assigned with the workers' offices. The prediction function will have high priority weights for detecting his location with the model of lecture hall 1 than the other two. Therefore, the sequence of checking would be lecture hall 1, 2 then 3. If there is a matching, the generated output is the geofence name for any geofence in user's cluster. This implies that this person is inside his cluster, but if the output is outside each one of them after a predefined violation time interval, this indicates that the person is outside his pre-defined cluster. Then, an alarm will be produced to the user through the mobile application and an alert will be generated in the web management platform to inform the supervisors and authorized personnel of this violation to take proper action. The number of persons inside a certain geofence would be calculated in the web management platform to monitor the density of people and for the implementation of our idea of dynamic clustering. The system admins would set some pre-defined control measures so that the clusters would change over time according to the density of individuals.

4- WMS mobile application ML models deployment

The WMS's mobile application system aims to provide the WMS web management system with necessary geofence information regarding the user. This information is provided to the application by means of the GPS, Magnetic Field and Wi-Fi metrics of the mobile sensors that will be taken as location data to provide feedback on the user's actual geofence. The previously mentioned metrics provides the user's geofences digital signature which maps the real-time location that the app will use later to provide feedback to the admins on the state of their employees or determine the service to provide for regular clients as an example of a client in a bank queue.

The mobile application was developed using Flutter framework (Flutter version 2.8) [18] on Android Studio Arctic Fox (2020.3.1) edition and the architectural diagram is shown in Figure 2. The architecture of the app logic is structured as layers and each layer is an input to the next so all layers depend on each other, except for the top-layer, as will be shown and discussed later in this section.

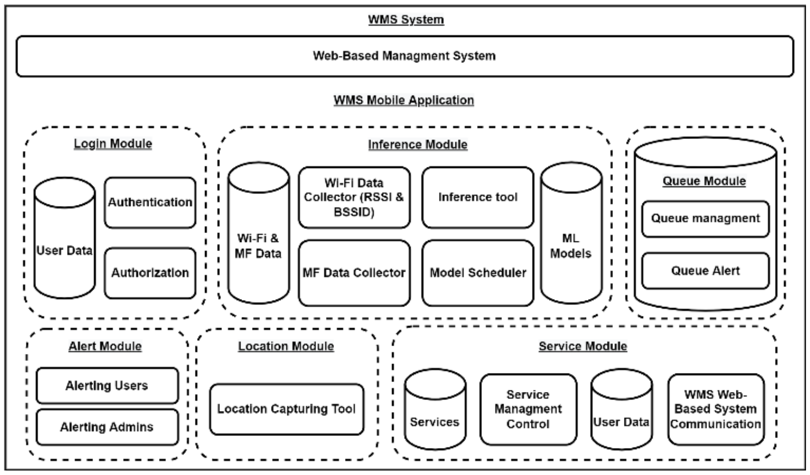


Figure 2. WMS Mobile System Architectural Diagram.

This section will demonstrate the application's layers of operation, and the hierarchy is the same implemented inside the code, this will provide the basic understanding of how the application works from the inside. Figure 3 is a block diagram of the layers of the application, the diagram shows that there are three layers of operation that this section will cover:

The first layer is the workplace layer, which will be the whole workplace for example a bank. This layer prevents access to other layers unless the client enters the building. This layer for simplicity will be considered as a circle in the map, the building will have a center and a radius. The web management system as mentioned provides the center and the radius of each organization. This layer takes the GPS readings using the location collector tool from the location module. Then the mobile application matches it with the data returned from the web management system for determination of the working place organization. If the location of the user is inside the organization coordinates its processed to the second layer, else the application will notify the user that he is not inside the working place area.

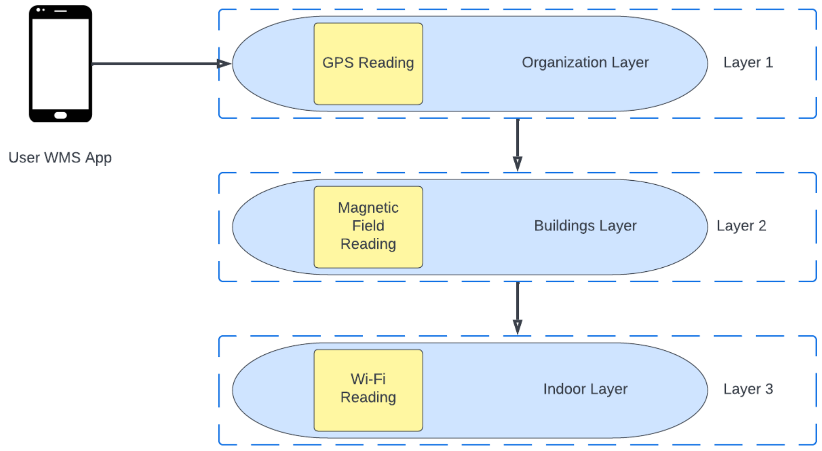


Figure 3. Block diagram of the layers of the application

The second layer activates only if the first layer's output is a known workplace. This layer uses MF data that was stored using the MF data collector and store these data in the Wi-Fi & MF database which will provide the user inside which building in the whole workplace. Using the Model scheduler to help to reduce the number of executed machine learning models that will know which room the user in and then provide service/feedback. Then the output of this layer goes to the last layer to determine the room of the user. The MF layer in the mobile application starts with the collection of the magnetometer sensor main parameters [19] (The three-axis parameters of the magnetic field Sx, Sy, Sz, Azimuth, Pitch, Roll), then using a formula for conversion takes these parameters to prepare them for the mobile inference to use in the machine learning models. This preparation is made using the standard deviation and mean of each set of data.

The last layer takes the output of the second layer and according to it. From the Model scheduler the models are rearranged to start with the models that's actually found in the detected building which helps to reduce the time of running all the models. The Wi-Fi data collector is used here to capture the exact room inside an organization using the BSSID and the RSSI. These parameters are being inserted to the machine learning model that predicts the place which the client's place inside the organization. The Wi-Fi layer scans the available Wi-Fi network and get the RSSI and BSSI of each available network that the mobile can connect to. The top 5 Wi-Fi networks are taken and if there's less than 5 the algorithm automatically adds (-100) [17] till the total of the networks collected reaches 5 then starts to normalize these data (5 RSSI, 5 BSSID) using the standard deviation and mean according to processing files which are received with the model for the proper configuration according to each geofence to be ready for inference using the Inference tool.

During the inference of the ML models in the WMS mobile app TensorFlow Lite framework [20] is used. It is a framework from Google that can run machine learning models on Android and iOS devices that contains tools used for all types of neural network-related apps. The WMS mobile application takes the TFLite model stored in the models database and utilizing it for better usage and to take less memory than the normal trained model. Calling this framework in the Flutter environment provides the necessary function to make inference to the TFLite model by taking as input the scanned and normalized Wi-Fi and MF data found in the Wi-Fi & MF data and then make inference using the inference tool to it to produce the prediction on the location of the user.

The WMS mobile app is designed for the workplace monitoring and the service providing. When the application starts it asks if the user is working in an organization or is waiting for a service. If the user is an employee he will choose to login as an employee, else the user is just headed there for a service. In both cases the user will login to the server using a special key, this key will determine which data is to be taken back from the server.

