Local Positioning with Bluetooth 5 for Smart Life

Kittathat	Jaruchaikul	60090013
Ricky	Ruiz	60090032
Sarin	Wanichwasin	60090034

Bachelor of Engineering in Software Engineering

Department of Computer Engineering

Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Academic Year 2020

ISBN: XXX-X-XX-XXXXXX-X

Thesis - Academic Year 2020

Bachelor of Engineering in Software Engineering

Department of Computer Engineering, Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Title: Local Positioning with Bluetooth 5 for Smart Life

Authors

1. Kittathat Jaruchaikul Student ID: 60090013
2. Ricky Ruiz Student ID: 60090032
3. Sarin Wanichwasin Student ID: 60090034

Approved for submission

……..

(Assistant Professor Dr. Visit Hirankitti)

Advisor

Date …19…./…7…../…2021…..

Acknowledgement

We would like to thank our advisor, Assistant Professor Dr. Visit Hirankitti, for supervising us throughout the project. This project cannot be completed without his support and guidance, as he has provided us with his test facilities and tools for us to work on and test our project since the beginning.

We also like to express our gratitude towards the seniors at Dr. Visit's laboratory who have also given us technical advice and support during the development of a major part of our system, and it would be impossible for us to complete the project without their help. Our colleagues offered help in fixing some bugs and testing our website. We really appreciated that.

Last but not least, we indebted to our parents, family members, and loved ones for their support throughout the four years of study at KMITL.

Abstract

At the end of 2019, the COVID-19 pandemic broke out and soon later has spread out rapidly all over the world. It has become a “new normal” for most of us to live with the pandemic, but we believe firmly that digital technology can help manage and control the pandemic such that one day we will be able to put it under control. The purpose of this thesis is therefore to investigate how we can control the pandemic using a local positioning technology based on Bluetooth 5.

For one to stay safe of the virus, one must always practice Social Distancing by keeping two meters away from another. When someone gets infected, the Contact Tracing is needed to inform people who have contacted the infected person. Timelines are to be drawn to trace past activities of this person, what activities he/she did and whom he/she met. These contacted people have different levels of risk of getting infected with the virus and they may need to quarantine themselves.

The purpose of this project is to solve these issues of Social Distancing and Contact Tracing using an indoor positioning system. This system consists of three positioning gateways and many Bluetooth tags. Each gateway is a small PC, i.e. a Raspberry Pi computer, equipped with a Bluetooth 5 chip and the tags are mobile phones equipped with Bluetooth low energy (BLE) chips. The gateways act as reference nodes which keep observing and scanning for the tags’ (Bluetooth) signal strength and with this signal strength received by the gateways, the gateways can use that information to locate positions of all the tags in a place. Once the location of a tag is calculated and identified, the gateways will send this position to a backend web server which in turn will pass it to a web browser running on the mobile phone which is the location of the tag itself.

In this project we develop the software operating on the gateways for location finding of tags and a web application on the backend web server to keep collecting and storing locations of tags sent from the gateways while performing Social Distancing and Contact Tracing when necessary, and also the client-side web application which allows the user to visualize tag's positions on a 2D map and a 3D map and at the same time interact with the local positioning system.

With our local positioning system, it can keep track of positions of people with tags automatically. The system can notify each user of how close he/she is to another user in order to practice proper social distancing. In addition, the system can notify of a risk of the user contacting an infected person directly or indirectly via contact tracing functionality where timelines can be automatically generated and shown on a 2D map.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

In the current situation, millions have suffered from COVID-19. It is not just a national issue, but it has become a global crisis of which many companies in all kinds of industries have attempted to find a solution. Therefore, as a software engineering student, we intend to develop a web application to track individual users for their location and exchange the information using an indoor positioning system (IPS). So that the users can be notified when a user with COVID-19 has been to the nearby places. After being notified, the user can then check for infection at the hospital. However, this problem can be optimized by using indoor positioning to allocate the users in the building for a more accurate way.

The idea is to use a Bluetooth signal to find the location of a user in the building and track where the user is; if the person is being closed to someone within less than 2 meters, the system will keep the record of the location of the two persons in a log. The Bluetooth equipment uses the RSSI (from Bluetooth Low Energy) to find the local position and calculate the location by utilizing three Bluetooth gateways.

Contact tracing is also another function to help the disease to be controlled quickly and for the government to check and clean up those places which might get contaminated with the virus. Furthermore, the technology for future development is the Bluetooth 5.1 technology because the new Bluetooth 5.1 provides a higher precision of the local positioning due to the angle of arrival (AoA) feature. But further research is required as the device support is still in the early stage, there are only a few companies who can supply merely a sample equipment for testing this Bluetooth 5.1 feature.

In addition, we are developing a full function of COVID-19 contact tracking. The system will show the map of the building where the user is and the places with COVID-19 contamination. Moreover, the system will also provide log history if the places you have visited might have COVID-19 contamination, the system will notify each user for the risk of contact with the disease. Users can update their COVID-19 status after they have the evidence of having COVID-19 infection and the system will notify other users who have been to the same places. Then the system can be set to the quarantine mode which will quarantine the user at home during the 14 days and gives the user the instruction how to do so.

1.2 Objectives

- Developing software and hardware for an indoor positioning system using Bluetooth technologies.
- Solving COVID-19 problems of social distancing, and contact tracing by using the indoor positioning system.
- Applying GIS (Geographic Information System) for referencing 2D positions in our indoor positioning system.

1.3 Scope of Work

- Researching and learning the concept of Bluetooth 4, 5.1 and RSSI.
- Developing both hardware and software for an indoor positioning system using Bluetooth technology.
- Build Bluetooth signal receivers by using three Raspberry pi 4 as the Bluetooth gateways.
- Raspberry pi 4s were used as a Bluetooth gateway for calculating a user current position and sending position to the SCADA application.
- Developing a web application for monitoring the real-time current position of the users.
- The web application can show all the previous locations of each user.
- The web application can show each user's timeline which displays all locations of activities of a day.
- Solving COVID-19 problems which are social distancing and contract tracing by using the indoor positioning system that are developed in this project
- Apply Geographic Information System (GIS) for analyze the distance between two tags by using PostGIS which is a spatial database extender for PostgreSQL

1.4 Thesis Structure

This thesis consists of seven chapters which are arranged as follows:

- Chapter 1 Introduction - refers to the motivation, objectives, scope of work, and thesis structure of this thesis.

- Chapter 2 Related work – proposes the Literature survey, various software that are relevant to this project, and comparison.

- Chapter 3 Background knowledge - explains the knowledge and technology necessary for the reader to understand the thesis.

- Chapter 4 Requirement analysis/ System Architecture and Design – presents the requirement of the system, the use case diagram and relevant system architecture diagram.

- Chapter 5 Software Development - explains the concepts, tools and techniques that are used in developing the project.

- Chapter 6 Results - refers to the results of software demonstration, which consists of the user interfaces of the software and real-world applications.

- Chapter 7 Conclusion and Future work - is the chapter that talks about the conclusion, future work and improvements to the project

Chapter 2
Related Works

This chapter primarily presents the comparison between existing software related to our indoor positioning systems, which includes Here, BT 5.1 Direction Finding, Thai Chana application, and COVID Alert mobile Application.

2.1 Here

Here Technology is a company which provides mapping location services to individuals and companies. At this moment, we focus on only one product of Here which is "HERE Indoor Positioning". HERE Indoor Positioning tells you where you are including a specified floor or room using Bluetooth 4.2 hardware. Moreover, Here can work well with an offline or with Wi-Fi availability or with Bluetooth beacons. In addition, Here will also provide indoor map applications or Indoor Radio Mapper App

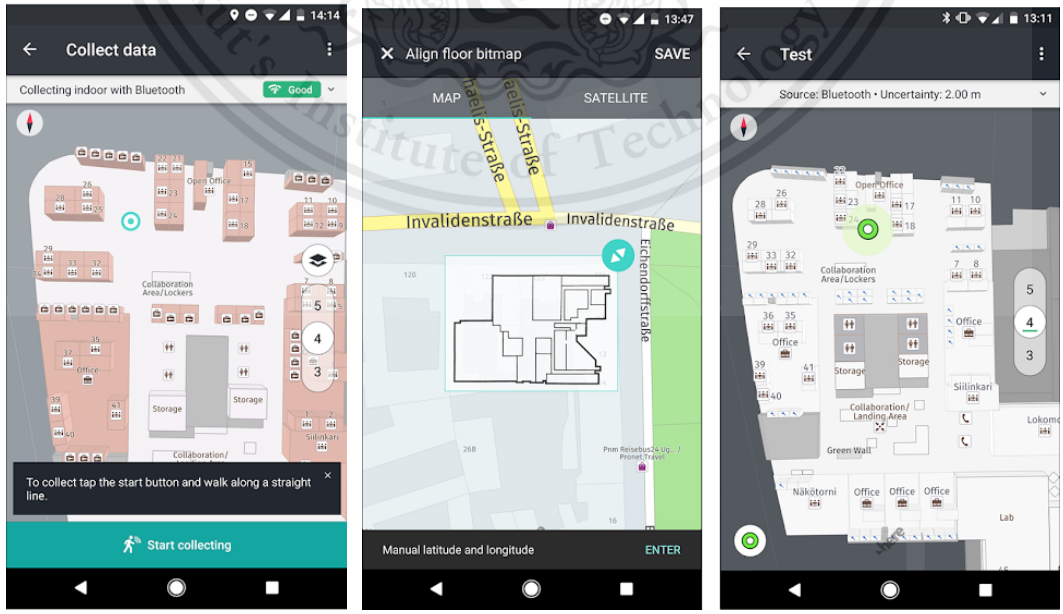

Figure 2.1: Screenshots from Indoor Radio Mapper App

To compare with our project, there are some similarities, such as User Interface (UI), Work in an offline environment and Bluetooth beacons (Bluetooth 4.0). However, there are some differences. For instance, in our project, we have a contact tracing feature which is used to notify the user when a user closes with COVID-19 user where the Here product does not have.

2.2 BT 5.1 Direction Finding

Texas Instruments is an American technology company in Dallas, Texas. Recently TI introduced a Bluetooth 5.1 developer kit which has the Direction-Finding feature and TI also provides a kit for testing out Bluetooth 5.1's direction finding feature which comes in the form of a transceiver board (a) and an antenna array (b). The Bluetooth transceiver is used as the receiver for Angle of Arrival (AoA) by integrating an antenna array module.

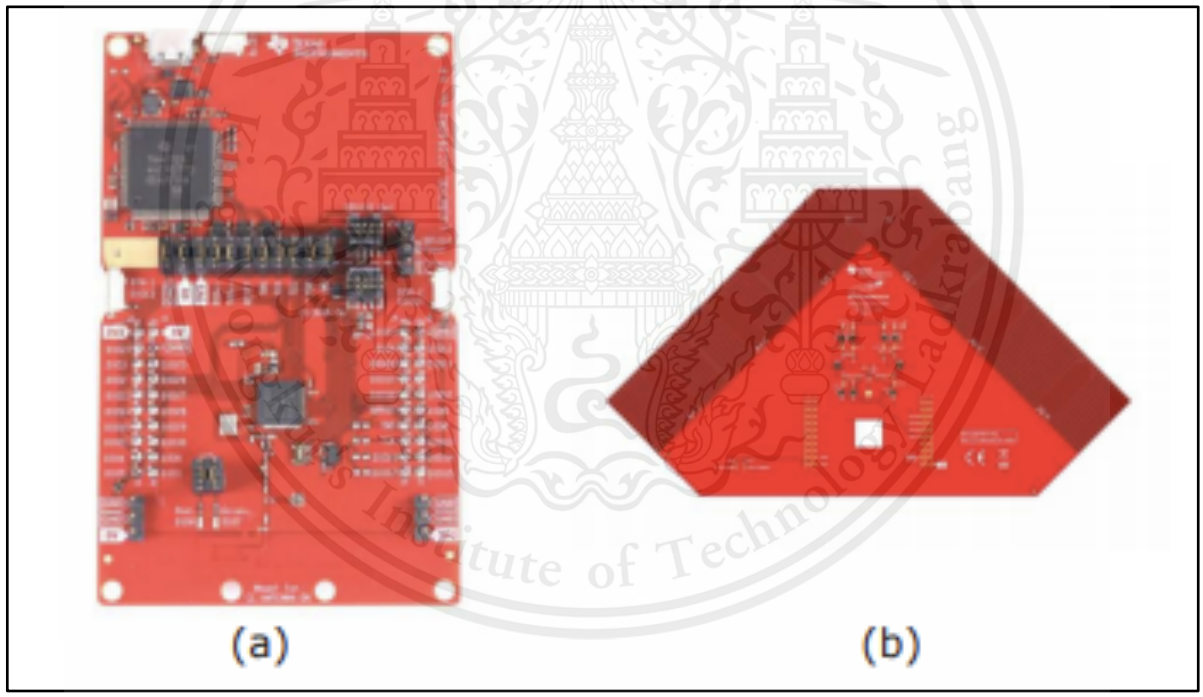

Figure 2.2: Texas Instruments BT5.1 Kit

The transceiver is equipped with an antenna array and the direction of the received signal. The transceiver then calculates the direction of the signal based on the phase difference of received signals from the transmitter equipped with the antenna array.

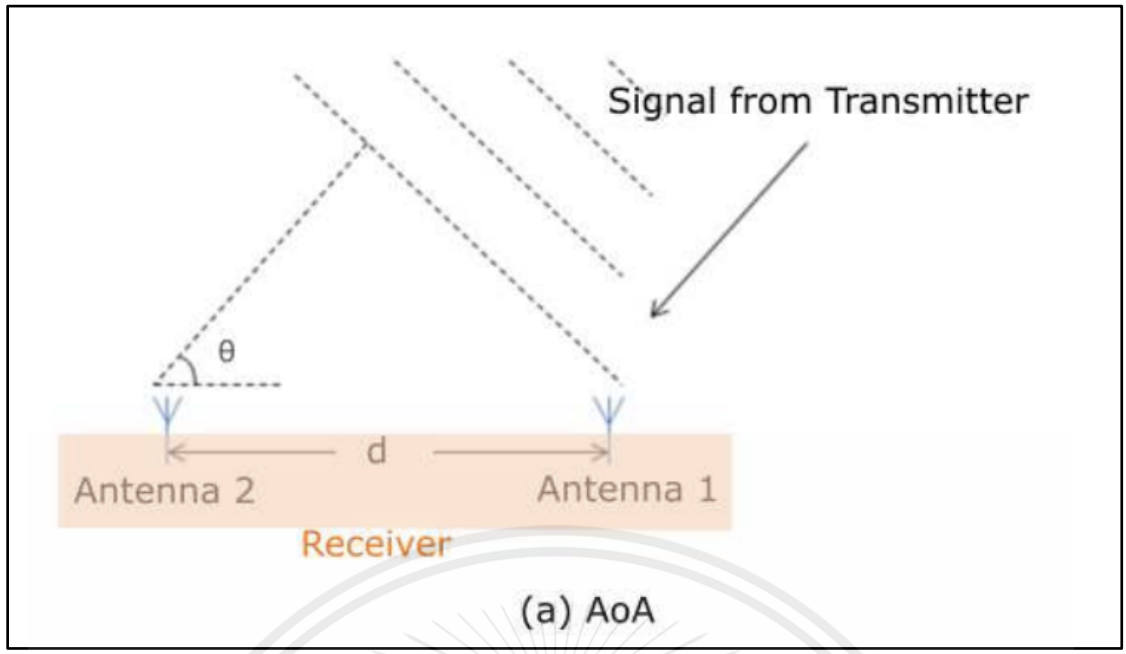

Figure 2.3: AoA Direction Finding

There are some similarities to our project, such as the use of Bluetooth technology in indoor positioning systems. However, the difference is in the Bluetooth version and calculation methods. Texas Instruments uses angles from AoA to do triangulation while ours uses Bluetooth Low Energy's RSSI in order to do trilateration.

2.3 Thai Chana (ไทยชนะ)

"ไทยชนะ" application. Thai Chana application is for the shopping stores or activity managers to register and watch over the population in that place. So the managers can control the population and limit the activity so as not to be over crowded that might cause the COVID-19 to spread out rapidly. Moreover, it lets the customers check beforehand if the places they decided to go are too crowded. However, we thought that we might have a way to improve some of the function of Thai Chana application by providing the users a notification and visualization to see where they have been to in the form of log history and notify them directly when they have been to the places reported as COVID-19 clusters and have high risk for visiting there.

Figure 2.4: Thai Chana Application

2.4 COVID Alert Mobile Application

COVID Alert is Canada's free COVID-19 exposure notification application. It can alert your contact before you have symptoms. This application uses Bluetooth technology to exchange random codes between your phone and nearby phones. Each day, the application will check a list of random codes from users who have informed the positive COVID-19 status through a one-time key. If you are tested positive for COVID-19, you need to enter a one-time key in the application to notify others. The application will notify you if a phone near you in the last 14 days from a user who tested positive COVID-19. When you are near someone else with the application, both phones exchange random codes every 5 minutes. The application will estimate how close people are around you by using the strength of the Bluetooth signal. Therefore, you must enable Bluetooth on your mobile phone all the time. The condition of exposure is recorded if you are within 2 meters of someone with COVID-19 for 15 minutes or longer. Furthermore, your mobile phone needs to connect to the internet to get a list of the random codes of app users who have reported a diagnosis. If it finds a matching code on your phone. Then, the application will notify you that you have contact with someone who is positive COVID-19 and the application will explain the next step which you should do. The purpose of this application is to reduce and limit the spread of COVID-19 by using Bluetooth technology which is on our smartphone.

To compare COVID Alert with our project, there are some similarities, such as using Bluetooth technology to find a nearby person and an application that runs on mobile phones. However, there is some difference. For example, COVID-19 Alert uses a one-time key system to notify other users.

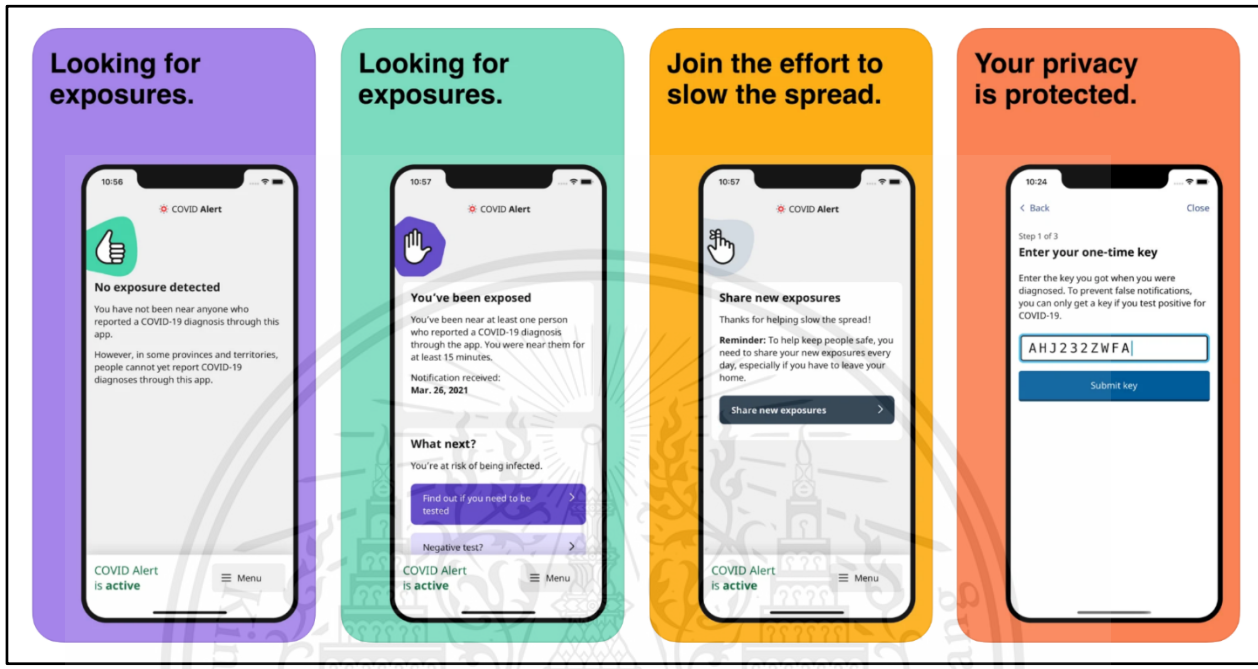

Figure 2.5 : Screenshot of COVID Alert Application

Chapter 3

Background Knowledge

In this chapter, relevant background knowledge of the project is described, which is divided into five sections.