So, the main functions in the WMS mobile app are the workplace monitoring and the service providing. The workplace monitoring part will give the user/employee feedback on his/her organization and the working room they are in at a given moment shown in Figure 4. While on the backend of the Graphical User Interface (GUI) the application every 15 seconds will recheck the real-time location of the user and send to the server if and only if there's an update happened to the location of the user like the user for example left the working room and went to another room or if the user went to an unidentified/unauthorized room.

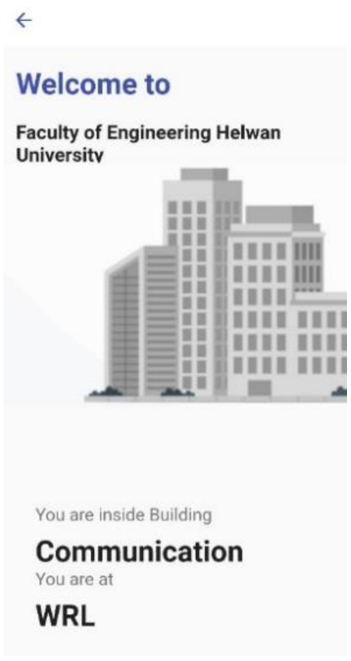
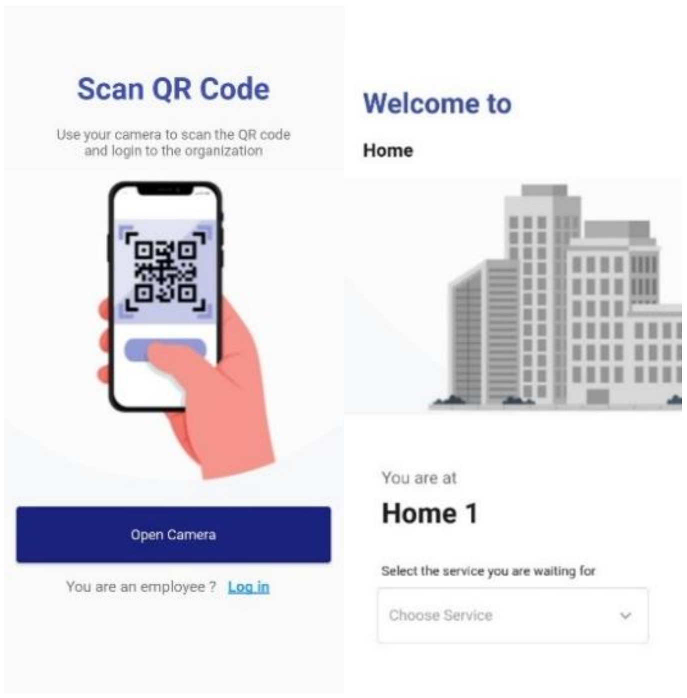


Figure 4. Workplace detection screen

The service Module part of the app. Figure 5(a) is the first screen that the user will see upon scanning a QR code, this screen will tell the user the organization name, the room that the user is in and will provide a selection of multiple services that this room can offer. When the user tabs on a service from the list, the user is taken to another screen, which will show the name of the organization, as in Figure 5(b), chosen service and the user's number in queue for the selected service. The mobile application scans the geofence digital signature of the user each 15 seconds and each time checks if the user is still in the service area or moved from it, when the user choose a service. The application sends to the web management system to tell it that the user chose a particular service. When the returned data from the web management system states that a specific area to be buffering (i.e. waiting area), the mobile application starts to provide the services that came along with the data from the web management system to view them on the UI.So the web management returns to the application the user's number in the Queue using the queue module to wait for his turn. Whenever the queue changes the web management sends to each user inside the service a notification that the queue has been changed thus changing the queue on the GUI.


Scan QR Code
Use your camera to scan the QR code and login to the organization
Open Camera
You are an employee ? Log in
Welcome to
Home
You are at
Home 1
Select the service you are waiting for
Choose Service

(a) (b)

Figure 5. (a) Login to the organization, and (b) Service area main screen

The famous google framework for real-time database (Firebase) is being used inside the mobile application, especially FCM [21] (Firebase Cloud Messaging). The FCM provides the mobile application with messages coming from the database to do a specific action. This was used to allow the web management service to communicate with the mobile device without the need to wait for the mobile device to send a request to the web management first in order to get a specified response. Such feature is implemented using the Google APIs provided from the firebase package in the official web site for firebase integreation with flutter. The web management system sends 3 tags, when the tag is alret the mobile application uses a plugin called “flutter_local_notification” to send the notification to the admins of the workplace. When the tag is update, the mobile application starts to call the web management system to get the updated data of the user. When the tag is queue update this tells the mobile application that the queue in a specific service has changed so it updates its value on the UI.

2.6 Infield test for the developed mobile client service

For testing the system, we have chosen three different buildings with five lecture halls and two labs to represent the proposed buildings and indoor geofences in the Faculty of Engineering, Helwan University, Cairo, Egypt (FE-HU) as a workplace geofence to represent a real practical implementation of our system. The proposed approach is tested on three buildings: Communication Department, Mechanical Department, and the Neo Lecture Halls Building. Inside them, we have chosen five lecture halls which are Lecture Hall 1, Lecture Hall 2, Maraa'shly Lecture Hall, Mech 606 Lecture Hall and Lecture Hall 26. Two other labs were chosen which are Wireless Research Lab (WRL) and Innovation and Product Development Support Center (IPDSC). Figure 6 shows the buildings and the chosen locations inside them. Figure 7 and 8 have some images of the indoor and outdoor locations we have tested on.

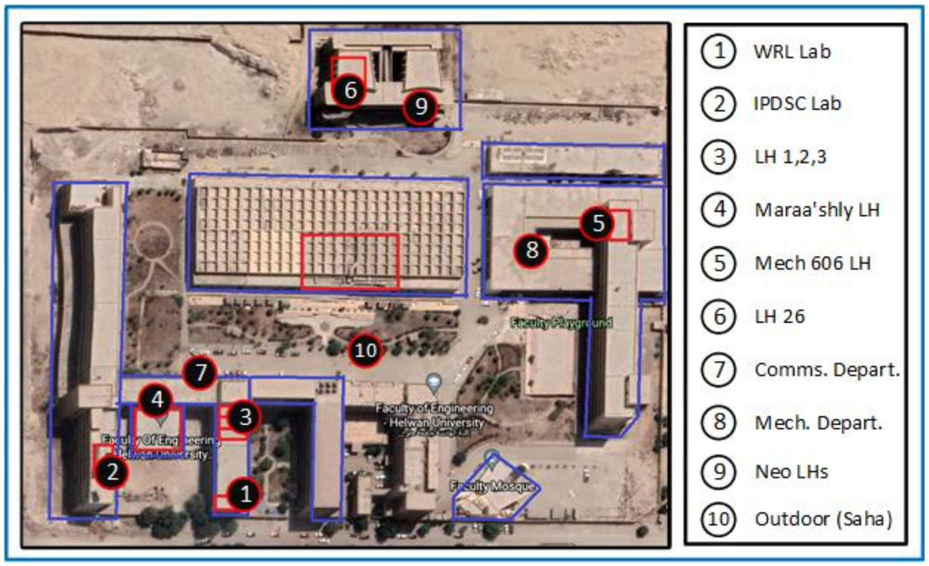


Figure 6. FE-HU top-view map.

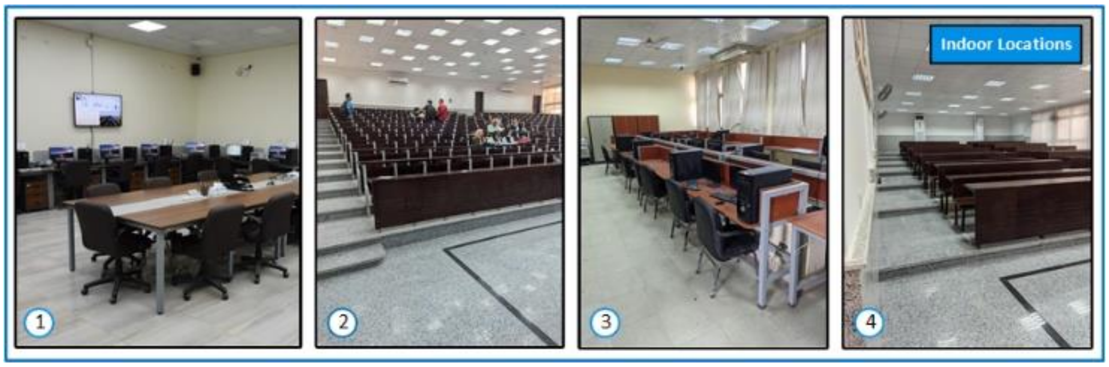


Figure 7. Images of indoor locations.

Figure 8. Images of outdoor locations.

To test the system practically in (FE-HU) locations we have chosen some metrics as showed in Table 1. First is the detection accuracy of each location, as each time the mobile detects the geofence and outputs the right geofence to the user and the WMS system. The results showed that all of them are above the 94% which is significantly good results. Then the distance measurement metric which is the distance the system takes to detect that the geofence is really changed into a new one. The results we got mostly is 1 meter and a range of 250 cm. The last metric is the time to detect changes when the user moves from one geofence to another. The time is roughly 2.5 seconds. These results approved how the system is robust enough practically in real-life tests in our proposed workplace environment.

TABLE 1. WMS results in FE-HU.

Geofence	Detection Accuracy	Distance Measurement	Time to Detect Changes
WRL	0.95	1.34m	2.87s
IPDSC	0.96	1.125m	2.2s
LH1	0.94	1.22m	2.54s

The previous statistics are captured from the app running on an OPPO F5. It is equipped with an Octa-core 2.5 GHz Cortex-A53 CPU and Mali-G71 MP2 GPU with 4 GB random access memory (RAM) on Android 7 (API 25). Other tests were done on a Xiaomi Redmi 9T. It is equipped with an Octa-core Qualcomm SM6115 Snapdragon 662 chip and Adreno 610 GPU with 4 GB random access memory (RAM) on Android 11 (API 30).

2.7 Evaluate the performance of the developed Geofence digital signature service.

- ***Tests and Results Analysis***

Now the evaluation and the results of the developed ML models after training and testing can be discussed. The results analysis is done by applying Wi-Fi and MF models separately, and the combined WF-MF models on three different indoor locations and other three different buildings (Outdoor) locations. From now on, we would call these different types of models WF-ML, MF-ML and WF-MF models. The target of this analysis is to find the best-used metrics in different situations.
In Figures 9, 10 and 11, we can see that the results of the indoor locations indicate different behavior. This is due to the nature of the datasets in each location and the availability of the metrics for the GDS generation. In the WRL Lab, the WF-ML and the WF-MF shows nearly the same performance, and they are outperforming the MF-ML. In Mech 606 LH WF-ML and MF-ML does not have a significant change, and WF-MF is worse than them. In LH 26 WF-MF has superior performance due to the low number of Wi-Fi access points in this LH. We can conclude by studying the three locations that the WF-ML is a better model for the indoor locations in the case of the availability of at least two or three Wi-Fi APs. WF-MF was good in two cases but having a model inference with all these parameters would make a big amount of processing power and time in the long range.

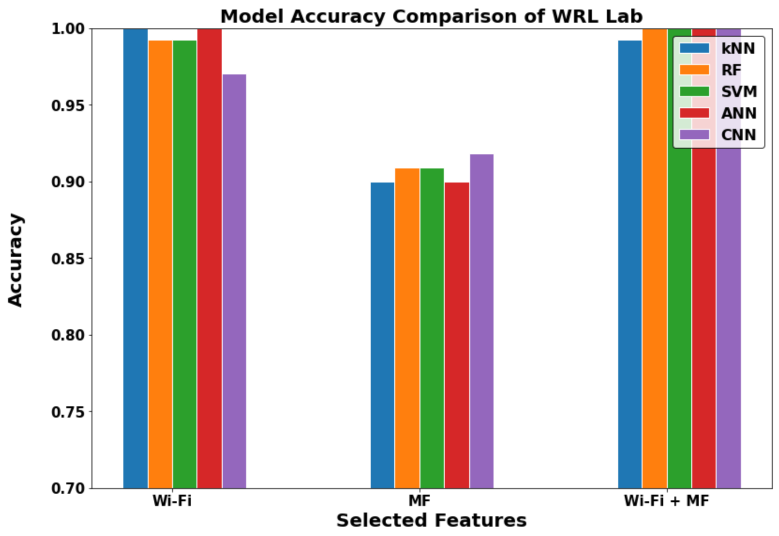


Figure 9. The models comparison in WRL Lab (Indoor).

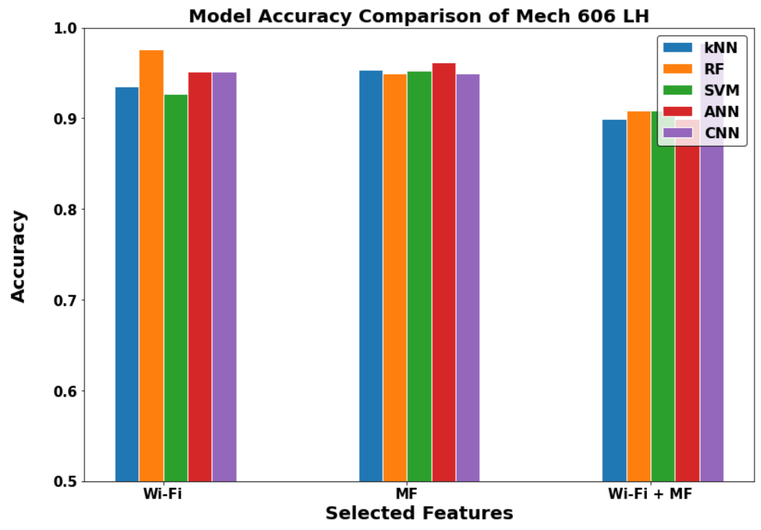


Figure 10. The models comparison in Mech 606 LH (Indoor).