3.1 Indoor Positioning System (IPS)

The Indoor Positioning System (IPS) is a network of devices used for locating people or objects locally where GPS and other satellite technologies lack precision or fails entirely, such as inside a multi-story building, an airport, a parking garage, and an underground location. Information from the system is fed to applications to make it useful. For example, IPS technologies enable a number of location-based solutions, including real-time location systems (RTLS), wayfinding, and inventory management.

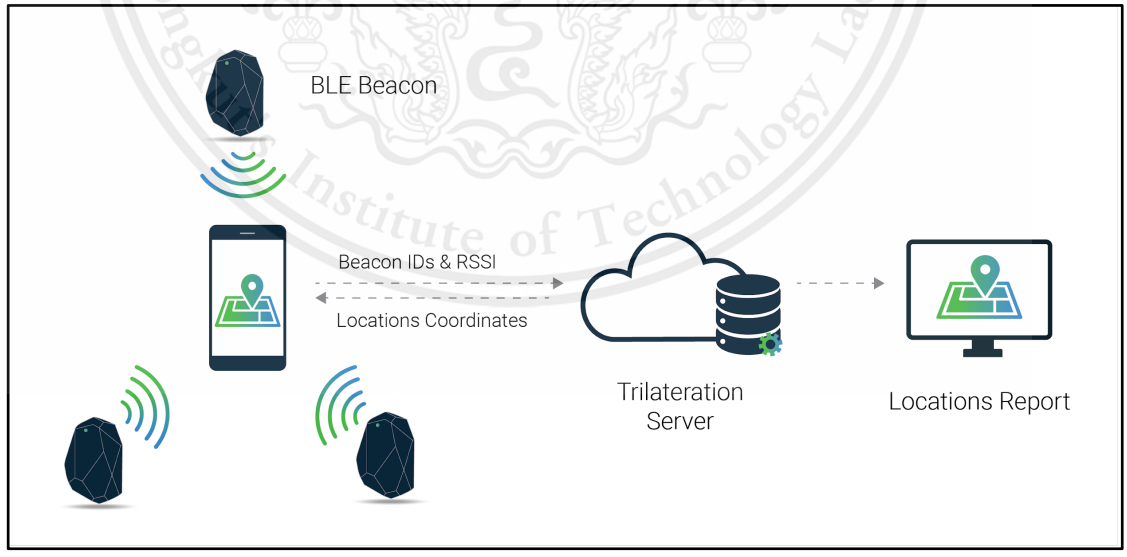

Figure 3.1: Using BLE Beacons for Location Finding

Essentially, indoor positioning is based on the same transmitter-receiver principle as GPS. It works by having transmitters send radio signals to a receiver, then the receiver interprets these signals and calculates its position within the given area using triangulation and trilateration.

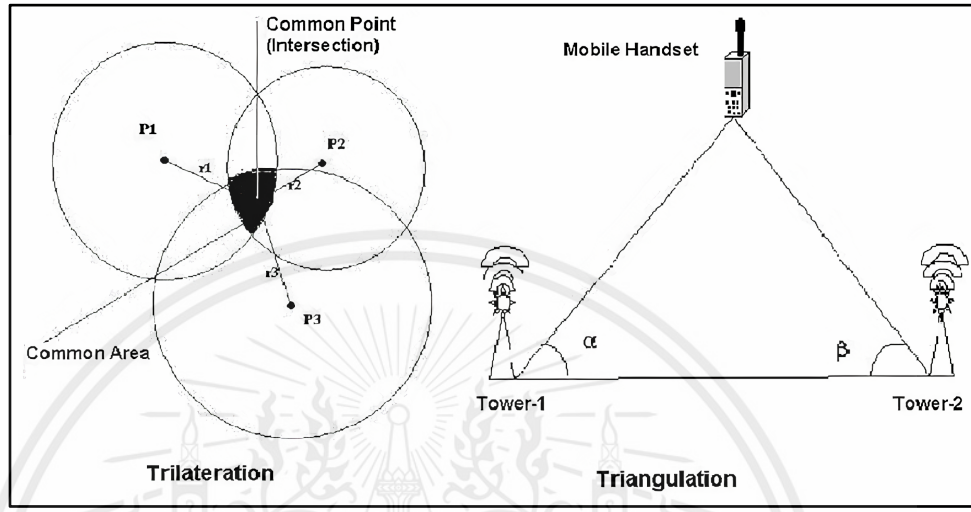

Figure 3.2: Trilateration and Triangulation

3.1.1 Triangulation

Triangulation, or angle-based method, is an estimation technique that uses a trigonometric approach to determine a moving object's location. Two or more fixed reference nodes are required for location estimation by receiving mobile signals (e.g. Bluetooth signals) sent from the signal-transmitting device attached to the target object. The signals sent from the target node form a geometric relationship as shown in Fig. 3.1, which can further be used to estimate the location of the target object.

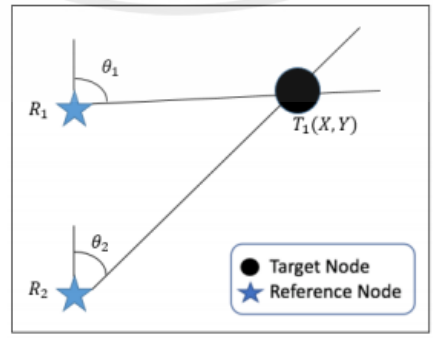

Figure 3.3: Triangulation

3.1.2 Trilateration

Trilateration, also called distance-based method or true-range multilateration, is an estimation technique similar to triangulation, but requires at least 3 fixed reference nodes. The only difference between the two methods is that trilateration uses the distances between the receiving and transmitting nodes to calculate the coordinate of the target object, instead of the angles. In trilateration, the distances between the transmitter (target object) and receivers can be viewed as the radius of many circles with centers at the receivers. The intersection of all the circles is the location of the target object where the coordinate of the target object (X, Y) can be estimated using the coordinates of the receivers ((X1, Y1), (X2, Y2), and (X3, Y3)) and the distances which are each circle's radius (r1, r2, and r3).

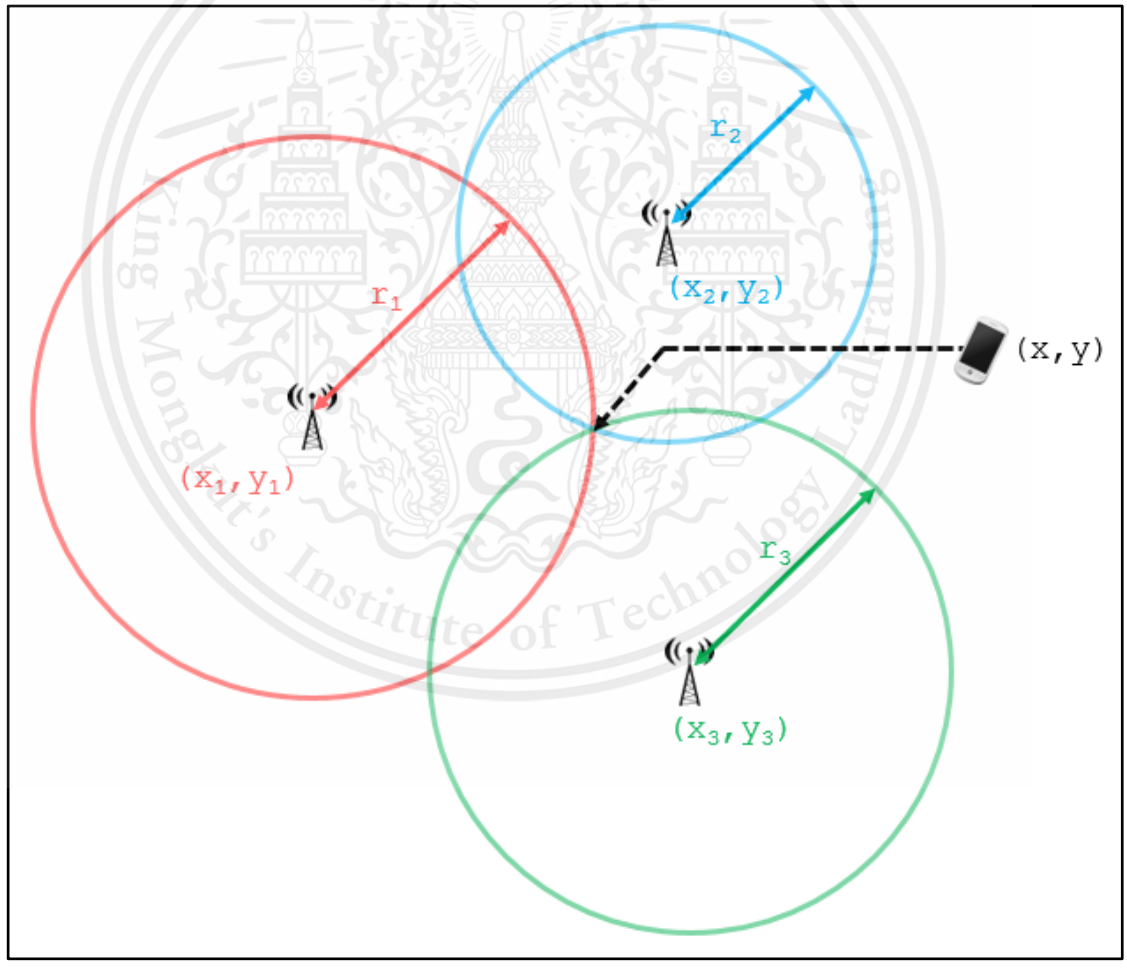

Figure 3.4: Trilateration with 3 Reference Nodes and 1 Target

The previous figure shows 3 circles, each with a radius *r* which represents all the possible locations of the target at a given distance (radius) of a cell tower. The aim of trilateration is to calculate the (x, y) coordinates of the intersection point of the three circles. Each circle is defined by the coordinates of its center e.g. (x1, y1) and its radius *r*.

First, each of the circles' equation can be written as so:

$$\begin{aligned}(x-x_1)^2+(y-y_1)^2&=r_1^2\\(x-x_2)^2+(y-y_2)^2&=r_2^2\\(x-x_3)^2+(y-y_3)^2&=r_3^2\end{aligned}$$

Then, we can expand the squares in each of the three equations:

$$\begin{aligned}x^2-2x_1x+x_1^2+y^2-2y_1y+y_1^2&=r_1^2\\x^2-2x_2x+x_2^2+y^2-2y_2y+y_2^2&=r_2^2\\x^2-2x_3x+x_3^2+y^2-2y_3y+y_3^2&=r_3^2\end{aligned}$$

Now, we subtract the second equation from the first:

$$(-2x_1+2x_2)x+(-2y_1+2y_2)y=r_1^2-r_2^2-x_1^2+x_2^2-y_1^2+y_2^2$$

And do the same for the third equation, subtracting it from the second:

$$(-2x_2+2x_3)x+(-2y_2+2y_3)y=r_2^2-r_3^2-x_2^2+x_3^2-y_2^2+y_3^2$$

To simplify, we will rewrite each coefficient expression using the alphabet A-F, resulting in the following system of 2 equations (on the left), and with some manipulation, we get our desired x and y values (on the right).

$$\begin{aligned}Ax+By&=C\\Dx+Ey&=F\end{aligned}$$

$$\begin{aligned}x&=\frac{CE-FB}{EA-BD}\\y&=\frac{CD-AF}{BD-AE}\end{aligned}$$

3.2 Bluetooth Profiles and Characteristics

3.2.1 Profiles

The BLE protocol operates on multiple layers with many profiles for various applications. The ones that are being used are the General Advertising Profile (GAP) and the General Attribute Profile (GATT). BLE devices let other devices know that they exist by advertising using GAP, which provides advertising packets that can contain a device name, some other information, and also a list of the services it provides. General Attribute Profile (GATT) is the layer that defines services and characteristics, similarly to the client-server structure. The target node is the GATT server (since it provides the services and characteristics such as RSSI), while the reference nodes are the GATT clients.

3.2.2 Received Signal Strength Indicator

The Received Signal Strength Indicator (RSSI) is a measure of the power level at the receiver. When a device scans for Bluetooth devices, the Bluetooth radio inside the device provides a measurement of the RSSI for each seen device. The signal strength depends on distance and Broadcasting Power value. Generally, the more negative the value (less than -100), the weaker the signal strength, while signals measuring -60 or above indicate a decent quality signal. RSSI is measured in dBm, or decibel-milliwatts.

The accuracy of the distance measurement depends on many factors such as the type of sending device used, the output power, the capability of the receiving device, obstacles and most importantly the distance of the beacon from the receiving device. The RSSI measurement is heavily affected by environmental interference, since the signal strength reduces significantly when it travels through obstacles.

We can calculate for the distance from the known RSSI measurements, but keep in mind that it is just an approximation since factors such as the line-of-sight blocking materials and interfering radio waves will cause inconsistency and inaccurate measurements. The formula is as follows:

$$\boldsymbol{Distance} = \mathbf{10}^{((\boldsymbol{Measured\ Power} - \boldsymbol{RSSI})/(\mathbf{10} * \boldsymbol{N}))}$$

where *Measured Power* is the expected RSSI at a distance of 1 meter from the transmitter to the receiver and *N* is the constant that depends on environmental interference (usually ranges from 2-4, 2 being less interference, 4 being heavily disrupted). Combined with RSSI, it allows us to estimate the distance between the target device and the beacon.

One must exercise caution when applying the formula though, since it's an exponential function so it will grow exponentially thus giving an inaccurate measure once the difference between the *RSSI* and the *Measured Power* becomes too great. Consider the graph below where the *Measured Power* is -69. If the *RSSI* is also -69, the distance of 1 meter is achieved, which is the desired value.

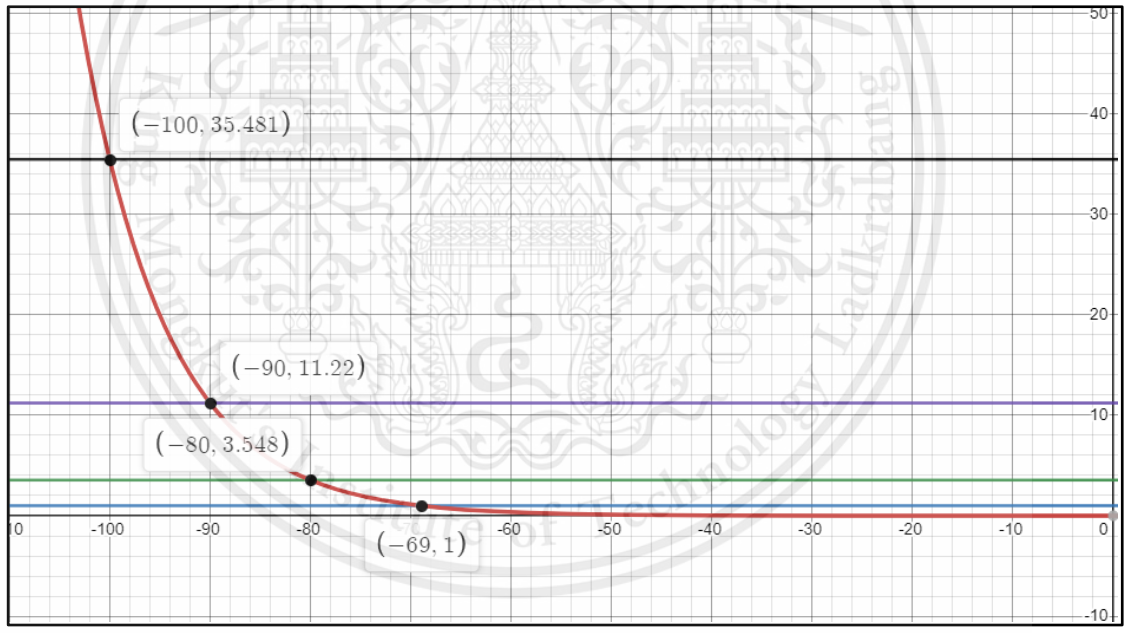

Figure 3.5: Exponential Growth from the Distance formula

However, the calculated values start growing exponentially starting from the difference of around 15-20, therefore the calculations from the *RSSI* measurements beyond that mark should be evaluated individually.

3.2.3 Bluetooth Device Address

Every Bluetooth device has a unique 48-bit address which is commonly abbreviated as BD_ADDR, or Bluetooth Device Address. This will usually be presented in the form of a 12-digit hexadecimal value, similarly to MAC addresses.

There are 2 main types of BD_ADDR: the public and random addresses. The Public address is a globally fixed address that needs to be registered with the IEEE (Institute of Electrical and Electronics Engineers), however registering for one isn't free. The following diagram shows the format of a Public Bluetooth address:

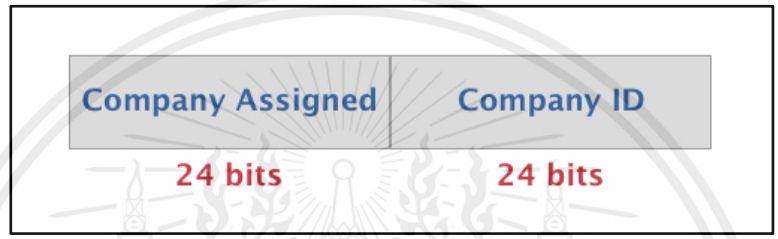

Figure 3.6: Public Bluetooth Address format

where the first 24 bits are assigned by your company and the IEEE will assign the unique Company ID.

The second type is the Random address. It is typically programmed into a device or randomly generated during bootup and doesn't need to be registered with the IEEE. A Random Static address looks something like this:

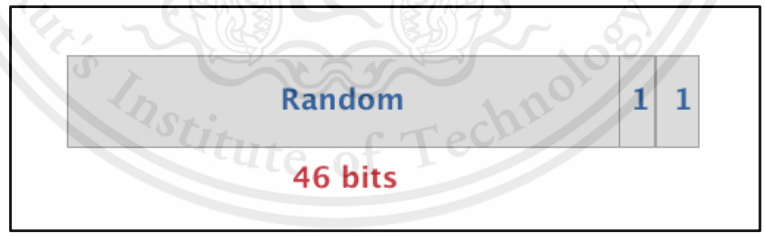

Figure 3.7: Random Static Address format

where the last 2 bits are fixed most significant bits while the remaining 46 are randomly generated by the program or manufacturer.

3.3 BLE (Bluetooth Low Energy) Beacons

BLE beacons are beacons that communicate over the Bluetooth Low Energy technology introduced in Bluetooth 4.0 in 2011. Beacon devices are small radio transmitters, strategically located to broadcast low energy Bluetooth signals in their given range. This range depends on hardware capability. A specialized beacon device can broadcast BLE signals up to 80 meters on average.