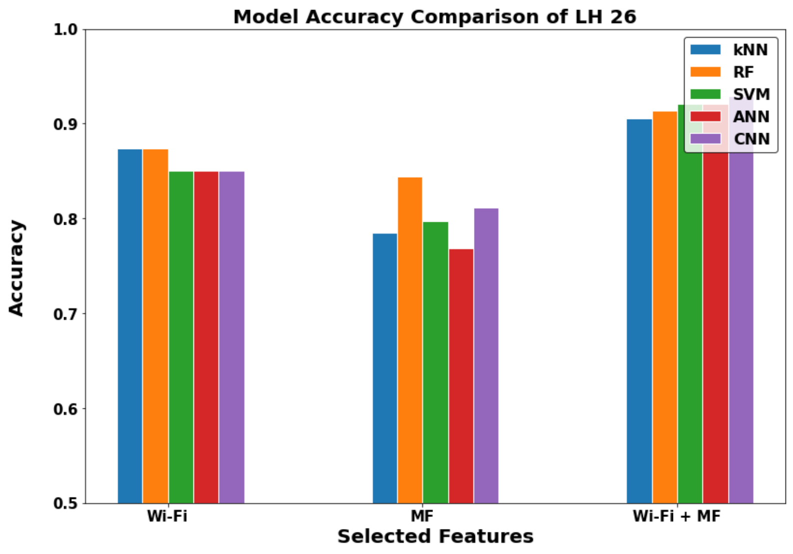


Figure 11. The models comparison in Lecture Hall 26 (Indoor).

The buildings or the outdoor environment layer in Figures 12, 13, and 14 can now be discussed. In the communications department building and Neo Lecture Halls building the MF-ML have noticeably higher performance. The mechanical department building had better performance with WF-MF. It is important to note that most of the outdoor environments lack the presence of Wi-Fi coverage. Therefore, we would not get good results with WF-MF or WF-ML as in the first two locations discussed. Then, we can conclude that MF-ML is the best model for the buildings (outdoor) locations.

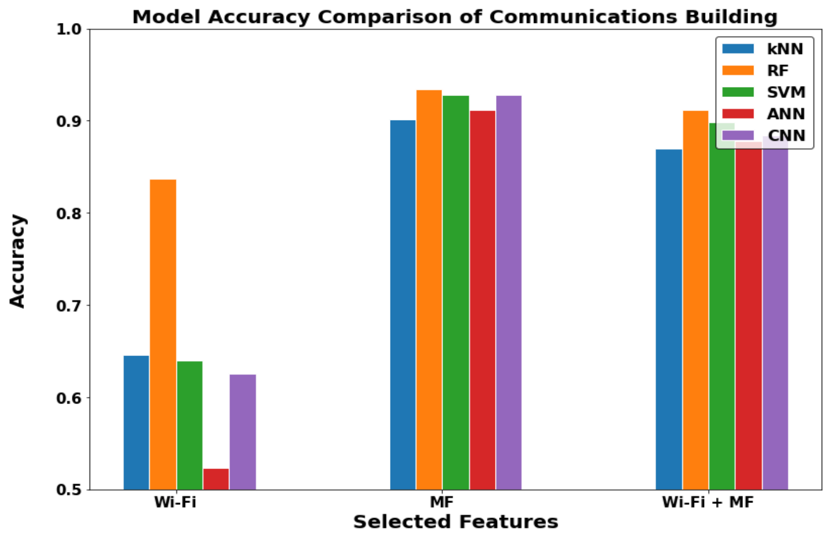


Figure 12. The models comparison in Communications department building (Outdoor).

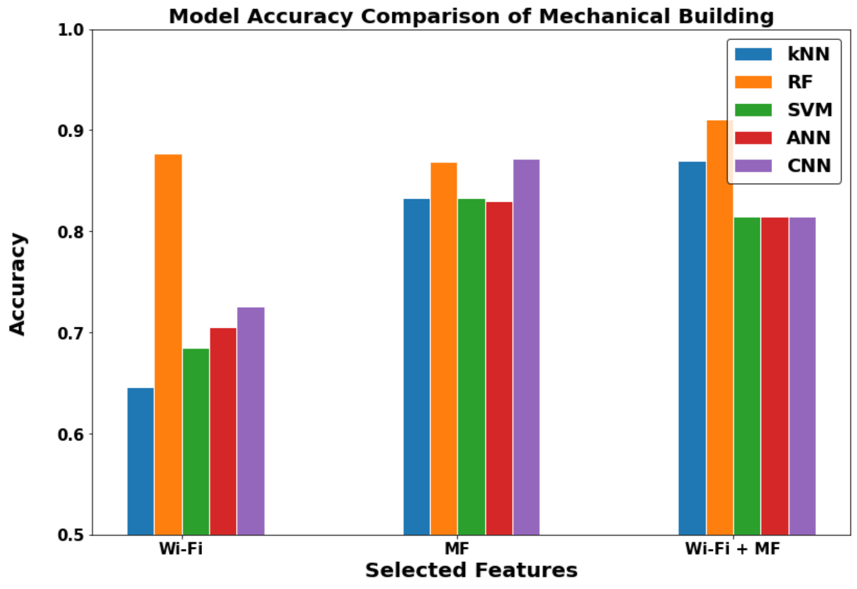


Figure 13. The models comparison in Mechanical department building (Outdoor).

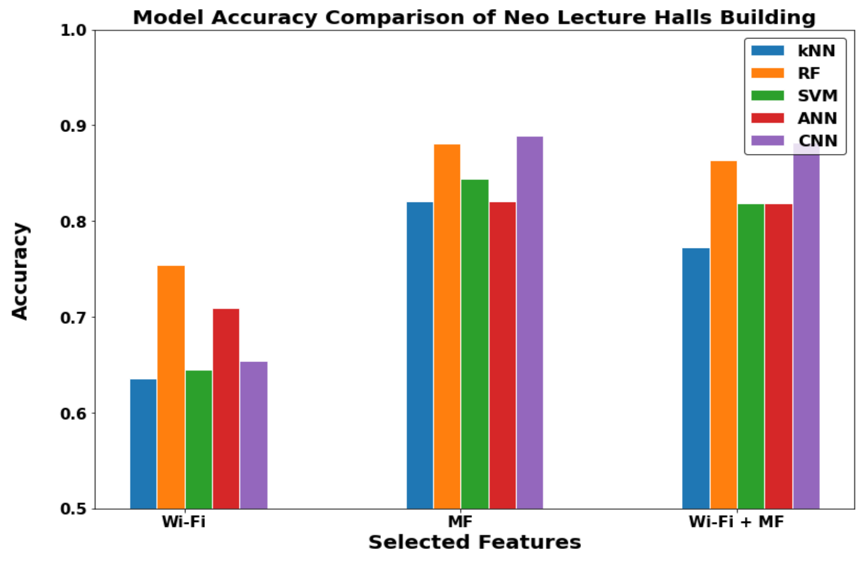


Figure 14. The models comparison in Neo Lecture Halls building (Outdoor).

After we have chosen the WF-ML for the indoor locations layer and MF-ML for the buildings layer, Figures 15 and 16 show the results of each location we have tested on. The performance of the WF-ML models in Figure 15 is above the 95% accuracy taking into consideration that each indoor location offers a challenge to the ML models. Each location contains at least two or three Wi-Fi access points that are detected by the mobile application. We can see that the best models are the Random Forest, ANN, and CNN as each one has an advantage to a specific case. The random forest is mostly the highest accuracy model as the classical machine learning models do not require large datasets. However, the deep learning methods require large datasets with at least 1000-1500 data rows or wireless metrics scans. But for our tests, we achieved good results with the CNN due to its ability of deep feature extraction. In the IPDSC Lab which contained only two Wi-Fi networks, the CNN model performed even better than the random forest model and this confirms our thoughts. Also, the models were robust to the close halls LH1 and LH2 as the detection models indicated most of the readings in their right geofences. On the other hand, Lecture Hall 26 showed the lowest accuracy using all the models due to the weak Wi-Fi coverage. The building's accuracy using MF-ML in Figure 19 has nearly the same conclusions as the discussed indoor layer with lower accuracy around 90% average. Tables 2, 3, 4, and 5 have more detailed results for the developed models in two indoor locations and two buildings with WF-ML and MF-ML models, respectively.

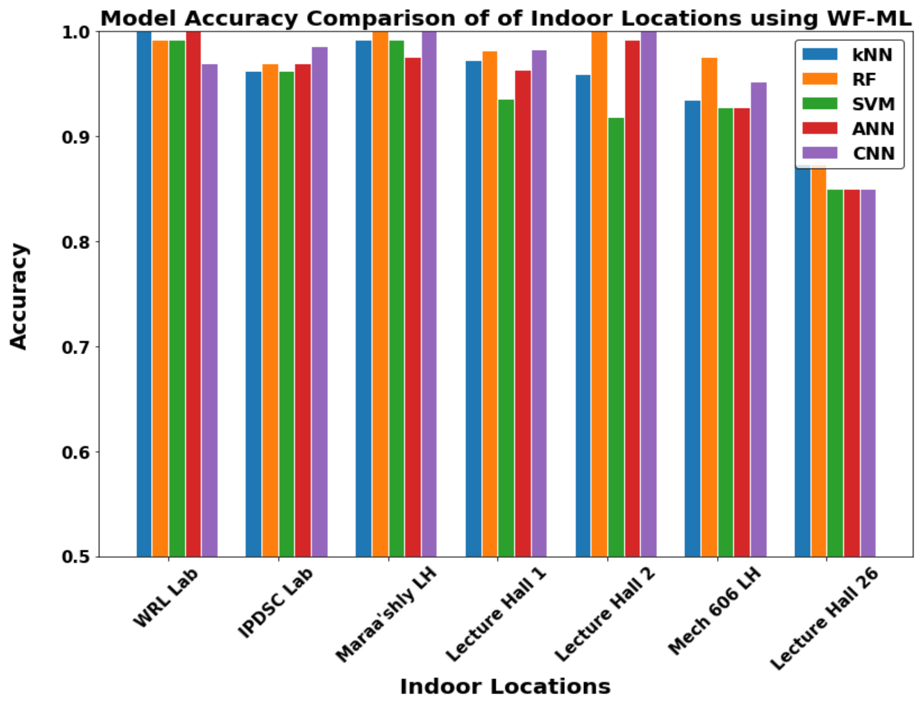


Figure 15. The models accuracy in all indoor geofences using WF-ML.

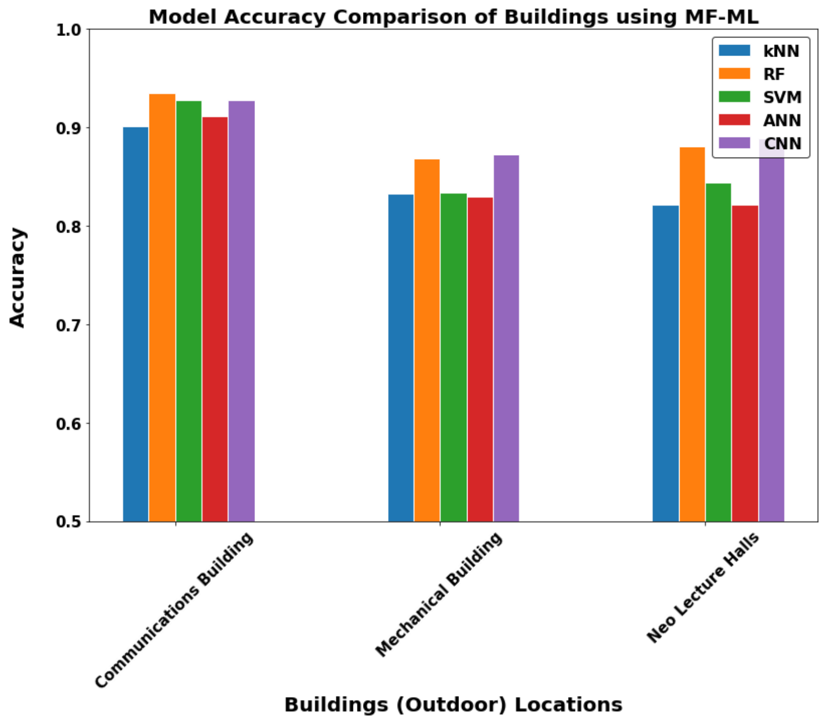


Figure 16. The models accuracy in all outdoor buildings geofences using MF-ML.

Table 2: WRL Lab Indoor WF-ML results.

Geofence	TSMF Data	ML Model	Accuracy	Precision	Recall	Kappa	F1	AUC
WRL Lab	WF-ML	kNN	1.0	1.0	1.0	1.0	1.0	1.0
		RF	0.99	1.0	0.98	0.98	0.99	1.0
		SVM	0.99	1.0	0.98	0.98	0.99	0.99
		ANN	1.0	1.0	1.0	1.0	1.0	1.0
		CNN	0.97	0.94	1.00	0.93	0.97	0.96

Table 3: Maraa'shly LH Indoor WF-ML results.

Geofence	TSMF Data	ML Model	Accuracy	Precision	Recall	Kappa	F1	AUC
Maraa'shly LH	WF-ML	kNN	0.99	1.0	0.98	0.98	0.99	1.0
		RF	1.0	1.0	1.0	1.0	1.0	1.0
		SVM	0.99	0.98	1.0	0.98	0.99	1.0
		ANN	0.97	0.97	0.98	0.94	0.98	0.99
		CNN	1.0	1.0	1.00	1.0	1.0	1.0

Table 4: Communications Department Building Outdoor MF-ML results.

Geofence	TSMF Data	ML Model	Accuracy	Precision	Recall	Kappa	F1	AUC
Communications Building	MF-ML	kNN	0.90	0.89	0.91	0.80	0.90	0.95
		RF	0.93	0.94	0.92	0.86	0.93	0.97
		SVM	0.92	0.89	0.96	0.85	0.93	0.96
		ANN	0.91	0.91	0.90	0.82	0.91	0.96
		CNN	0.92	0.90	0.96	0.85	0.93	0.92

Table 5: Neo LH Building Outdoor MF-ML results.