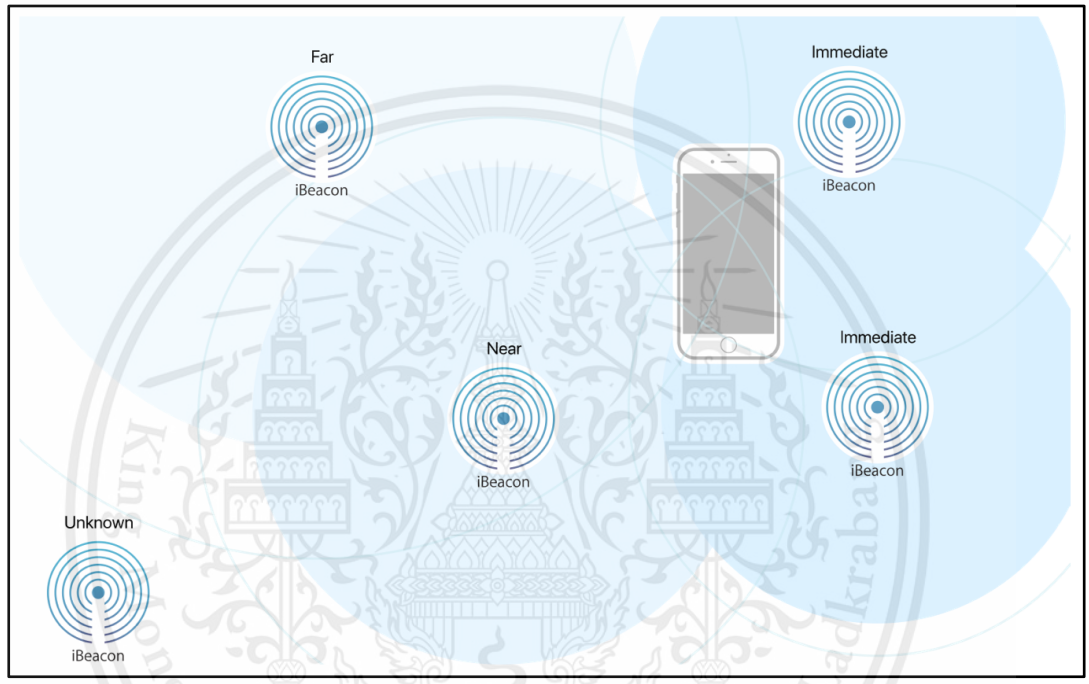

Figure 3.8: iBeacon distance approximation using RSSI

The BLE infrastructure works great for indoor positioning and navigation. A combination of three indoor beacons is already adequate for finding the accurate position of a smartphone. Indoor navigation using beacon technology offers turn-by-turn directions, marking of the important venues and indicating the recommended route. This is especially helpful for multistory stores, shopping malls and museums.

3.4 Kalman filter

Raw signal readings are inconsistent and to help in filtering out the most accurate result, we have chosen to use the Kalman Filter. The Kalman filter is an algorithm named after Rudolf E. Kálmán used for producing estimates of hidden variables based on inaccurate and uncertain measurements. Unlike what the name suggests, the Kalman filter is more like an estimator than a filter that essentially does 2 things:

- Predict the current state based on the last state, taking noise into account and
- Updates the current estimate by accounting for the current measurements

First let's look at what a Kalman filter consists of:

$$\hat{X}_k = K_k * Z_k + (1 - K_k) * \hat{X}_{k-1}$$

where:

$\hat{X}_k$ is the current estimate of *X* at state *k*

Z_k is the real measured value

K_k is the Kalman gain: relative weight given to the measurements and current state estimate that can be tuned to achieve desirable values

$\hat{X}_{k-1}$ is the estimate of the signal from the previous state

K is the state (e.g., time intervals, such as *k* =1 equals 1 second, *k*=2 equals 2 seconds, and so on)

These are the key components of a Kalman filter.

To actually do Kalman filtering, we first need to build a model. We can with the 2 Kalman filter equations:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$

$$z_k = Hx_k + v_k$$

where:

x_k is the signal value, or linear combination of its previous value plus a control signal k and a process noise

u_k is the control signal

w_{k-1} is mean of the noise

v_k is the standard deviation of the noise

A, B, H are general form matrices

First equation says that each x_k (signal values) may be evaluated by using a linear stochastic equation (the first one). Any x_k is a linear combination of its previous value plus a control signal k and a process noise.

The second equation says that any measurement value that we aren't 100% sure of its accuracy is a linear combination of the signal value and the measurement noise. They are both considered to be Gaussian. The process noise and measurement noise are statistically independent.

The entities A, B and H are in general form matrices. But in most signal processing problems, we use models such that these entities are only numeric values. While these values may change between states, we will also assume that they're constant, since they do not fluctuate most of the time.

If we are pretty sure that our system fits into this model, the only thing left is to estimate the mean and standard deviation of the noise functions w_{k-1} and v_k. We know that in real life, no signal is pure Gaussian, but we may assume it with some approximation. This is not really a problem, because we'll see that the Kalman Filtering Algorithm tries to converge into correct estimations, even if the Gaussian noise parameters are poorly estimated. With Kalman, the better you estimate the noise parameters, the better estimates you get.

Chapter 4
Requirement Analysis / System Architecture / Design

4.1 Requirement Analysis

Our system has 3 parts namely, the hardware, backend, and frontend. The hardware scans for the Bluetooth tag, calculates its location, then sends it to the backend. The backend then analyses the information, then sends it to the frontend for display.

4.1.1 Functional Requirement

1. The system must be able to find the location of the Bluetooth tag.
2. The system must be able to show the current position of the Bluetooth tag.
3. The system must be able to keep the Bluetooth tag's location and timestamp throughout the day.
4. The system must be able to conduct contact tracing with the available information provided by the tag.
5. The programming language for the hardware part must be the Python programming language.
6. The programming language for visualization or website must be JavaScript, CSS, HTML and JSX (JavaScript XML).
7. The web application must be made on the React Web framework.
8. The web application must be able to show a timeline for each visited location and timestamp

4.1.2 Non-Functional Requirement

1. (Usability) The objects in the floor plan should be easy to identify.
2. (Usability) The floor plan should be clear and easy to pan.
3. (Performance) The server must be able to receive information from the Bluetooth tag smoothly and efficiently and produce fine results for displaying the current position (floor plan) of the user.
4. (Performance) The current position of the Bluetooth tag should provide consistent and accurate results.
5. (Performance) The board needs to include Bluetooth beacon technology.
6. (Performance) The board must be able to connect with the WIFI.
7. (Supportability) The indoor-map application must be able to support web and mobile access.

4.2 Use Case Diagram

For the use case diagram, we grouped related services (use cases) according to the web page they are located.

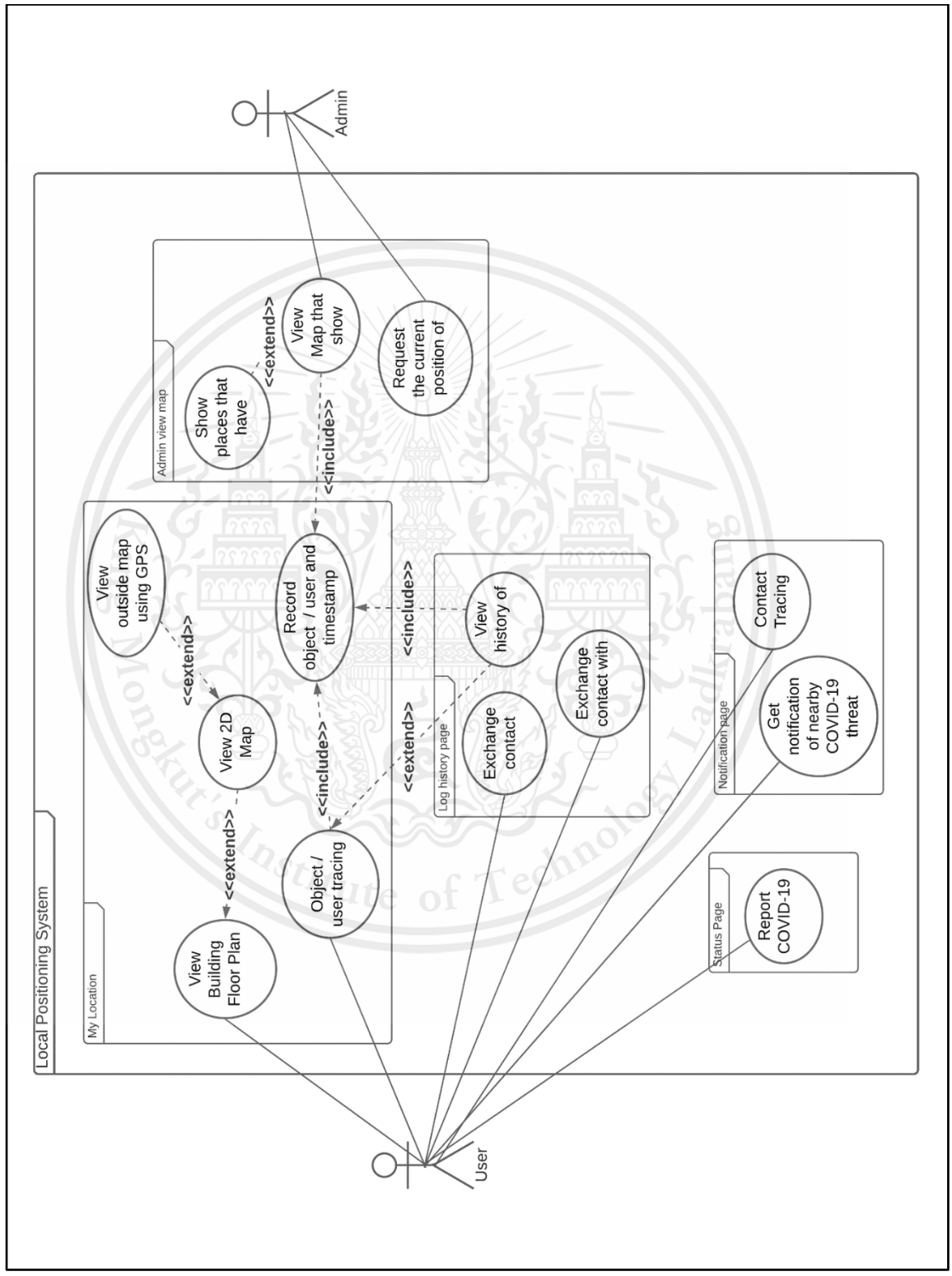

Figure 4.1: Use Case Diagram of our System

4.3 Use Case Scenarios

As I have mentioned before, the use cases were group into the following web pages.

4.3.1 Mylocation Page

In the Mylocation page, users will allow users to see the indoor building map for each floor and be able to select which room you want to get in. Also, when users go out of the building, they can view an outdoor map that indicates where that particular user has been to.

Use Case: View Building Floor plan

Primary Actor: User (Authorized user)

Scenario: After user access to the website, the user will straight away go to the user's current location. Then the user can also choose the floor and room to see.

Use Case: View 2D Map

Primary Actor: User (Authorized user)

Scenario: After the user selects the room and floor, then the website will display the room in 2D or 3D.

Use Case: Object user tracing

Primary Actor: User (Authorized user)

Scenario: The system will calculate Bluetooth signal strength to find the location of each user and display it on the room map.

Use Case: Record the user timeline

Primary Actor: User (Authorized user)

Scenario: After the system calculates the user's location, then the system will record the data to the database for displaying and analyzing data later on.

Use Case: Show Bluetooth signal strength

Primary Actor: User (Authorized user)

Scenario: The system will calculate the signal strength to find the user's location, then the user can see if his signal is good or bad.

Use Case: View Outdoor Map

Primary Actor: User (Authorized user)

Scenario: In Mylocation page, the user will see the outdoor button to view the outdoor map and the user's timeline. The user can also see the detail on each timeline for when and where he has been to

4.3.2 Log History Page

For the Log History page, users can view their history of places, where they have been to which consist of tag owner name, timestamp, x_coord and y_coord in the room, room name, floor, location.

Use Case: Exchange contact

Primary Actor: User (Authorized user)

Scenario: This page is where the contact exchange is handled, as it is where we call the data from the database, then the data will be used in this page. The system will check and exchange the data with each user for any contact.

Use Case: View History

Primary Actor: User (Authorized user)

Scenario: As this page is where the data is being displayed, the user can view the history of his data which contains where he has been and when.

4.3.3 Notification Page

The Notification page will show the activity of the users for the distance, the users are closing to someone. The closer they meet, the riskier they take. For example, we indicate the risk by colors. Red for high risk, orange for normal risk, yellow for low risk, and green for very very low risk. And when the user is indicated any color rather than green, then they have to take the procedure and get into quarantine.

Use Case: Get notification of nearby COVID-19 threat

Primary Actor: User (Authorized user)

Scenario: User will be notified by the system when he or she has connected a COVID-19 user

Use Case: Contact Tracing

Primary Actor: User (Authorized user)

Scenario: The system will provide a tracing back function for the admin to trace back for who has been contacted with COVID-19. However, the user needs to give the information to the data and pass it to the admin page for tracking purposes.

4.3.4 Profile Page

The Profile page is where the user can see their information and edit them also for some information, it will be displayed in the sidebar for a simple look at who is currently logged in.

Use Case: Report COVID-19 status

Primary Actor: User (Authorized user)

Scenario: Users can view their profile in this page and this user can also report their COVID-19 status in this page. However, the user needs to have the COVID-19 proof to attach on the system before the admin can confirm that the person is really infected.

4.3.5 Admin Page

In the Admin page, admin of the website can sign in and track each user for where they have been visited.

Use Case: View admin map

Primary Actor: Admin (Authorized admin)

Scenario: Admin map is the map that is similar to the user's map but admin can see every user's location. Admin can see only the outdoor map for where the users have been to.

Use Case: Show places that have COVID-19 timeline

Primary Actor: Admin (Authorized admin)

Scenario: IN admin page the admin can view the timeline of all users to trace back where the COVID-19 started.

Use Case: Request the current position of each user

Primary Actor: Admin (Authorized admin)

Scenario: In the admin page, the admin can request for a specific timeline of each user.

4.4 Class Diagram

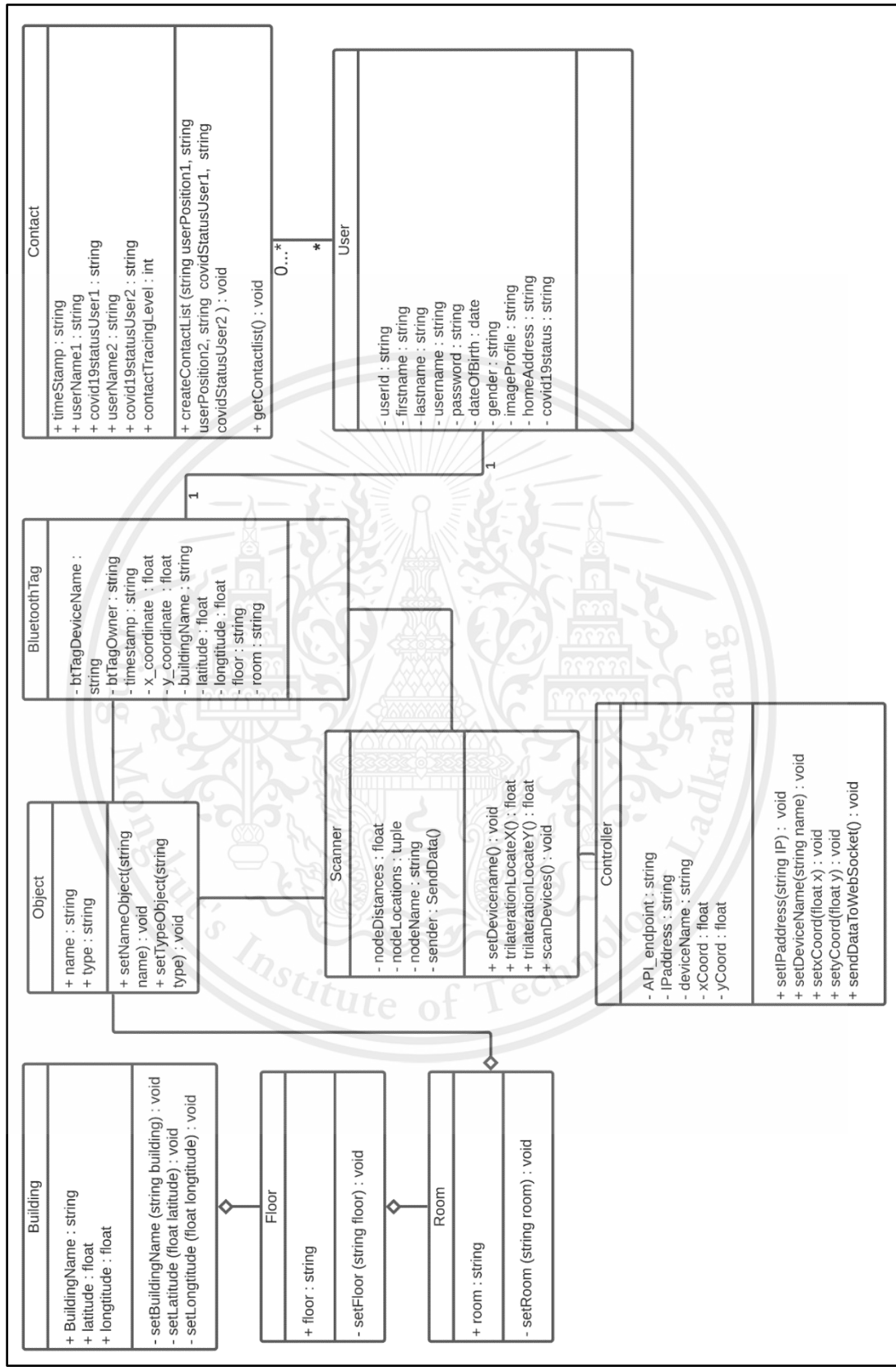

Figure 4.2: Class Diagram

From left to right:

Class Building is used for mapping the current building's location onto the real-world map. It is used to relate where our application's building is compared to the actual building.

Class Floor is the name of the floor in the building, e.g., floor 2A, 2B, etc.

Class Room is the name of the room on the floor of said building.

Class Object is used for keeping the objects present in the room.

Class Scanner is the class that represents the reference nodes, which are the Raspberry Pis in the real world. The Scanner would scan for Bluetooth tags.

Class Controller is used to communicate with the backend. Its main function is to send the location information to the backend.

Class BluetoothTag is the class that represents the target nodes, which are the Bluetooth tags in the real world. These tags would contain its device and location information that will be requested by the Scanner.

Class User refers to the users of our system. Each user would have a Bluetooth tag associated with them that's used to identify their location.

Class Contact keeps track of when 2 users come into contact with one another. A contact is created when a user in our system comes into close contact with another user in the same general location.