Geofence	TSMF Data	ML Model	Accuracy	Precision	Recall	Kappa	F1	AUC
Neo LH Building	MF-ML	kNN	0.63	0.68	0.45	0.26	0.54	0.73
		RF	0.75	0.84	0.60	0.50	0.70	0.83
		SVM	0.64	0.88	0.30	0.45	0.27	0.73
		ANN	0.70	0.86	0.47	0.40	0.60	0.73
		CNN	0.88	0.84	0.88	0.77	0.86	0.88

Figures 17 to 20 show a typical receiver operating characteristic (ROC) curve for the chosen models in some test locations. ROC can be considered as an evaluation metric for binary classification problems that shows the performance of a classification model at all classification thresholds. The higher the area under the curve (AUC), the better the performance of the model at distinguishing between the positive and negative classes. When AUC = 1, then the classifier can perfectly distinguish between all the positive and the negative class points correctly. If the AUC had been 0, then the classifier would be predicting all negatives as positives, and all positives as negatives. When $0.5 < AUC < 1$, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is because the classifier can detect more numbers of true positives and true negatives than false negatives and false positives. Therefore, the higher the AUC value for a classifier has the better ability to distinguish between positive and negative classes. When AUC = 0.5, the classifier is not able to distinguish between positive and negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.

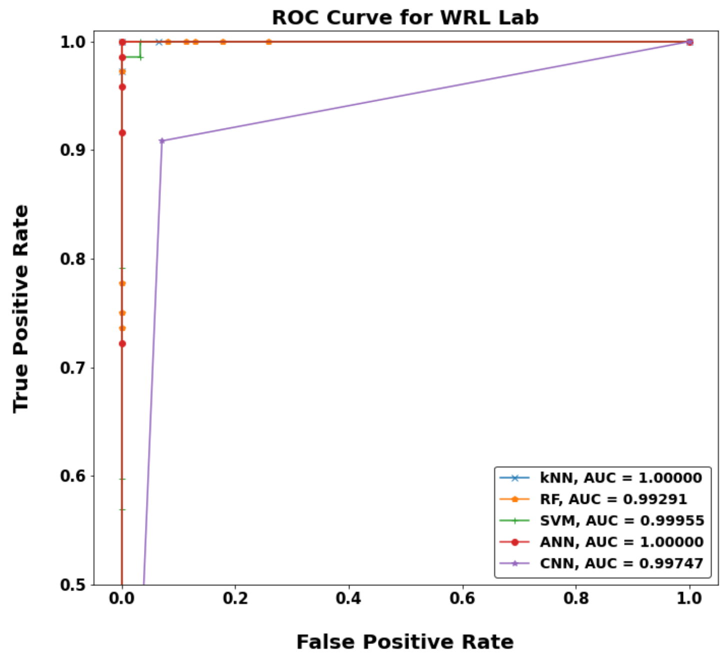


Figure 17. ROC curves for WF-ML models in WRL Lab.

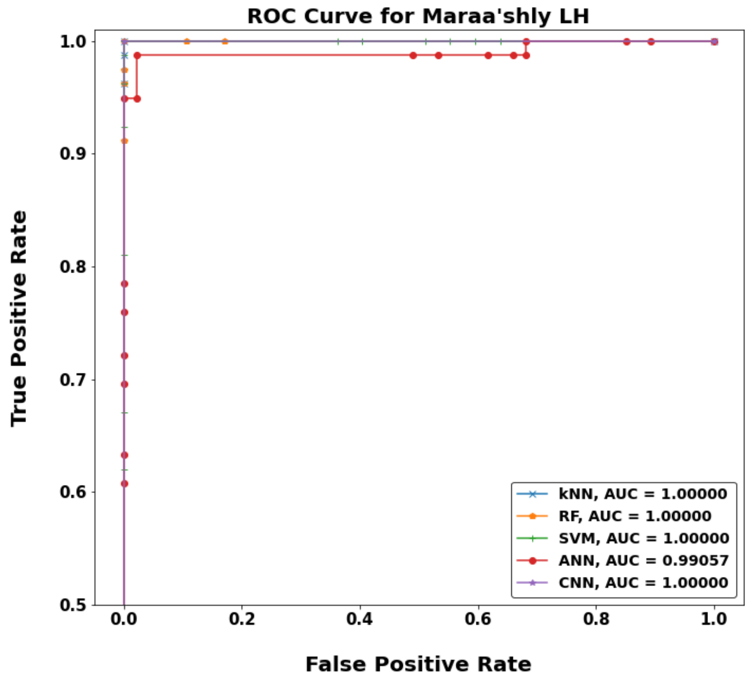


Figure 18. ROC curves for WF-ML models in Maraa’shly LH.

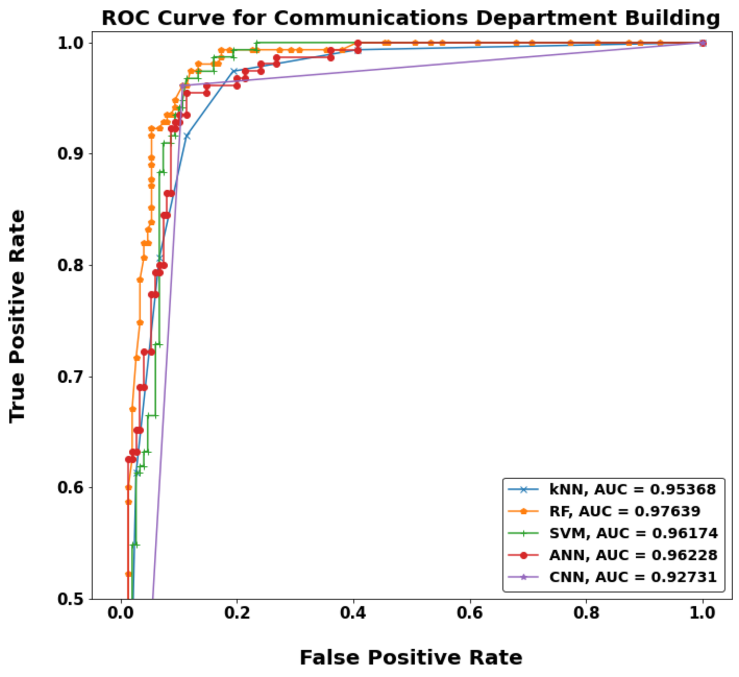


Figure 19. ROC curves for MF-ML models in Comm. Building.

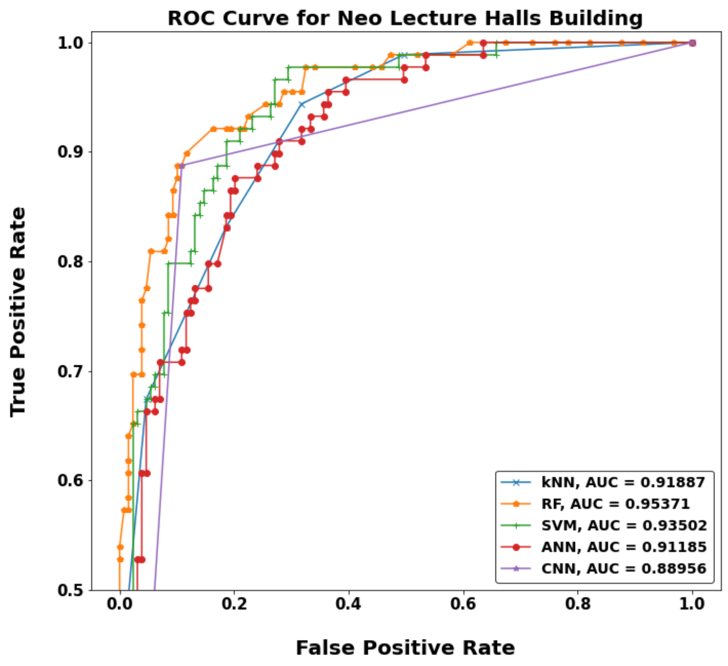


Figure 20. ROC curves for MF-ML models in Neo LH building.

Figures 21 and 22 show a confusion matrix that summarizes the prediction results of the whole system when it classifies the user's predicted geofence across all his associated geofences in the building layer and the indoor layer. The chosen ML deployed model was the ANN one due to the TensorFlow lite [38] support on the android platform in general and the less complexity and size comparing to the CNN

lite mode. Also, the classical models are not supported yet by TensorFlow lite. The accuracy of most of the chosen algorithms is really close with the change in accuracy ranges of 2% - 4% mostly which is not that big on the overall prediction performance of the deployed model on the mobile application. The problem was with the ML inference framework time and the optimization on the mobile application. During testing, the ANN and CNN models were easily tested with the TensorFlow lite library which is supported by the Flutter framework [39] that was used in the mobile application development. Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, and Windows. So, the developed application can be deployed to work on an iOS device with no problems with the current implementation.
The inference time was nearly 0.0016 sec with all of the ANN models and CNN models achieving 0.12 sec. Then the classical models were not supported by the TensorFlow lite library. The solution was to use another library called ml_algo [40] to test the models. The inference time of kNN got 10.18 sec as an example of the classical ML models. These numbers were expected as the TensorFlow lite library leverages specialist mobile accelerators and it provides on-device machine learning inference with low latency and a small binary size by decreasing the precision of the ML models. So, the choice was to use the ANN model as the difference in the accuracy is pretty low and it has the best inference time so far among the other algorithms.
Each geofence has taken as input 50 scans of the Wi-Fi and MF metrics to indicate the performance of the geofences classification. The buildings layer accuracy of the system is 76%. Meanwhile, the indoor layer accuracy of the system is 98.25%, which is highly acceptable taking into consideration that each location offers a significant challenge to the ML models.

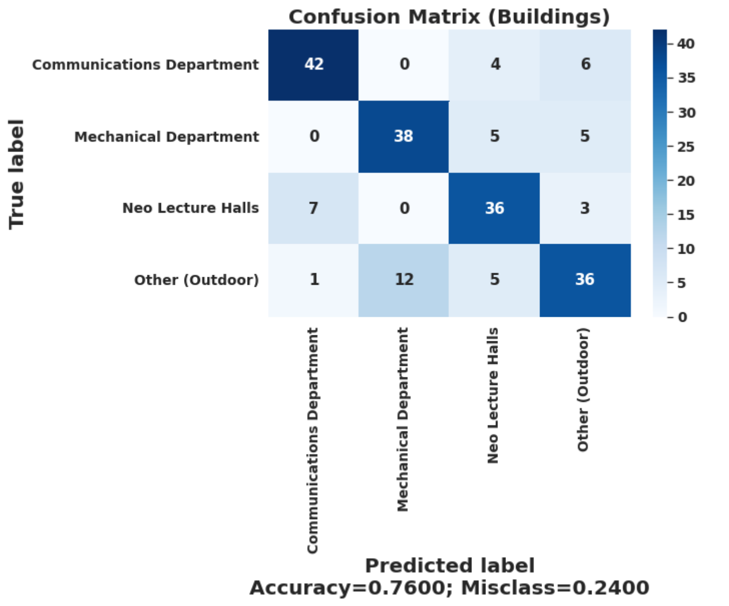
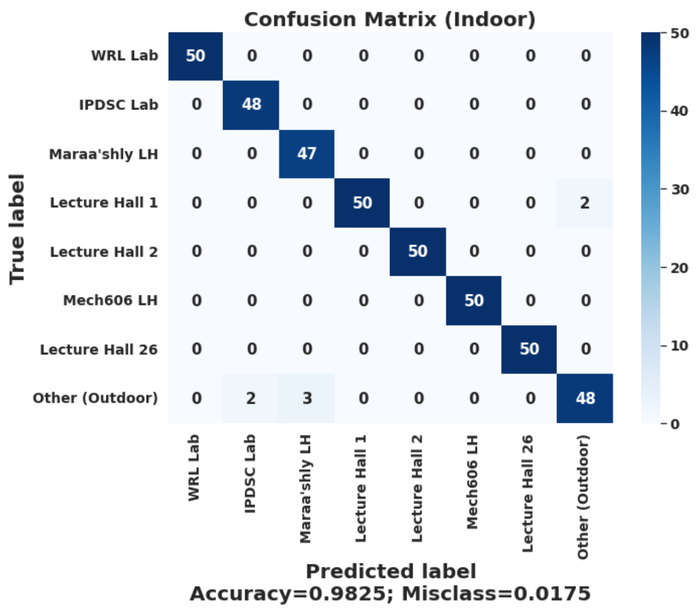


Figure 21. Confusion Matrix of the overall system performance with MF-ML models.



True \ Predicted	WRL Lab	IPDSC Lab	Maraa'shly LH	Lecture Hall 1	Lecture Hall 2	Mech606 LH	Lecture Hall 26	Other (Outdoor)
WRL Lab	50	0	0	0	0	0	0	0
IPDSC Lab	0	48	0	0	0	0	0	0
Maraa'shly LH	0	0	47	0	0	0	0	0
Lecture Hall 1	0	0	0	50	0	0	0	2
Lecture Hall 2	0	0	0	0	50	0	0	0
Mech606 LH	0	0	0	0	0	50	0	0
Lecture Hall 26	0	0	0	0	0	0	50	0
Other (Outdoor)	0	2	3	0	0	0	0	48

Figure 22. Confusion Matrix of the overall system performance with WF-ML models.

- ***LAGD Architecture***

From the results discussed along with the practical testing, The idea of the Layered-Architecture Geofence Division (LAGD) shown in Figure 23 can be introduced. The problem of using the Wi-Fi metrics only for the GDS generation is their absence in the outdoor environment. However, they provide good indication features in indoor environments. The opposite happens with the magnetic data, as they are significantly good for classifying the outdoor open area environments of the buildings from their insides. But in indoor environments, the general readings of magnetic data are similar. The proposed solution was using the idea of active geofencing (GPS-based service) as a first layer of detecting the user's current workplace or organization. Then, the magnetic data is used as classification features for the detection of buildings as the second layer of detection of the user by using MF-ML models. Then after the detection of the user's building location, the indoor rooms WF-ML models of this building are invoked for applying the third layer of detection.
The benefit of applying this technique is reducing the inference and processing time as we would invoke the geofences models only after entering the workplace. So, after the user is identified in a geofence or a hall by the WF-ML model, the LAGD architecture would only invoke the WF-ML models in the user's detected workplace and building. Then, it would only return to the previous layers if he weren't in any of the geofences of the formally detected building to recheck if the user was still in the building and the workplace. Thus, the GPS would not be used continuously but invoked only when needed to reduce power consumption and save battery life. Also, taking two types of features together would require more processing power of the

user's smartphone along with increasing the power consumption. In addition, it would simplify the GDS datasets collection and the ML models creation for each type of the geofences.