4.5 System Architecture

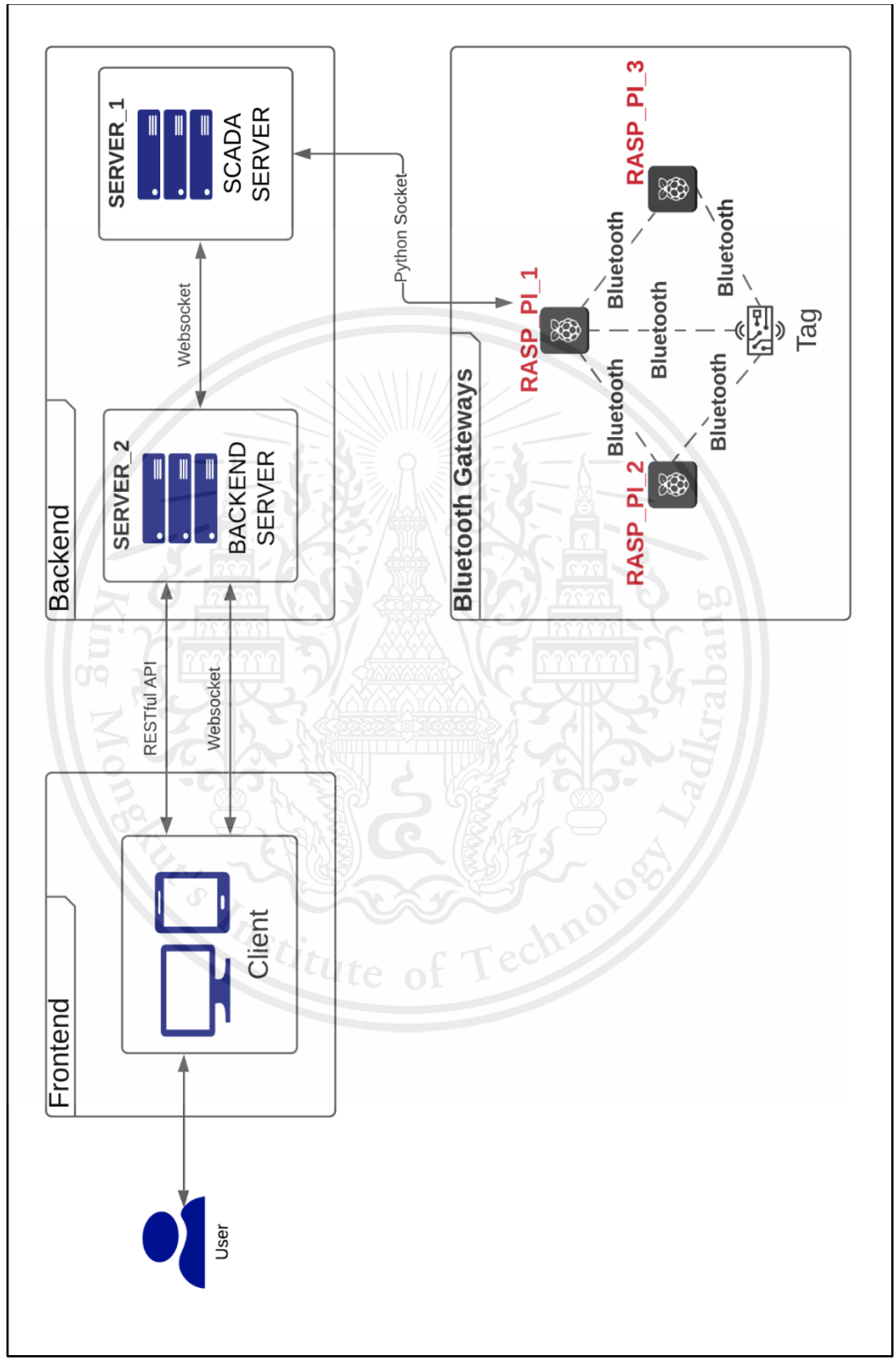

Figure 4.3: System Architecture

For system architecture, there are three main parts of the system which are frontend, backend and hardware (Bluetooth gateways).

The hardware part consists of three Raspberry pi (Bluetooth receivers) and Bluetooth tag (Bluetooth transmitter) can be either mobile phone or device which has Bluetooth module.

The backend part consists of the first server (SCADA server) and second server (Django server). SCADA server will get data from raspberry pi no.1 via python socket worker and send the data to Django server. Django server will get all location data from SCADA server via WebSocket and store all location data on PostgreSQL database with PostGIS extension. The tasks of Django server are analysis and sending all necessary data to the frontend part.

For the frontend part, the frontend part will get all data from backend side via RESTful API (User information, login-session information, log history information) and WebSocket (Realtime location data of Bluetooth tag and Real Time contact tracing data in notification page).

4.6 Developer Tools Diagram

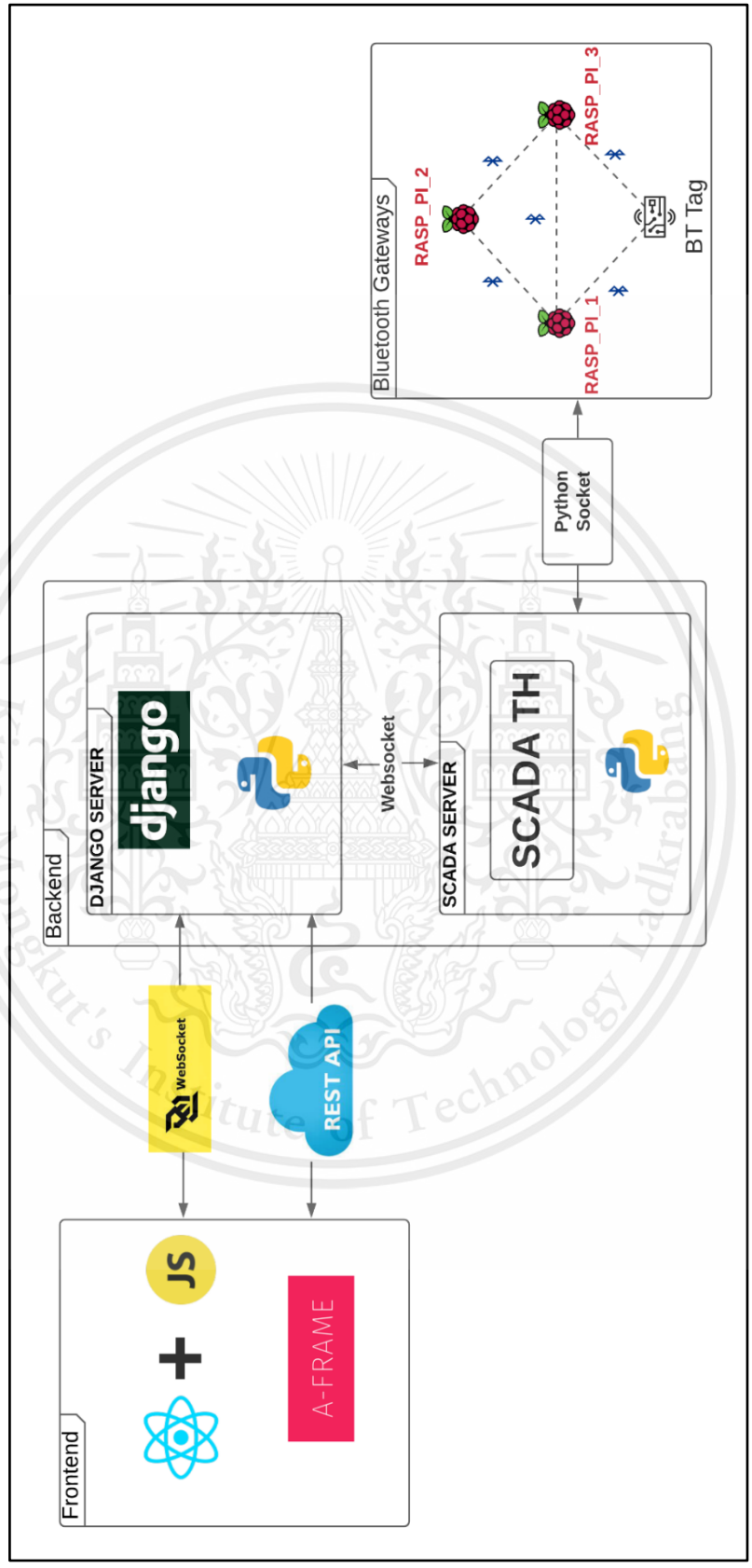

Figure 4.4: Developer Tools Diagram

We shall explain about our developer tool diagram. In the hardware part, we use the Python programming language with the Bluepy library to implement all necessary applications on the hardware side. In the backend side, Python programming language is used to implement everything in the backend side. We also use the Django web framework and PostgreSQL database with PostGIS extension to create a second server. These are reasons why we select and use the Django web framework in our project.

Firstly, we studied and learnt the Django web framework last year. Second, the Django web framework takes a little time to implement the server system. Third, the Django web framework has many supported libraries. In the frontend part, we use React framework with A-frame library to create 3D models which are user and room to virtual current location of users who use our system. JavaScript programming language is used in every part of the frontend section. Both REST API protocol and WebSocket protocol used to transfer the data between frontend and backend.

Chapter 5
System Development

5.1 Hardware Development

The Raspberry Pis are using a library called Bluepy to interface with their Bluetooth sensors. We will be using Bluetooth Low Energy for measuring the RSSI of each tag. To “turn on” Bluetooth Low Energy on each tag, we need to create a service which would be advertised, and when that is achieved, we have effectively turned on BLE. There are 3 tags in total: 2 smartphones and 1 smart band. The smart band is natively BLE, while the smartphones need an app to call the BLE. The app for doing this is made with the framework Apache Cordova using the cordova-plugin-ble-peripheral.

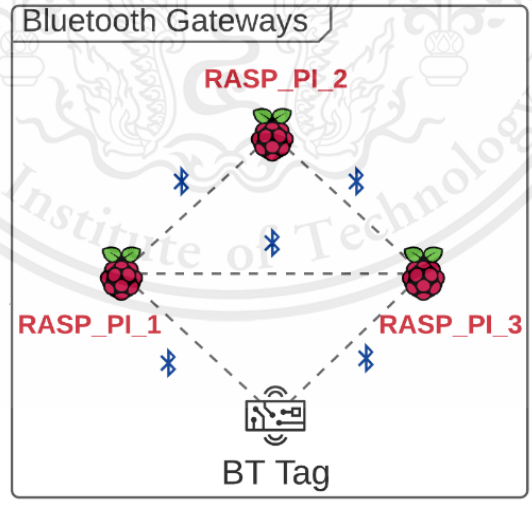

Figure 5.1: Hardware Diagram

5.1.1 Reference Nodes' Setup

In our experiment, we are going to conduct trilateration in order to determine each peripheral's location. To do so, we require 3 (or more, optionally) reference nodes. 3 Raspberry Pi 4 Model B boards are used as the reference nodes in this experiment. Each RPi is 3 meters apart from each other, forming the x-axis and y-axis of a 3x3 xy-plane.

Figure 5.2: RPi's Real World Locations

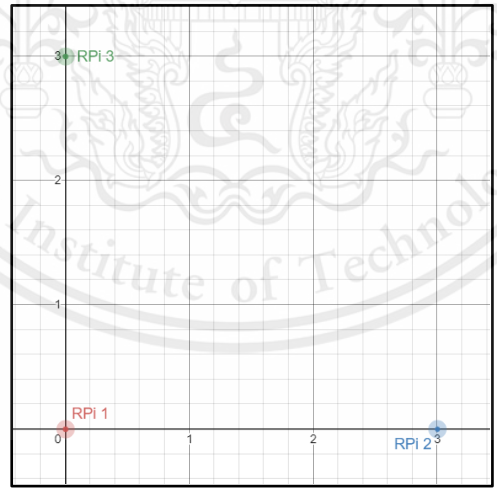

Figure 5.3: RPi's Locations on the XY-plane

5.1.2 Device Calibration

Back in section 3.2.2, we have shown the formula for converting from RSSI to distance:

$$\boldsymbol{Distance} = \mathbf{10}^{((\boldsymbol{Measured\ Power} - \boldsymbol{RSSI})/(\mathbf{10} * \boldsymbol{N}))}$$

where an unknown variable *Measured Power* is left to figure out. To get this variable, we need to measure the RSSI of a peripheral that is placed 1 meter away from a reference node.

Figure 5.4: Peripheral Measured Power Calibration - (OnePlus) 3T

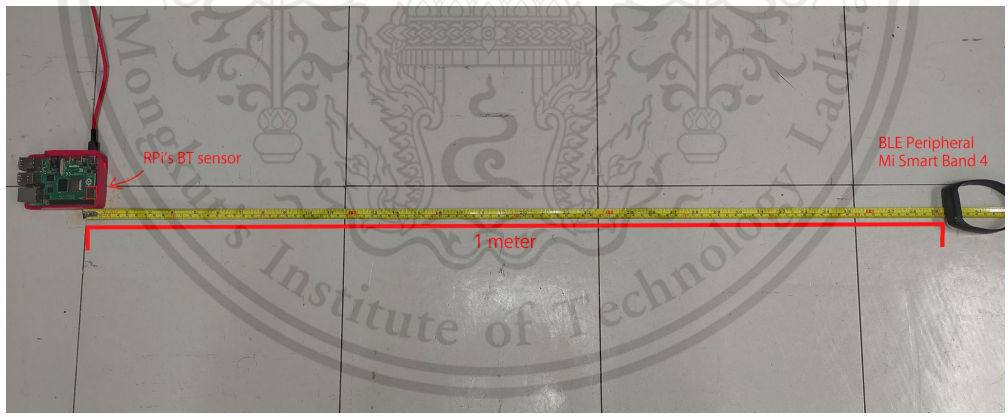

Figure 5.5: Peripheral Measured Power Calibration - Mi Smart Band 4

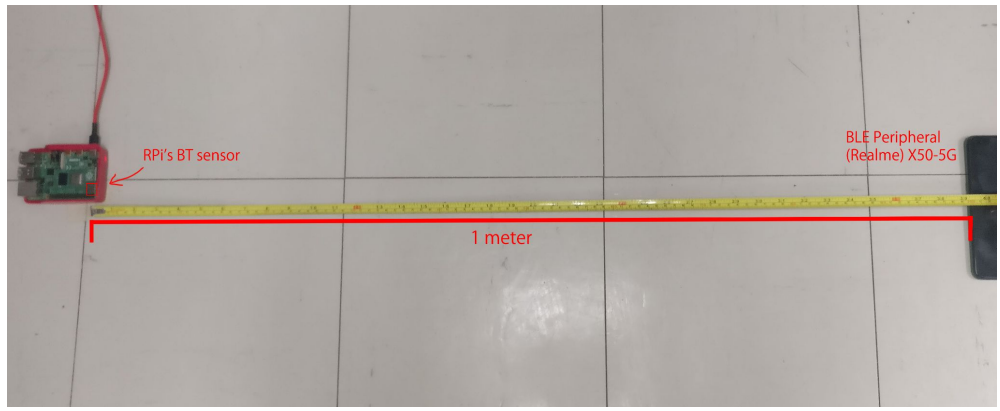

Figure 5.6: Peripheral Measured Power Calibration - (Realme) X50-5G

Each calibration test has a sample size of 20. Using the Kalman filter would be the main method of getting the final RSSI value, but for comparison, values selected using mode and mean would also be listed as comparison. This is done for each peripheral used in the experiment.

```
Device name:  3T
List size: 20

RSSI:  [-71, -70, -71, -71, -70, -71, -74, -70, -71, -74, -70, -74, -71, -74, -71, -71, -70, -74, -71, -74]
RSSI from mode:  -71
RSSI from mean:  -71.65

[-53.69230769230771, -61.86700767263427, -65.41274642745384, -67.32796182972363, -68.18981867824414, -69.0697712
4844465, -70.591529117899, -70.41021664118149, -70.59038504665408, -71.63026286359002, -71.13345149152282, -72.0
0667862577318, -71.70007414778877, -72.40050077588526, -71.97400632616115, -71.67739842064128, -71.1665961964314
3, -72.02942154744822, -71.71594384825764, -72.41147994727518]

RSSI from kalman:  -72.41147994727518
```

Figure 5.7: Calibration Test Sample

The figure above shows the raw measured RSSI values. It is then filtered using mode and mean, and then ran through the Kalman Filter, where in the end the final value is selected using the Kalman filter.

5.1.3 Location Finding and Calculations

To calculate the x and y location of each peripheral, each peripheral's RSSI value is measured from each of the reference nodes (Raspberry Pis). We then use the Kalman filter to estimate the RSSI from the raw scanned data, and then the selected RSSI values are converted to distance using the formula in section 3.2.2. After we have gathered all the distances between the reference node and each peripheral, we can find the (x, y) coordinates of each peripheral using the trilateration algorithm (section 3.1.2).

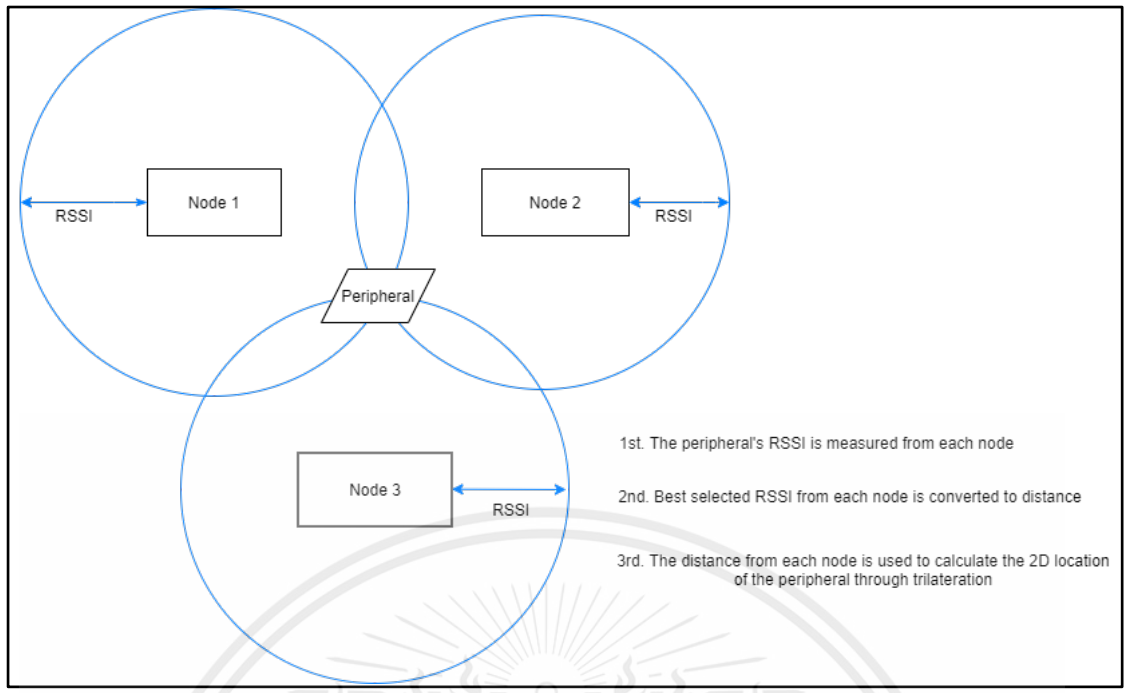

Figure 5.8: Location Finding and Calculation

5.1.4 MQTT

MQTT is a lightweight publish/subscribe messaging transport protocol for the Internet of Things. It is designed for connecting remote devices with a small code footprint and minimal network bandwidth. In our case, MQTT is utilized for connecting and sending data from one Pi to another.

5.2 Server-Side Development

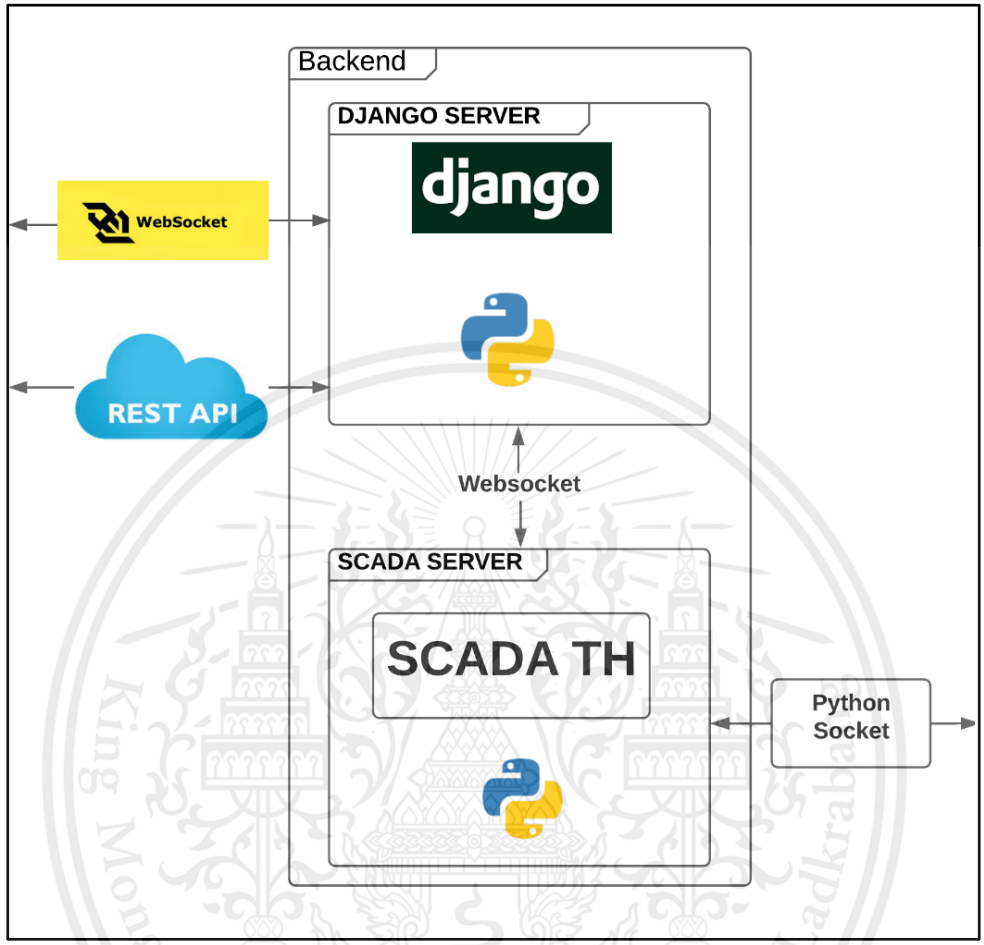

Figure 5.9: Backend Diagram

5.2.1 RESTful Web Service

RESTful web service is an architecture. It is used to communicate between frontend and backend. RESTful web service is created from the CRUD (Create, Read, Update, and Delete) concept. Commonly, it is used to create an interaction between frontend and backend. Restful Web services follow the RESTful guidelines in order to establish a communication channel between client and web server. In RESTful web service, request and response is used to transfer the data between frontend and backend by using HTTP protocol. Normally, a RESTful web service has only four types of HTTP protocol. According to this mapping

- HTTP_POST_METHOD: create a new resource on the server
- HTTP_GET_METHOD: retrieve a resource
- HTTP_PUT_METHOD: update or change the state of resource
- HTTP_DELETE_METHOD: remove or delete the resource

In our project, we use only 2 HTTP protocol methods, we use HTTP_POST_METHOD and HTTP_GET_METHOD for sending and receiving the data. Moreover, we also use JSON (JavaScript Object Notation) to transmit data in web applications. For example, send some data from the server to the client or vice versa.

5.2.2 Real-time Web Applications

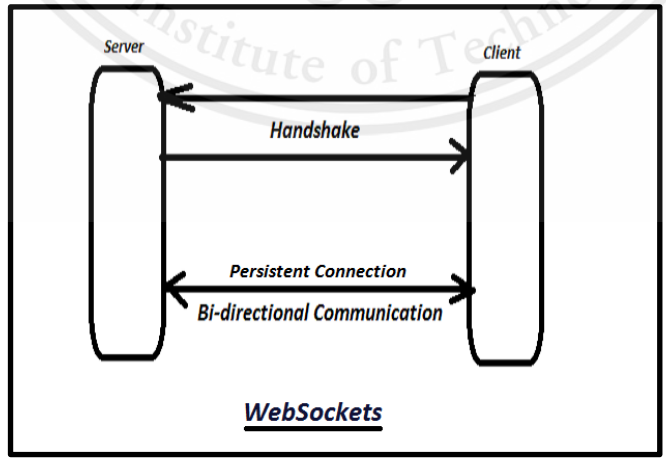

Figure 5.10: Real-time Web Application Structure

Real-time web application is one type of all types of web applications. Generally, In a normal web application, It uses only HTTP_METHOD to get and send the data. But, In the real-time application, we prefer to get more data than send the data. In a real-world situation, we probably send data only two times, but we get data one hundred times. The purpose of real-time web application is support and responsibility for real-time tasks. Why do we need to use real-time data in the application? For example, Firstly, a notification system is needed to notify immediately when a new event occurs, or something is changed without our acknowledgement. Secondly, streaming a movie or video over the internet which movie data is forced to load more movie data immediately after the user clicks a play button.

5.2.3 Django Web Framework

Django web framework is one of all the most popular web frameworks. It takes a little time to run and deploy it. It's easy to create since it is basically based on python language. Furthermore, it also supports many kinds of databases. For instance, PostgreSQL, MariaDB, MySQL, Oracle and SQLite. All of these kinds of databases are supported by the Django web framework. Moreover, In our project, we will use PostgreSQL as our main database which I will talk about PostgreSQL in the next topic. In addition, The MVT (Model View Template) is a pattern which is a slightly modified version of the Model View Controller (MVC) pattern and used in the Django web framework. Next thing which you should know about the Django web framework is about how Django collects data. Django uses the Object Relational Mapping layer (ORM) to interact with various relational databases. For example, create, read or access, update and delete.

5.2.4 Django Rest Framework

Django Rest Framework is a framework which was used to create Web APIs. We will shortly call it DRF. DRF supports many HTTP protocols. For instance, POST, GET, PUT, HEAD and etc. DRF is also used in our project. For instance, In our project, We use DRF to return the list log history to see the past of the user including timestamp, x_coord, y_coord, room, floor and location of the user.

5.2.5 Django Channel

The Django channel was used in our project (In SCADA server and our server). Since, we want to use the ability of the Django channel which is handling WebSocket. In addition, WebSocket uses an asynchronous connection. The Django channel lets you choose to use a synchronous or asynchronous connection in your project. Django channel relies on the ASGI or Asynchronous Server Gateway Interface. What is an ASGI interface? Normally, Django web framework will use WSGI or Web Server Gateway Interface that supports only a synchronous python application. Whereas ASGI interfaces support both synchronous and asynchronous python applications. In the Django channel, it consists of several packages: 1) Channel: the Django integration layer 2) Daphne: Django Channels HTTP/WebSocket server 3) asgiref: the base ASGI library 4) channel_redis: the Redis channel layer backend.

5.2.6 Celery

Celery is a distributed task queue and designed to do real-time tasks or jobs. It comes with a scheduling system. In addition, Celery is a one kind of python worker which is used in both SCADA server and our server (Django server). Celery workers run on the thread. Therefore, it can run concurrently with the Django web server. In the SCADA system, Celery workers work as a communication middleware that works like a communication port. We use celery workers to read and save the data of each device on the real-time database on the SCADA system. Furthermore, On our server (Django server), We use celery to send real-time position information, real-time contact position and update contact table real-time.

5.2.7 Redis

Redis is an in-memory data structure store and supports many data structures which are strings, lists, sets, sorted sets, and hashes. Redis was used as a database cache and message broker. In addition, Redis is also used in our project. Because the SCADA system uses a redis to collect and store timestamp and data of each device in a SCADA real-time database. Redis uses a publish and subscribe technique to update real-time data by using a worker. In addition, Redis is used in our server (Django server) to set a hostname of channel layers and celery broker URL.

5.2.8 Docker

What is Docker? It is a container which contains services, command and OS (operating system) that you use in your application. In this project, We have already used Docker to set up the environment of the Django web framework (Our own server). Furthermore, It makes Django easy to deploy on the cloud service website. Since, these cloud service websites have already supported deploying a Docker container.

5.2.9 PostgreSQL

PostgreSQL is a one kind of open-source relational database management system (RDBMS) type. PostgreSQL is normally called only Postgres. In addition, Postgres is free for everyone. It is designed to handle a range of workloads from single machine to data warehouse or web services with many concurrent users. Furthermore, PostgreSQL is a default database which is used in macOS servers. It is also available in Linux, FreeBSD, OpenBSD and Windows. Next topic, I will talk about the structure of PostgreSQL. PostgreSQL uses a database cluster to initialize a database storage area. A database cluster is a collection of databases managed by a PostgreSQL server. We use PostgreSQL to collect and store all the data in our system.

5.2.10 PostGIS

PostGIS is a spatial database extender for the PostgreSQL object-relational database. PostGIS is free and open source. It is created to support geographic objects which allow spatial analysis and location queries in PostgreSQL and Django framework (If you set your database to use a PostGIS extension). For example, In our project, We use PostGIS to store the location (latitude and longitude) of a user.

5.2.11 SCADA System

SCADA stands for supervisory control and data acquisition. We also use the SCADA system in our project which was created by the research team of Dr. Visit Hirankitti ‘s laboratory. The goal of the SCADA system is to send, receive and visualize the real-time data in one place. Moreover, the Python language was used to develop a SCADA system. The SCADA system was created based on the Django web framework. In the SCADA system we can use a worker to run parallelly many task with Django server, get data from

many sensor, store data on the server and can export history or log data as in Excel or PDF file type, display the real-time data as line graph or bar graph, show the status on each sensor on SCADA dashboard and use a data tag system to store and classify data into single or compound tag and etc. Data tag system form is “SystemName.DeviceName.TagName”.

For example, We use IPS_SYSTEM.BT_TAG_1.X_COORD in our project to get the real-time x-axis of a Bluetooth tag. What do we use in the SCADA system? Since the SCADA system is quite stable and developed in many versions. Therefore, we use SCADA as a middleman to receive the data from Raspberry Pi by using the Python socket protocol and send the data to our server by using WebSocket protocol.

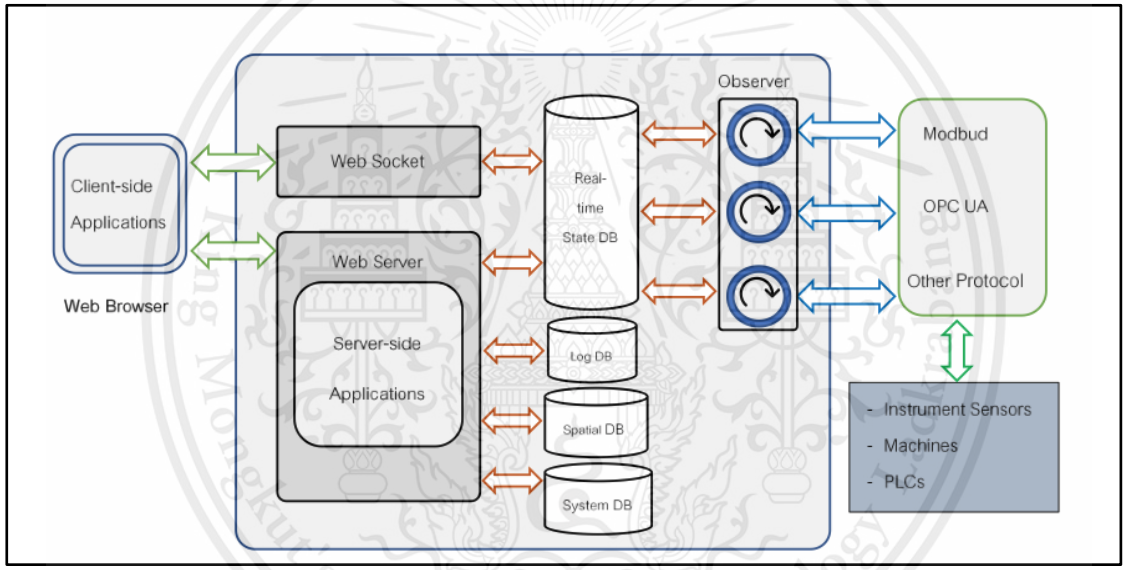

Figure 5.11: SCADA Architecture

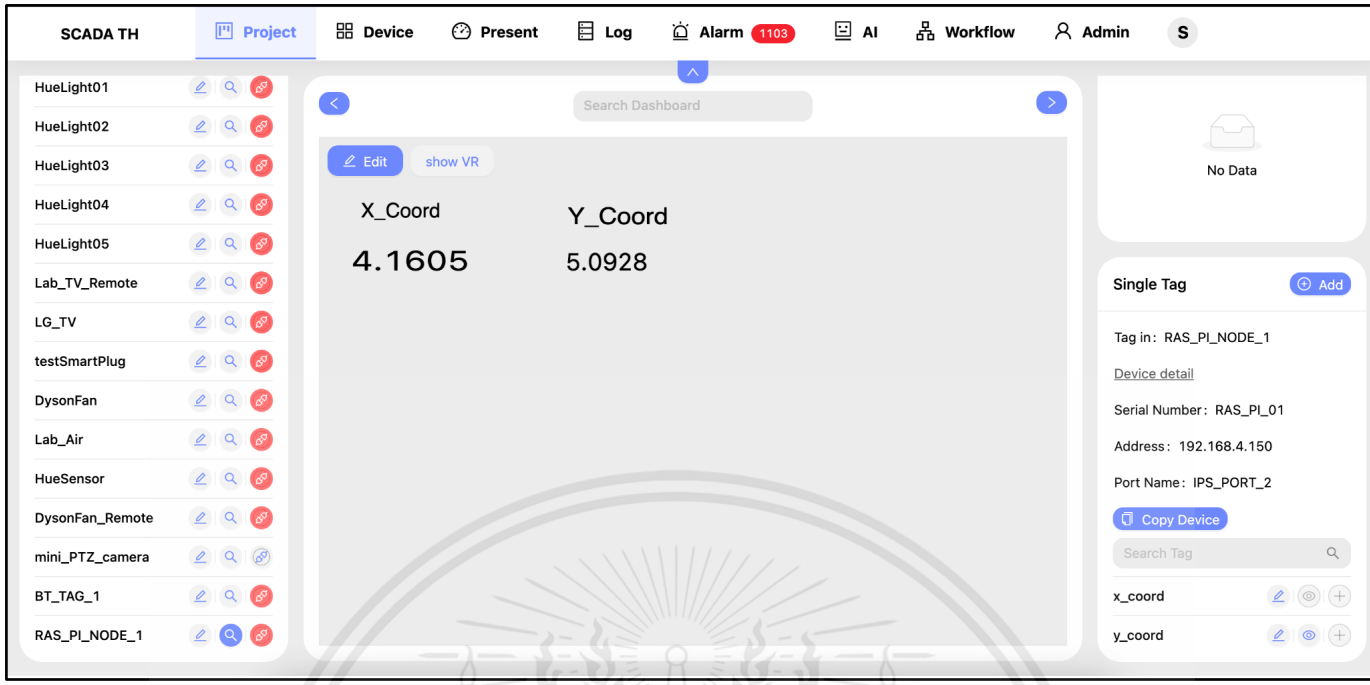

Figure 5.12: Screenshot from SCADA Dashboard

5.3 Frontend-Side Development

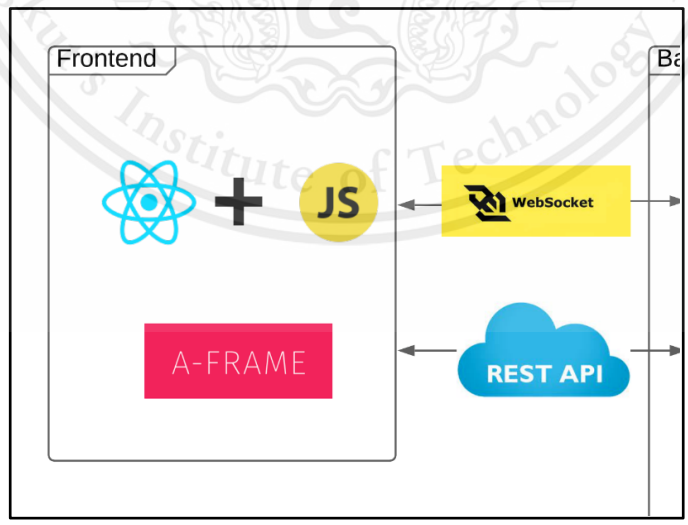

Figure 5.13: Frontend Diagram

5.3.1 React Web Framework

React is a frontend web framework which we use to create our web page. Moreover, React web framework works with DOM (Document Object Model). In react web framework, It also has a state which helps to choose which component should render or not. It helps quite a lot when you want to show data on a website as it uses JSX and HOOK for implementing. Which make the code look cleaner and easy to manage. With React, we can also use state and props which help creating, setting, moving data.

5.3.2 A-Frame

A-Frame is a web framework for building virtual reality (VR) experiences. A-Frame is based on top of HTML, making it simple to get started. But A-Frame is not just a 3D scene graph or a markup language; the core is a powerful entity-component framework that provides a declarative, extensible, and composable structure to three.js. Also, A frame is easy to test as you can test it on Glitch, an online code editor that instantly hosts and deploys for free.

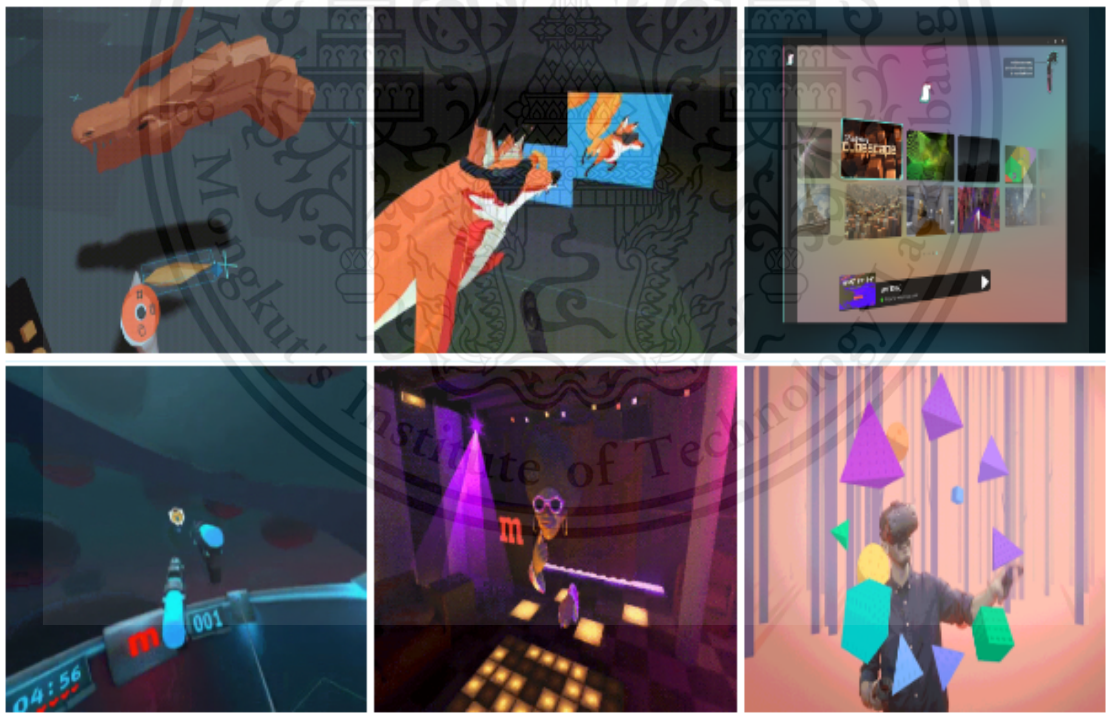

Figure 5.14: A-frame Screenshot

5.3.3 Web-Socket

The WebSocket API is an advanced technology that makes it possible to open a two-way interactive communication session between the user's browser and a server. With this API, you can send messages to a server and receive event-driven responses without having to poll the server for a reply. WebSocket is also being used for real time data transfer as it does not need to wait for request and response like HTTP as WebSocket needs only first-time connection to the server then the message can be transferred without waiting for the response of the server.

5.3.4 GPS Map

GPS Map is an API that gets from google map for getting that location and shows it on the map to show outdoor location. Google map provides spatial data for outdoor positioning which it receives from the satellite. The orbiting satellites transmit signals that allow a GPS receiver anywhere on earth to calculate its own location through trilateration.