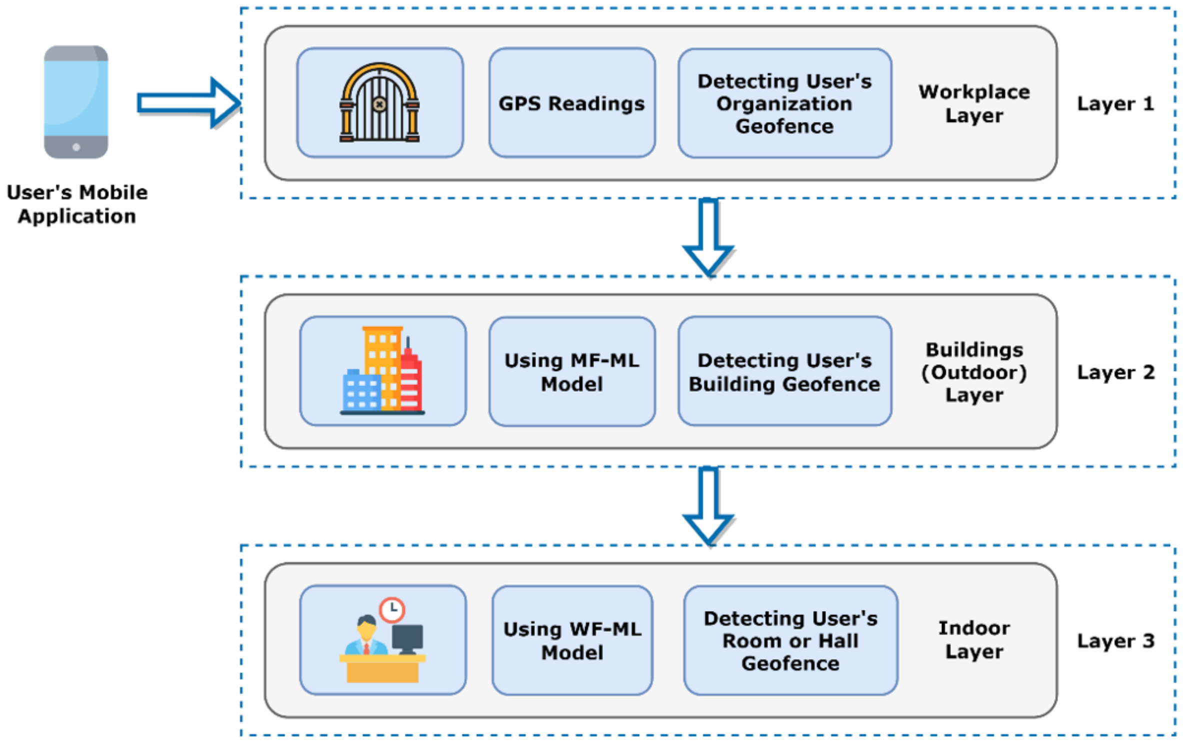

User's Mobile Application
GPS Readings
Detecting User's Organization Geofence
Workplace Layer
Layer 1
Using MF-ML Model
Detecting User's Building Geofence
Buildings (Outdoor) Layer
Layer 2
Using WF-ML Model
Detecting User's Room or Hall Geofence
Indoor Layer
Layer 3

Figure 23. LAGD Architecture.

References

[1] Ashraf, I., Hur, S., & Park, Y. (2020). Smartphone Sensor Based Indoor Positioning: Current Status, Opportunities, and Future Challenges. Electronics 2020, Vol. 9, Page 891, 9(6), 891. https://doi.org/10.3390/ELECTRONICS9060891E. Weiss and R. Alimi, "Low-Power and High-Sensitivity Magnetic Sensors and Systems," Artech House, 2018.

[2] Ashraf, I., Hur, S., & Park, Y. (2018). MagIO: Magnetic field strength based indoor-outdoor detection with a commercial smartphone. Micromachines, 9(10). https://doi.org/10.3390/MI9100534

[3] Matthew Gast, 802.11 wireless networks. Sebastopol [Calif.]: O'Reilly, 2005.

[4] E. Weiss and R. Alimi, "Low-Power and High-Sensitivity Magnetic Sensors and Systems," Artech House, 2018.

[5] Position sensors | Android Developers. (n.d.). [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_positionA

[6] B. Li, T. Gallagher, A. G. Dempster, and C. Rizos, "How feasible is the use of magnetic field alone for indoor positioning?," in Proc. International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp.1–9, 2012.

[7] "sklearn.preprocessing.standardscaler." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing. StandardScaler.html

[8] M. Nowicki and J. Wietrzykowski, "Low-effort place recognition with wifi fingerprints using deep learning," in International Conference Automation. Springer, 2017, pp. 575–584.

[9] 28. Batta Mahesh, "Machine Learning Algorithms-A Review", International Journal of Science and Research (IJSR), Volume 9, Issue 1, Jan. 2020, pp.381.386.

[10] 29. Sathya, R. and Abraham, A., 2013. Comparison of supervised and unsupervised learning algorithms for pattern classification. International Journal of Advanced Research in Artificial Intelligence, 2(2), pp.34-38.

[11] 30. Noble, W.S., 2006. What is a support vector machine?. Nature biotechnology, 24(12), pp.1565-1567.

[12] Peterson, L.E., 2009. K-nearest neighbor. Scholarpedia, 4(2), p.1883.
[13] Biau, G. and Scornet, E., 2016. A random forest guided tour. Test, 25(2), pp.197-227.

[14] A. J. Jinia et al., "An Artificial Neural Network System for Photon-Based Active Interrogation Applications," in IEEE Access, vol. 9, pp. 119871-119880, 2021, doi: 10.1109/ACCESS.2021.3108406.

[15] E. -S. M. El-Kenawy et al., "Advanced Meta-Heuristics, Convolutional Neural Networks, and Feature Selectors for Efficient COVID-19 X-Ray Chest Image Classification," in IEEE Access, vol. 9, pp. 36019-36037, 2021, doi: 10.1109/ACCESS.2021.3061058.

[16] sklearn.preprocessing.MinMaxScaler — scikit-learn 1.0.2 documentation. (n.d.). [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[17] Keras: the Python deep learning API. (n.d.). [Online]. Available: https://keras.io/

[18] Flutter.dev. 2022. Flutter - Build apps for any screen. [online] Available at: <https://flutter.dev/> [Accessed 19 June 2022].

[19] [Mostafa Abdelkhalek, Ahmed, Noor El-Deen Magdy Mohamed, Mostafa Mahmoud Abdelhakam, and Mahmoud Mohamed Elmesalawy. "Dynamic User-Centric Clustered Workplaces for COVID-19 Control Measures Based on Geofencing and Deep Learning." In 2022 7th International Conference on Machine Learning Technologies (ICMLT), pp. 230-236. 2022.

[20] TensorFlow. 2022. TensorFlow Lite | ML for Mobile and Edge Devices. [online] Available at: <https://www.tensorflow.org/lite> [Accessed 25 July 2022].

[21] Firebase. 2022. Firebase Cloud Messaging | Firebase Documentation. [online] Available at: <https://firebase.google.com/docs/cloud-messaging> [Accessed 25 July 2022].