Chapter 6
Experiments

This chapter explains the result of the software development. It displays the implementation of the project and the outcome of the product. The results are separated into 2 different parts: the hardware and the software part.

6.1 Hardware

6.1.1 Calibration

The following line graph shows the calibration results from OnePlus 3T. Each device is tested 10 times, with each instance having a sample size of 20 scans. The final value of each device is then chosen with each of the 3 methods for comparison. There are 3 peripheral devices that are being calibrated: 2 smartphones (OnePlus 3T and Realme X50-5G) and a smartwatch (Mi Smart Band 4).

To calibrate a peripheral, set the peripheral to be 1 meter away from the reference node. The value we get would be used for the *measured power* variable in the distance from RSSI conversion formula (figure 3.5).

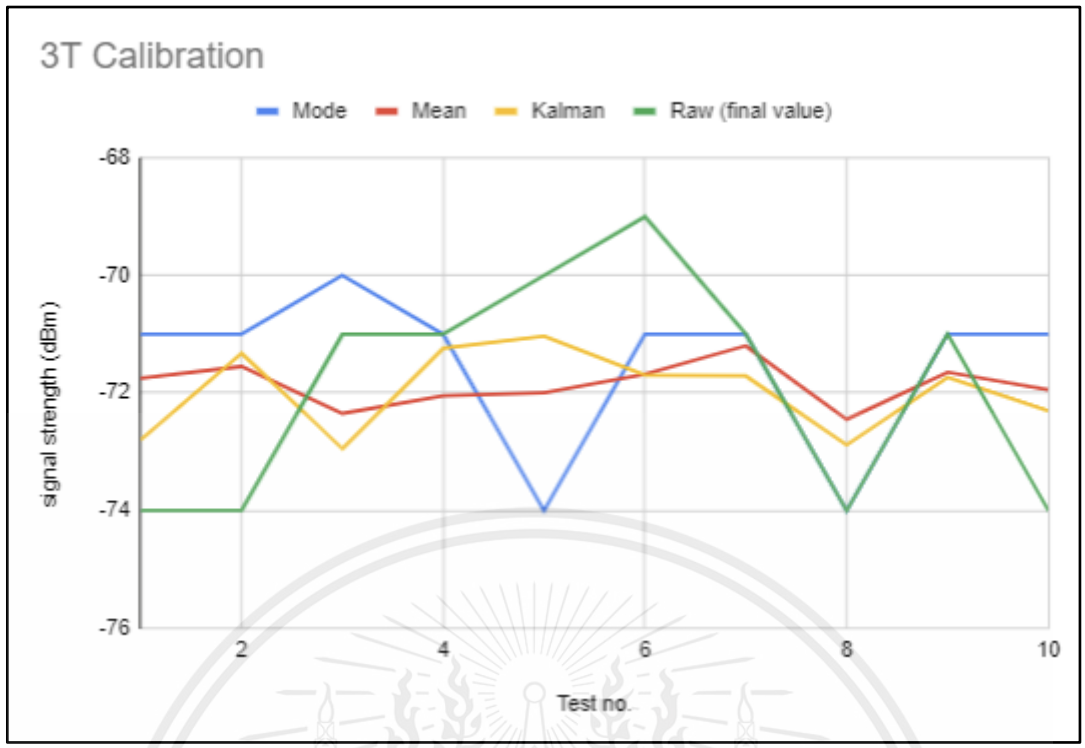

Figure 6.1: (Oneplus) 3T Calibration Results

The OnePlus 3T's calibration is shown as the sample and as you can see, the final calibrated value we get from mode has the most extremes out of all the methods. Using the average (mean) and the Kalman filter yields very similar results for the calibration test.

The Kalman filter is mainly affected by the latest measured values, so if the latest measured value starts to head to a different extreme, the final measurement will also steer in that direction, as shown from test 2 until 4. This behavior of the Kalman filter is both beneficial and a drawback. It's beneficial for being quicker to notice changing trends, however it can also mess up the final measurement if the latest measurement is off by a great margin. This can be further improved by spending more time in optimizing your Kalman filter.

From the data gathered, the measured power of -72, -75, and -67 would be chosen for the 3T, X50-5G, and Mi Smart Band 4 respectively (X50-5G's and Band 4's data can be found in the appendix).

6.1.2 Location Accuracy and Consistency

With some test cases, we can observe how different the measured and calculated location is compared to the real-world location. Each test case would randomly put the peripheral(s) in different locations, to at least replicate how devices move around in reality. The radius in the trilateration calculations would be calculated based on values selected from the Kalman filter. First, let's find the location of just one peripheral, then we'll move on to testing with 2 and 3.

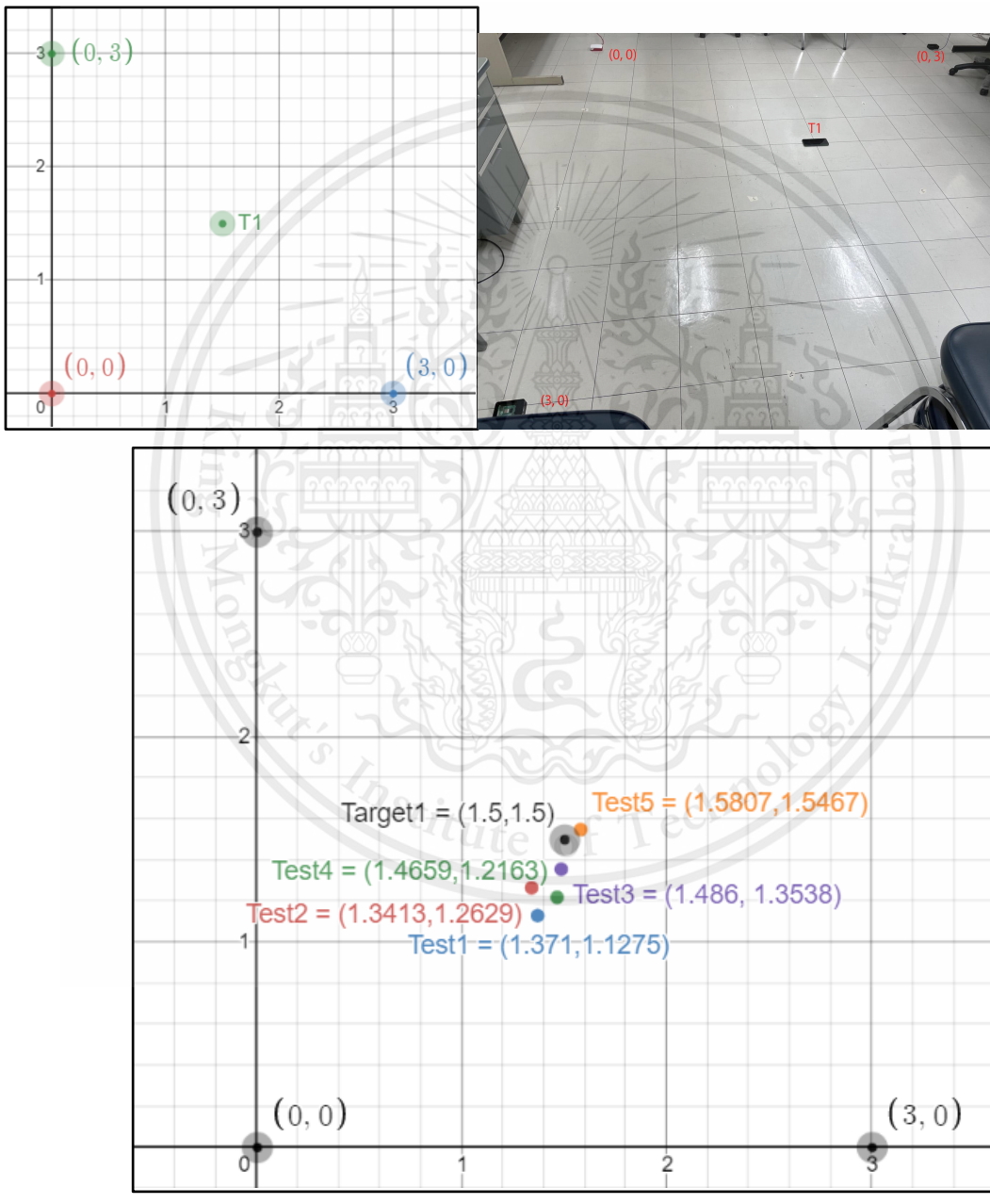

Figure 6.2: 1 Peripheral Location Test

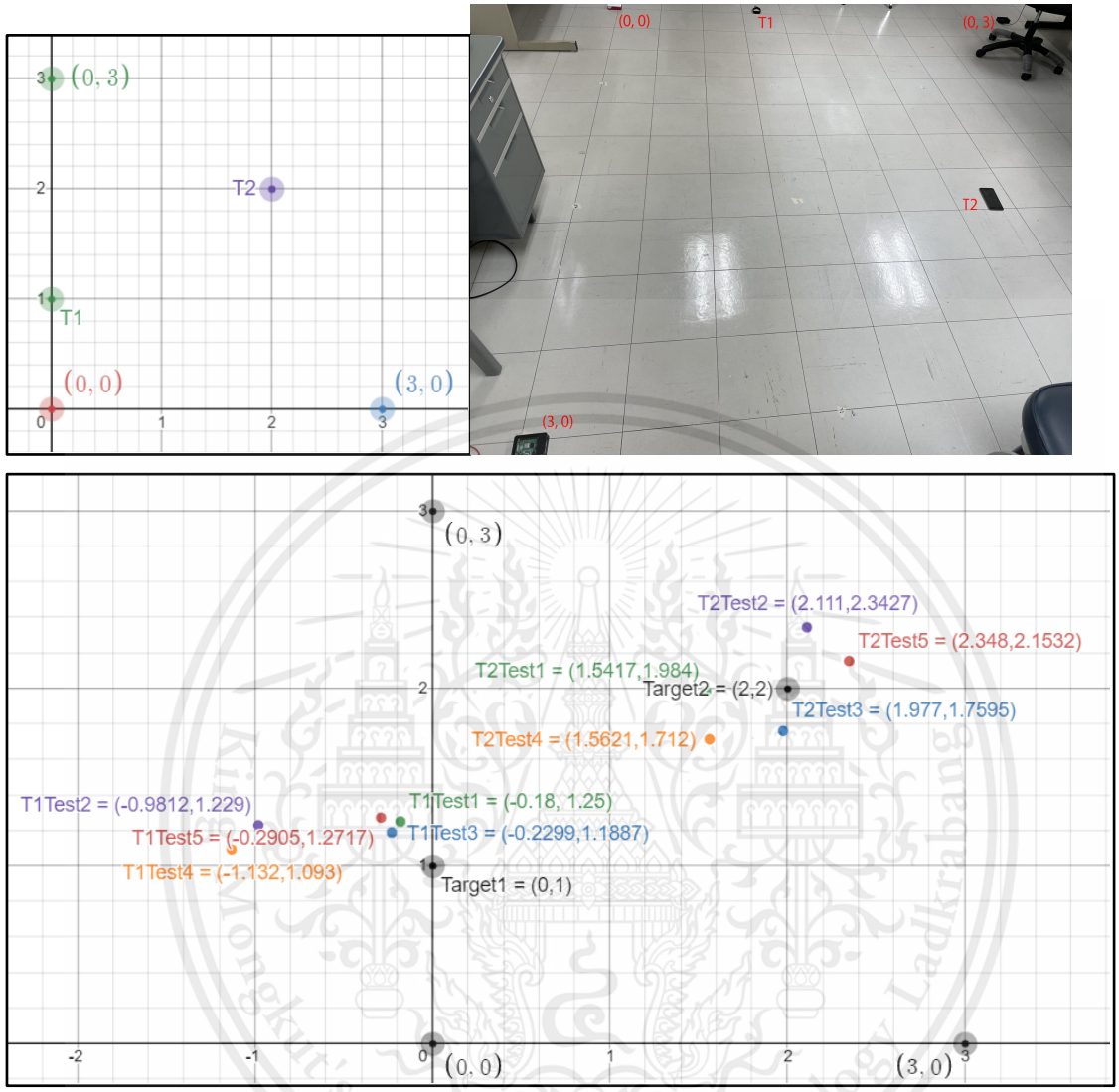

Figure 6.3: 2 Peripherals Location Test

When another device gets added, more interference is also added. The Bluetooth signal of the newly added device adds more noise and more traffic to the Bluetooth frequency, causing the average difference to vary greater. The average difference in the x-axis is higher than the difference in the y-axis due to the fact that there are 2 RPi nodes in the x-axis compared to just 1 RPi in the y-axis, adding more variance to the x-axis measurements.

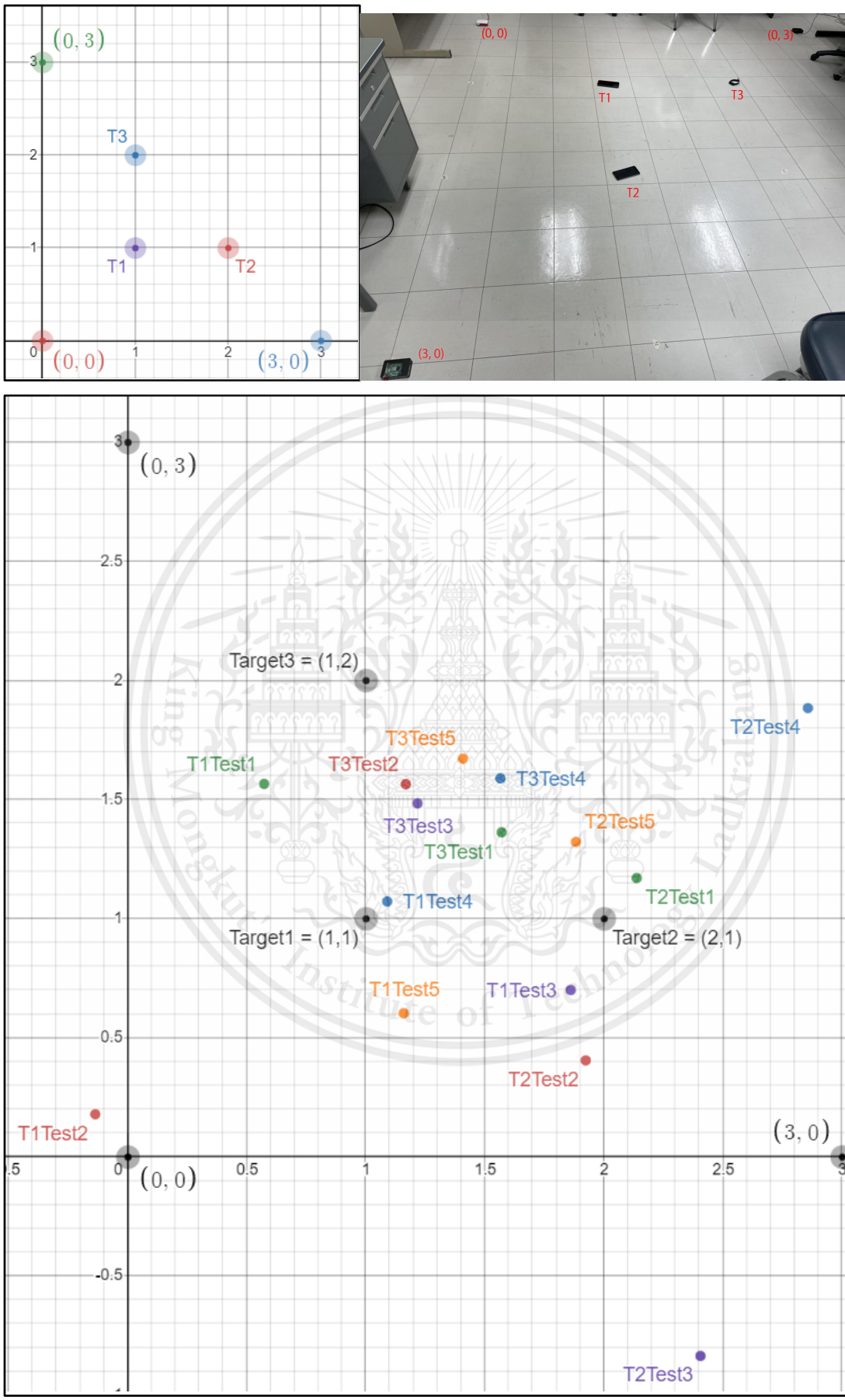

Figure 6.4: 3 Peripherals Location Test 1

Adding a 3rd peripheral greatly adds visual clutter, so each points' coordinates are omitted, but they can still be referred to in the table in the appendix. As expected, adding a 3rd device further increases the difference. I also chose this particular case to demonstrate that multiple devices could also block the line of sight each reference node has to its target. If another device is added to the area, the result difference is sure to increase, and the accuracy would definitely decrease. We can also see similar results for the following 2 tests.

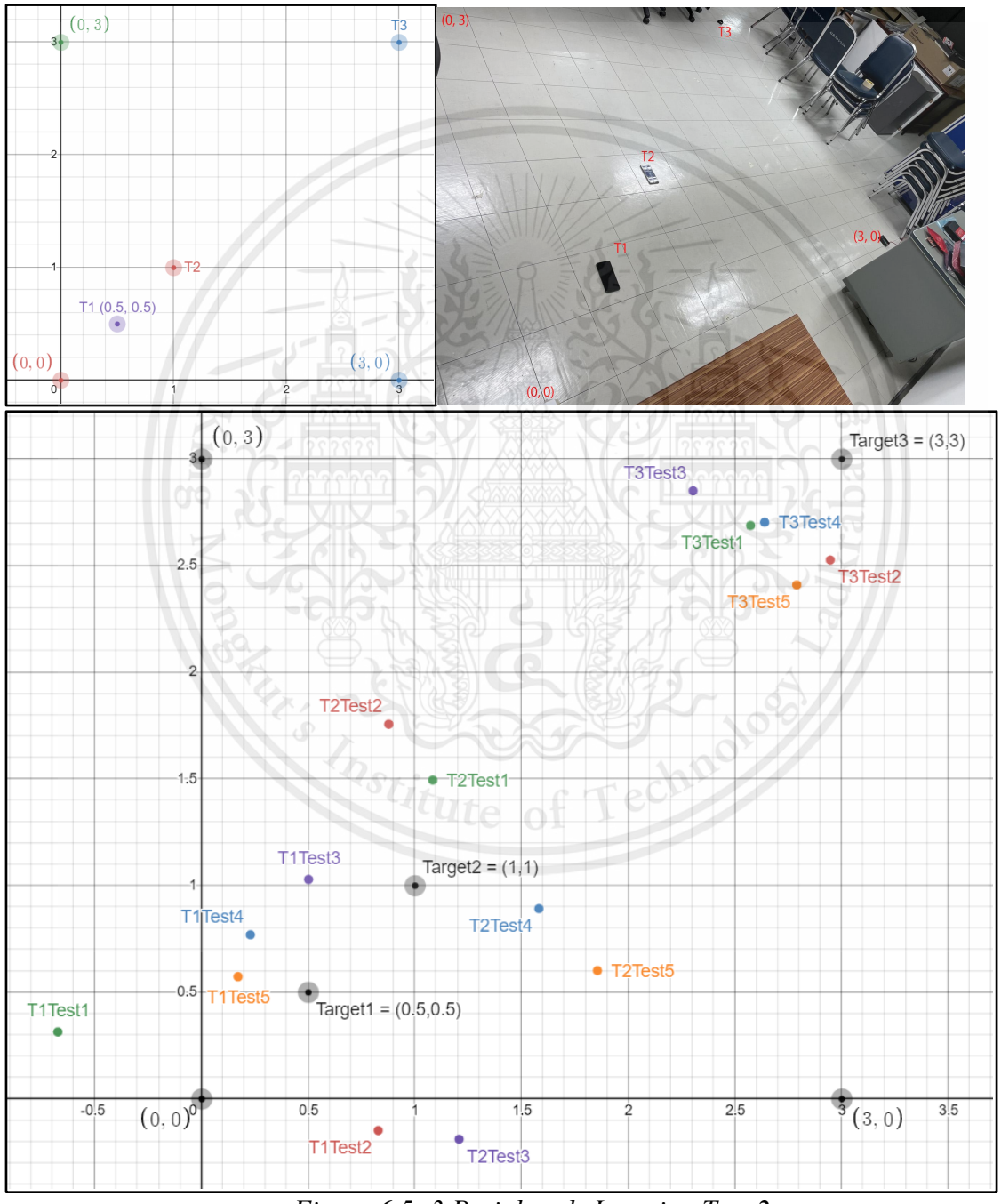

Figure 6.5: 3 Peripherals Location Test 2

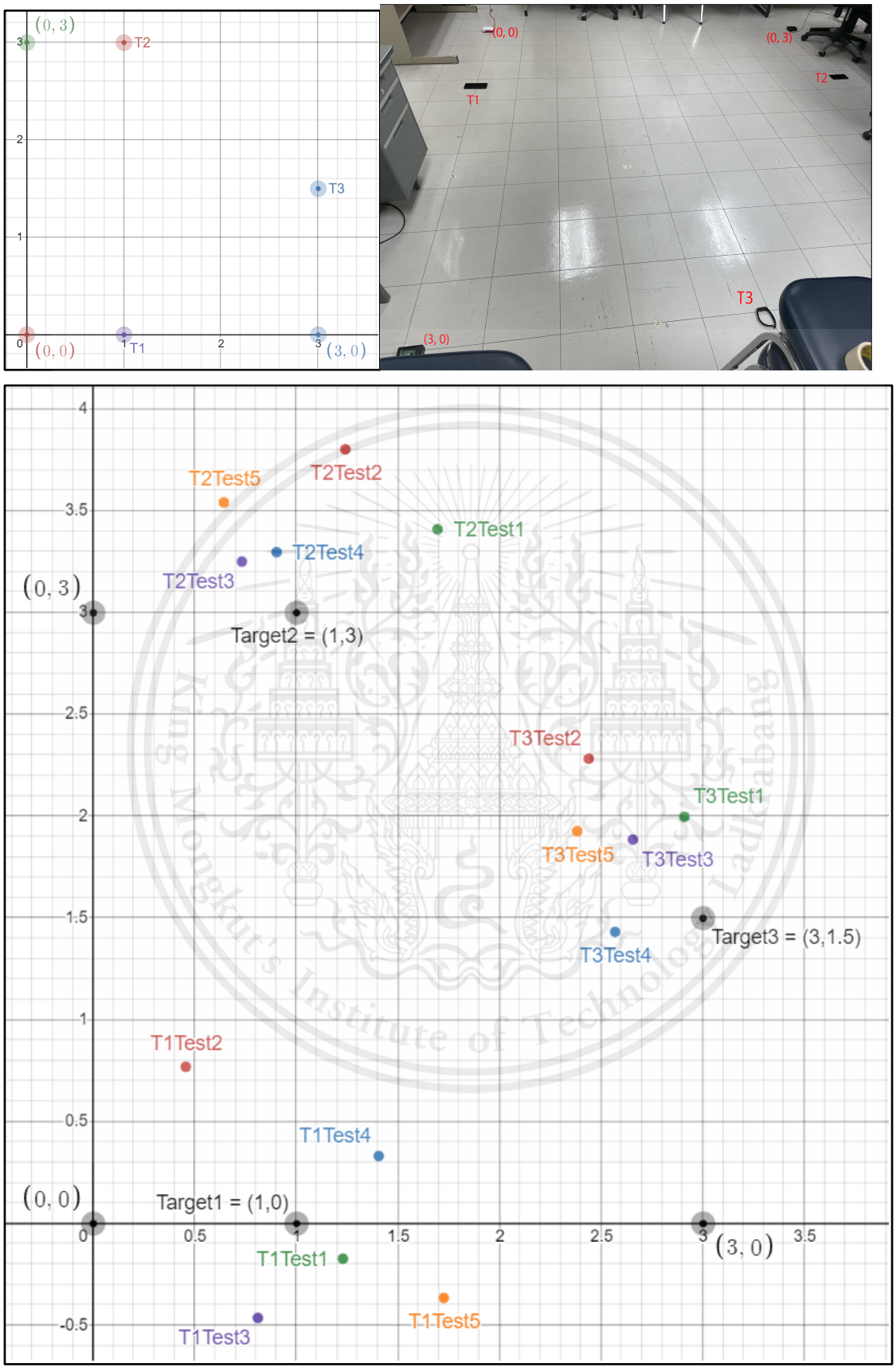

Figure 6.6: 3 Peripherals Location Test 3

6.2 Software

6.2.1 Frontend part

6.2.1.1 Bluetooth Low Energy (BLE) Application

A simple app for simulating a Bluetooth Low Energy peripheral device. Since mobile devices can't directly turn on BLE by default, a third-part application is needed to help turn on BLE.

Figure 6.7: BLE Peripheral Test made with Cordova

The application creates a simple UART service that advertises every second via Bluetooth Low Energy. By creating and advertising the service, BLE is effectively turned on. This allows the surrounding BLE central devices to scan and detect the simulated peripheral. The application is made using the Apache Cordova framework along with Cordova's BLE peripheral plugin.

6.2.1.2 React JavaScript Framework (Web Application)

Login Page

Since last semester, we have developed our website with React framework. We will show the login page in both mobile and desktop versions. And we created 2 versions of the interface which are the mobile and laptop version.

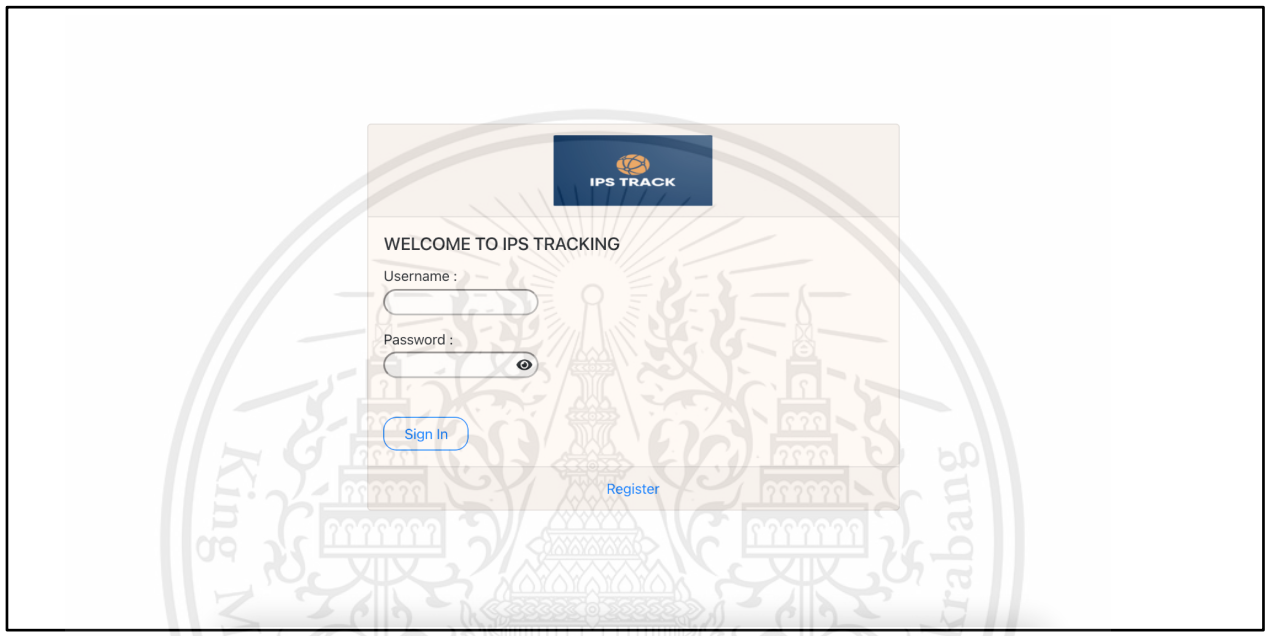

Figure 6.8: Login page in desktop version

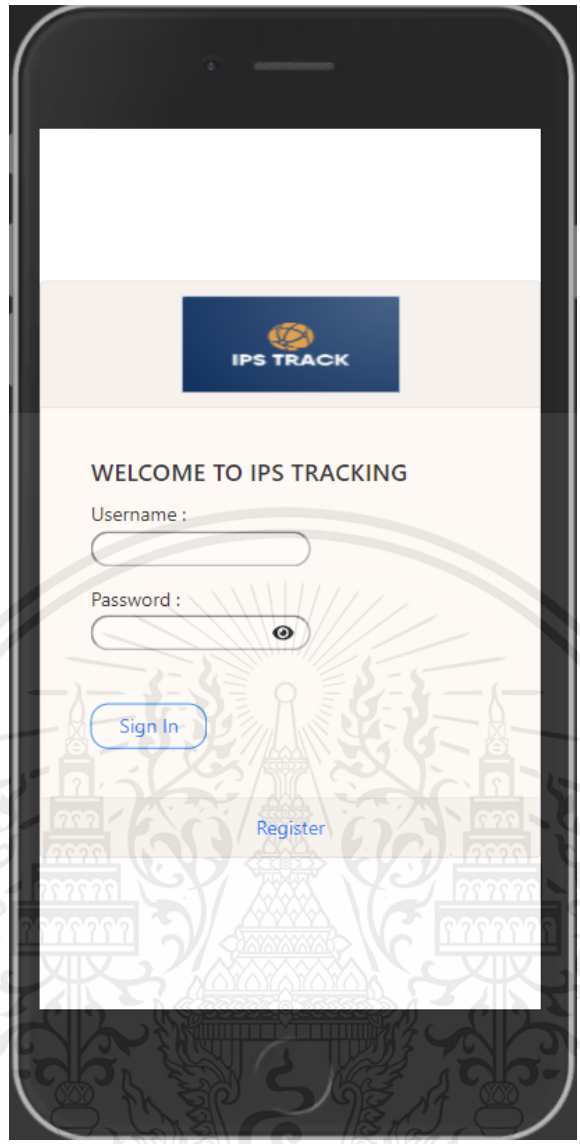

Figure 6.9: Login page in mobile version

Registration Page

We will show the registration page in both desktop and mobile versions for users to register and add a new account to the server.

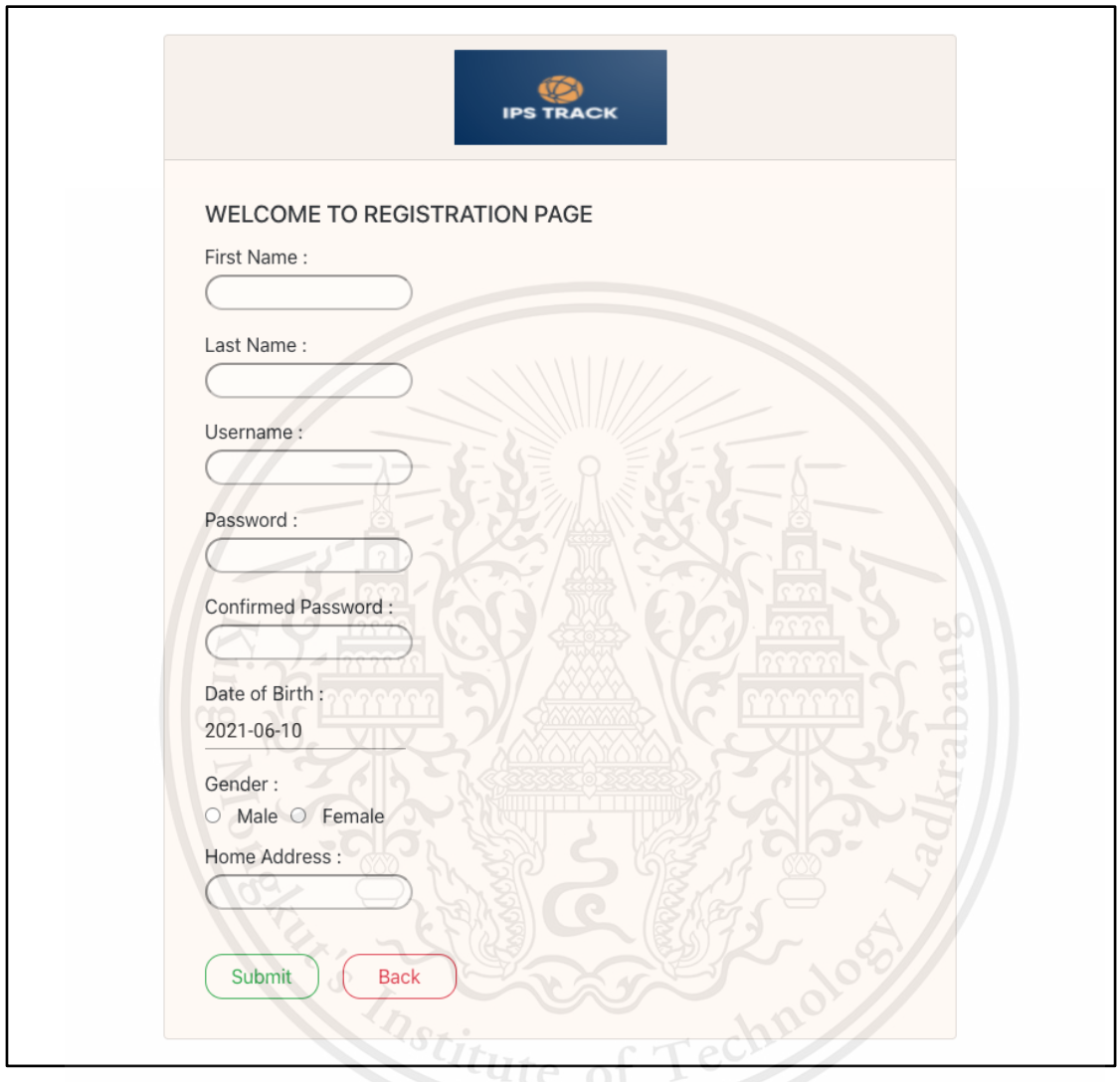

Figure 6.10: Registration page in desktop version

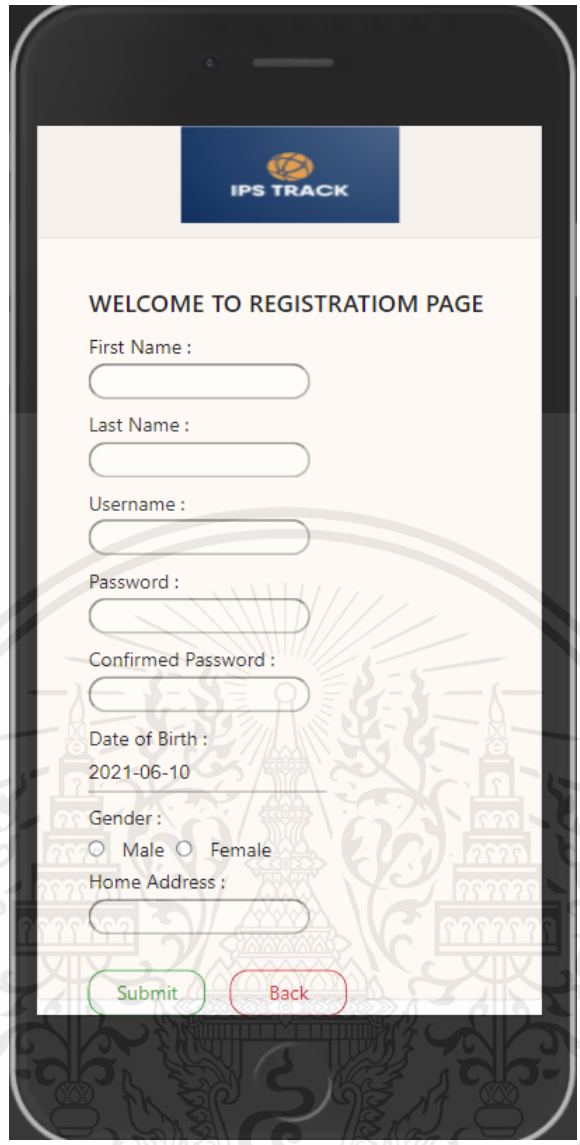

Figure 6.11: Registration page in mobile version

Indoor location Page

Indoor map will show the user current location within the select room for the user to see if he is too close to anyone.

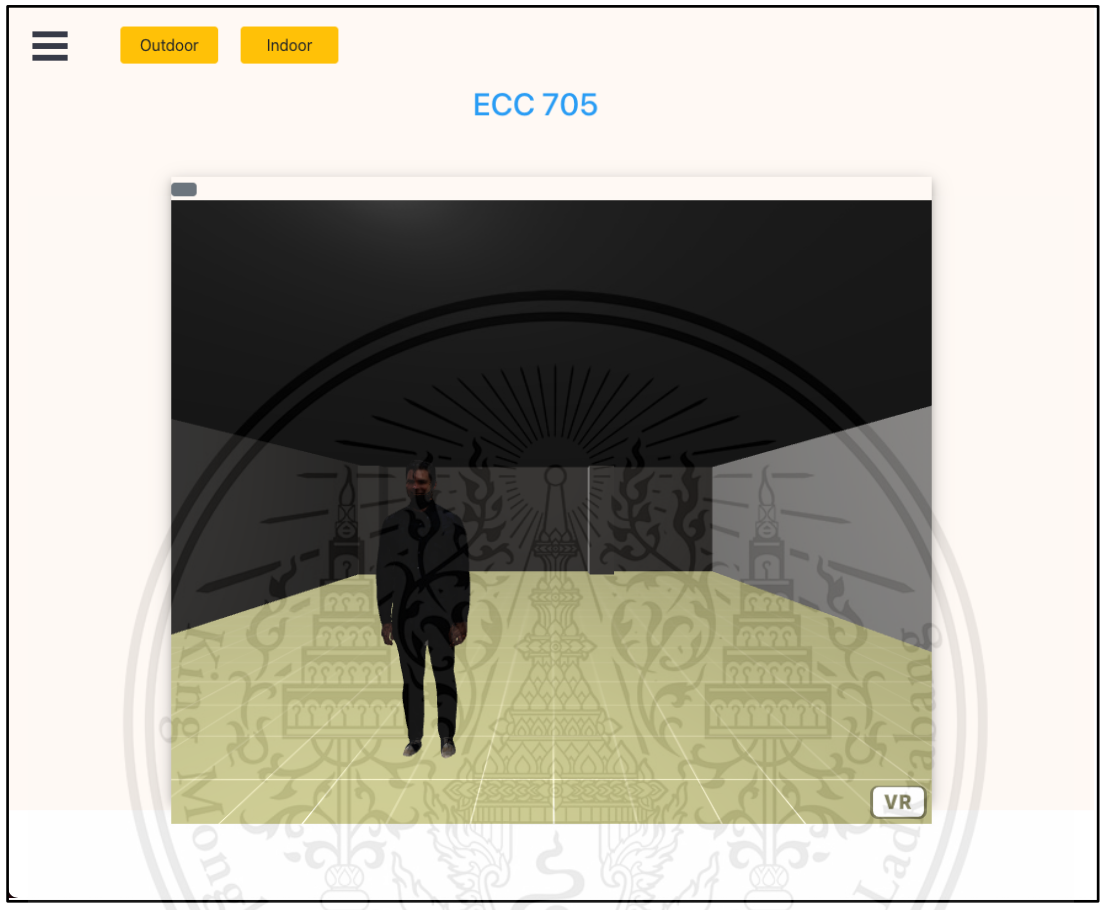

Figure 6.12: Indoor location page in desktop version

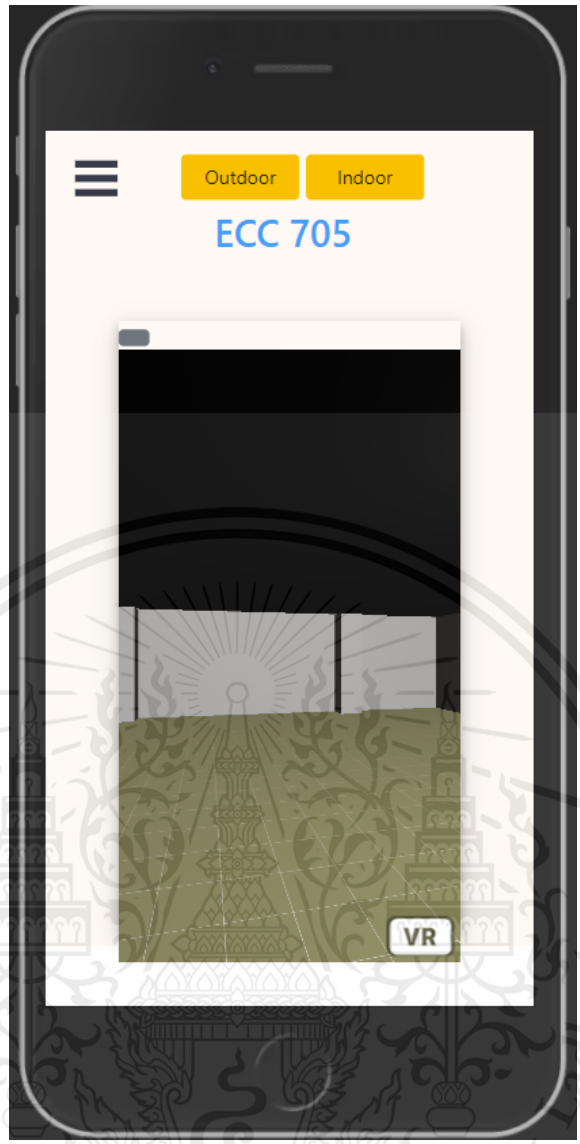

Figure 6.13: Indoor location page in desktop version

Outdoor location page

For an outdoor map our intention is to visualize a timeline for that particular server, so they know where they have been for the last 14 days with some information so that they can relate to those places.

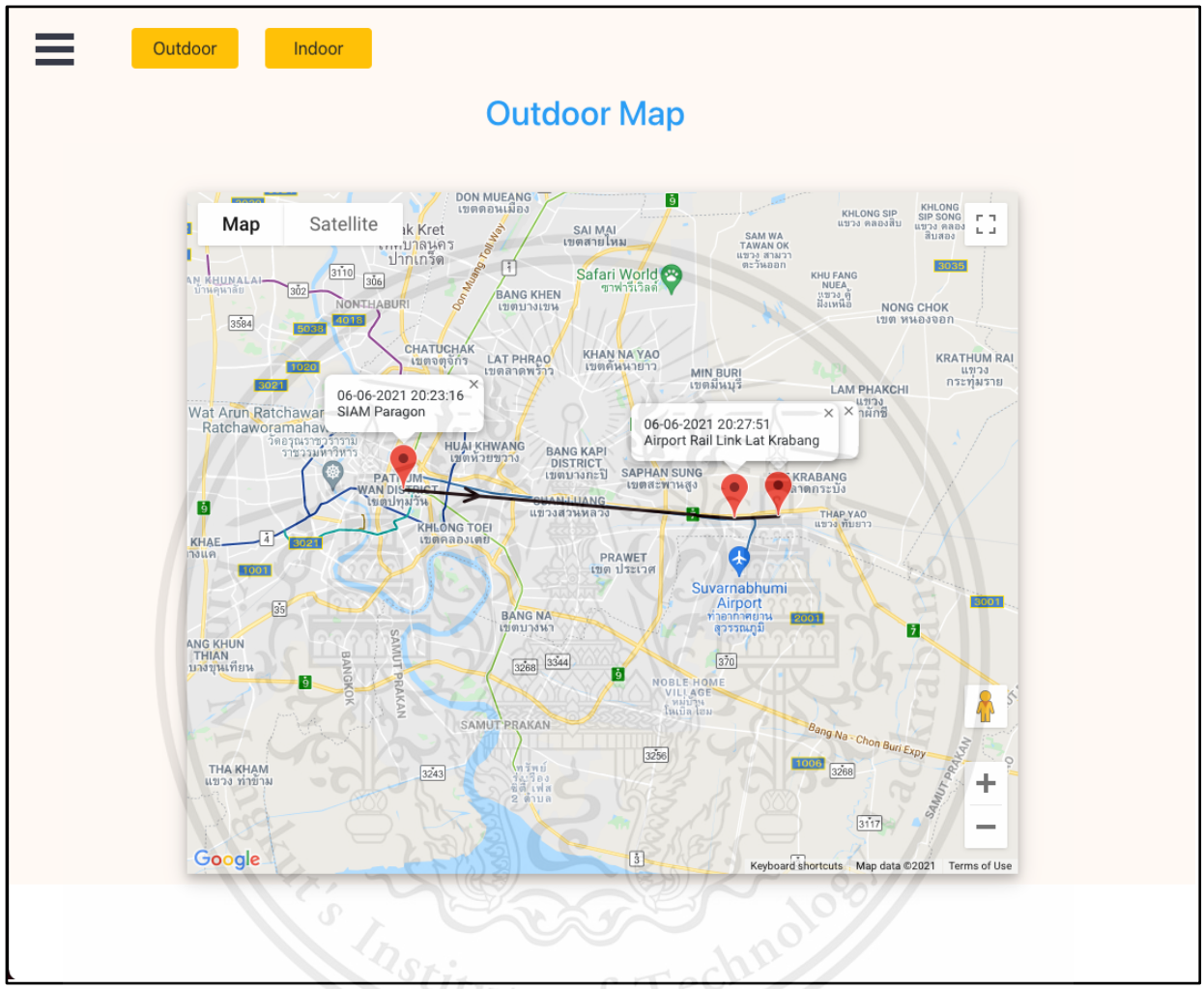

Figure 6.14: Outdoor location page in desktop version

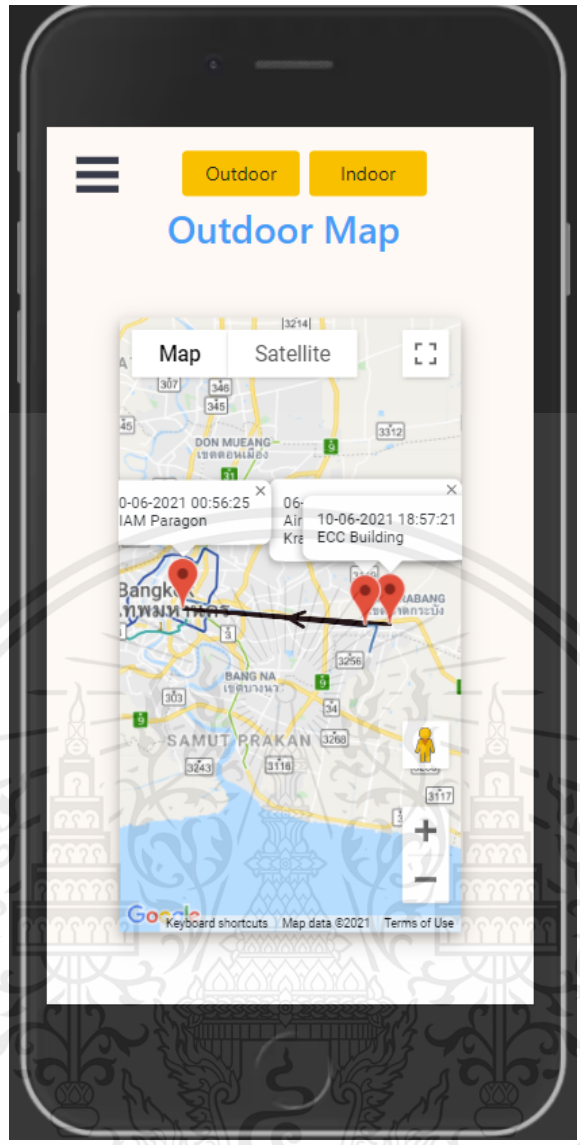

Figure 6.15: Outdoor location page in mobile version

Log History page

Log history is similar to the outdoor map page as it also shows the timeline of a user, but in this page, users could search and find the place for more than 14 days. Moreover, this page shows more detail of those places.

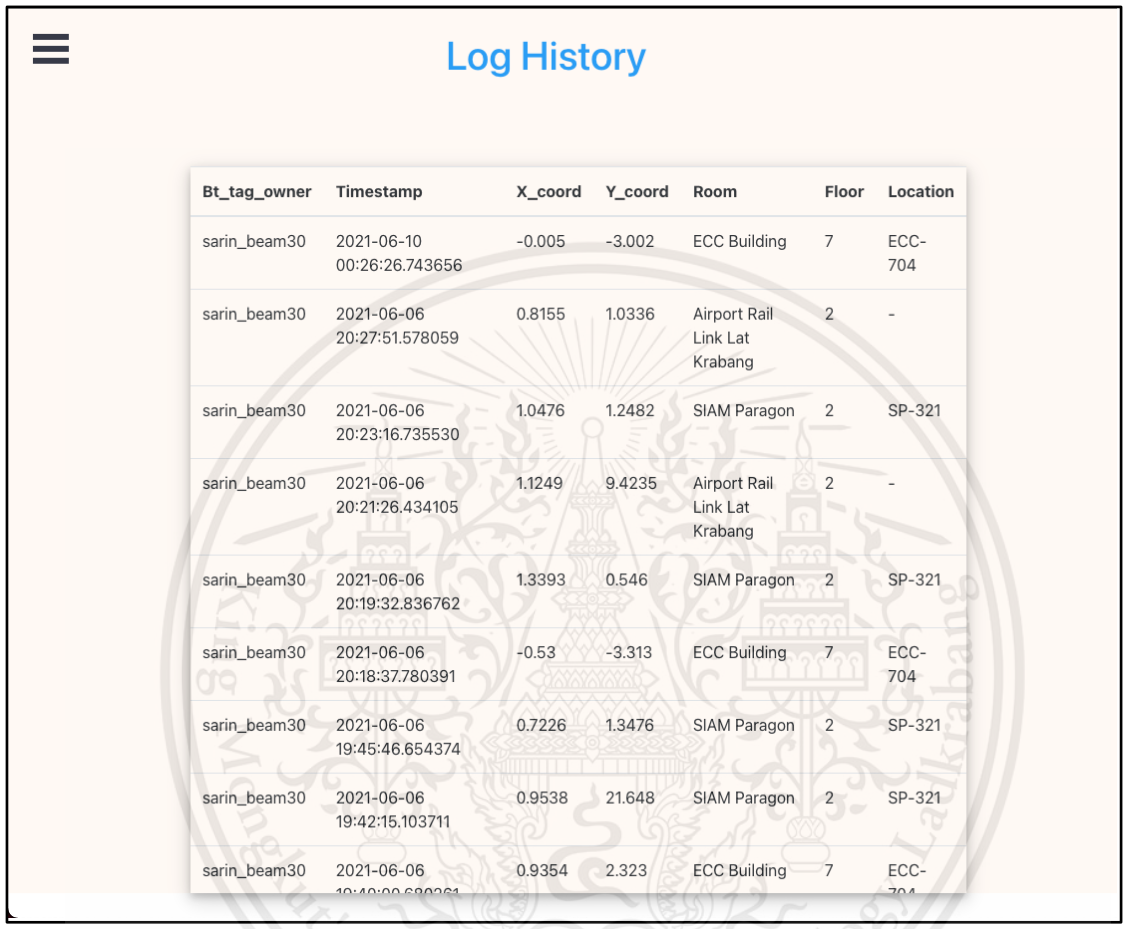

Figure 6.16: Log history page in desktop version

Figure 6.17: Log history page in mobile version

Notification page

Notification page will show the distance of 2 users, if both are negative, the color will be green indicating very low risk. If one of them is positive the color will be changing from green to red, orange, yellow depending on how close they are where red is the closest and yellow is the furthest.

Figure 6.18: Notification page in desktop version

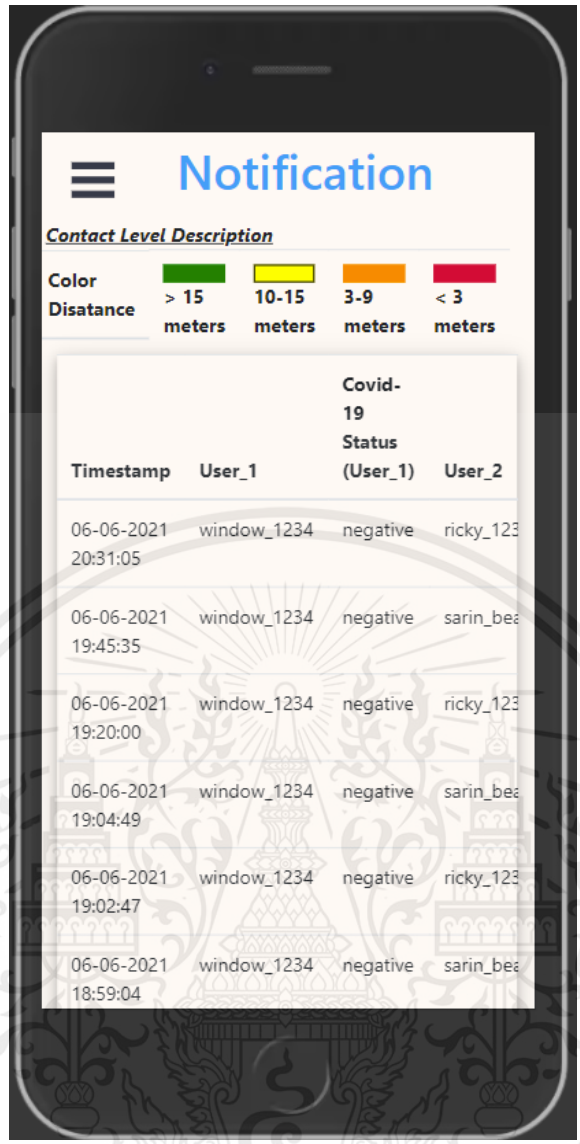

Figure 6.19: Notification page in mobile version

Profile Page

For the profile page, the user can see their information and be able to edit them anytime they want.

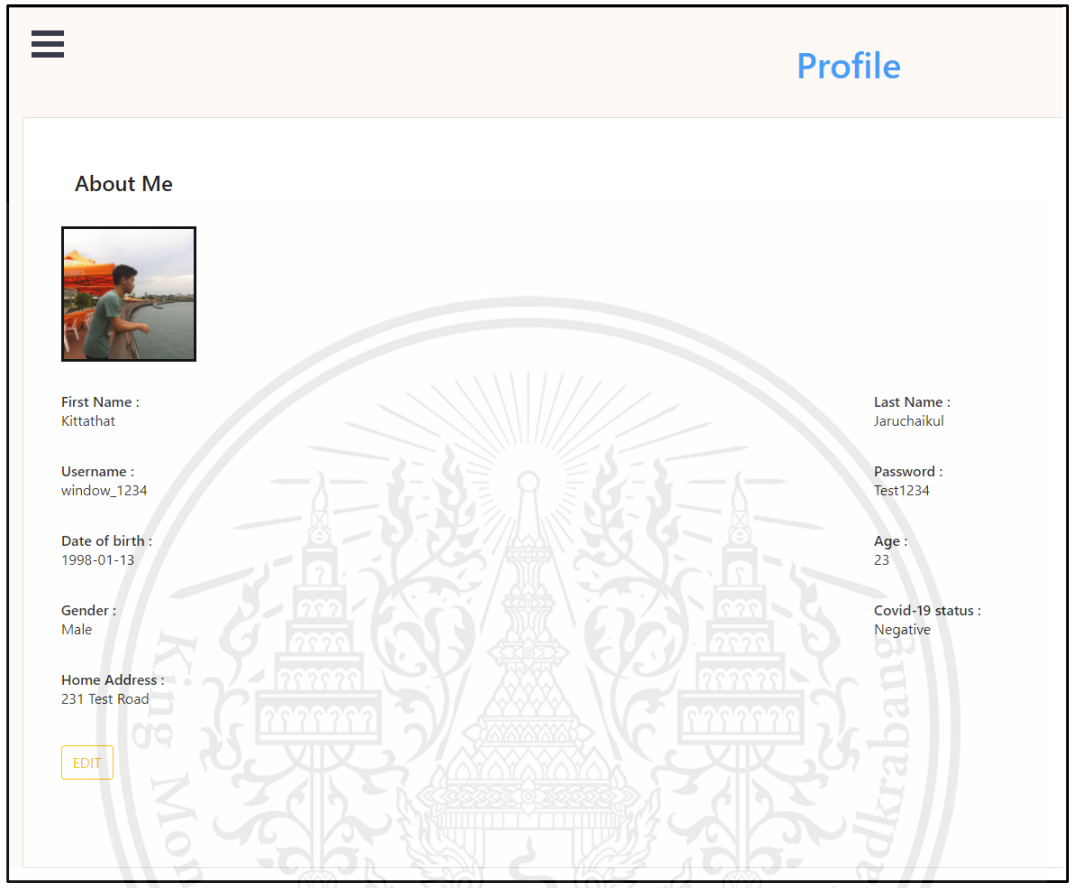

Figure 6.20: Profile page in desktop version

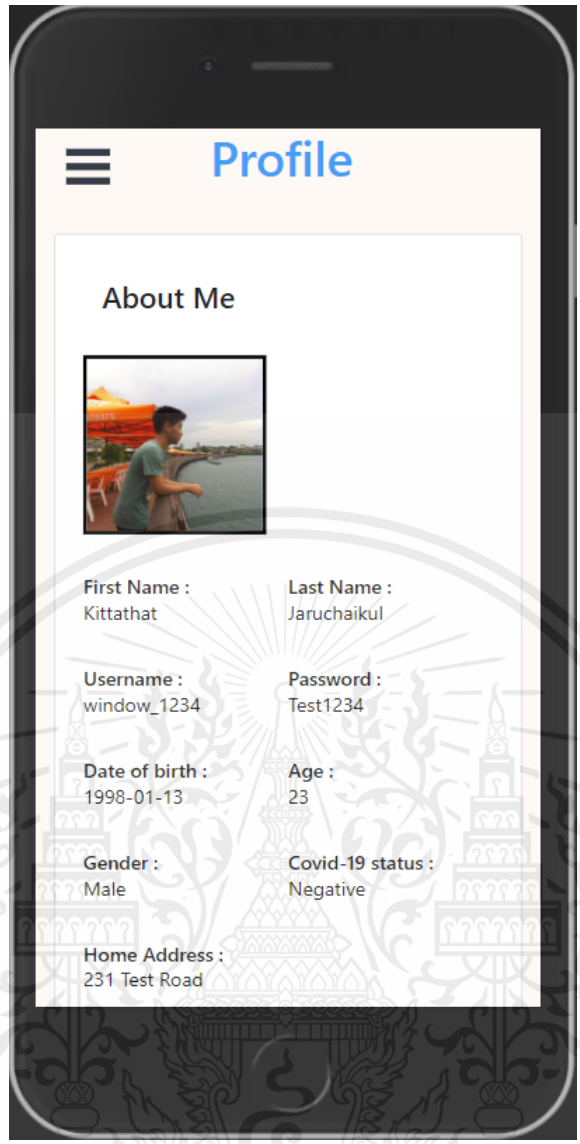

Figure 6.21: Profile page (1) in mobile version

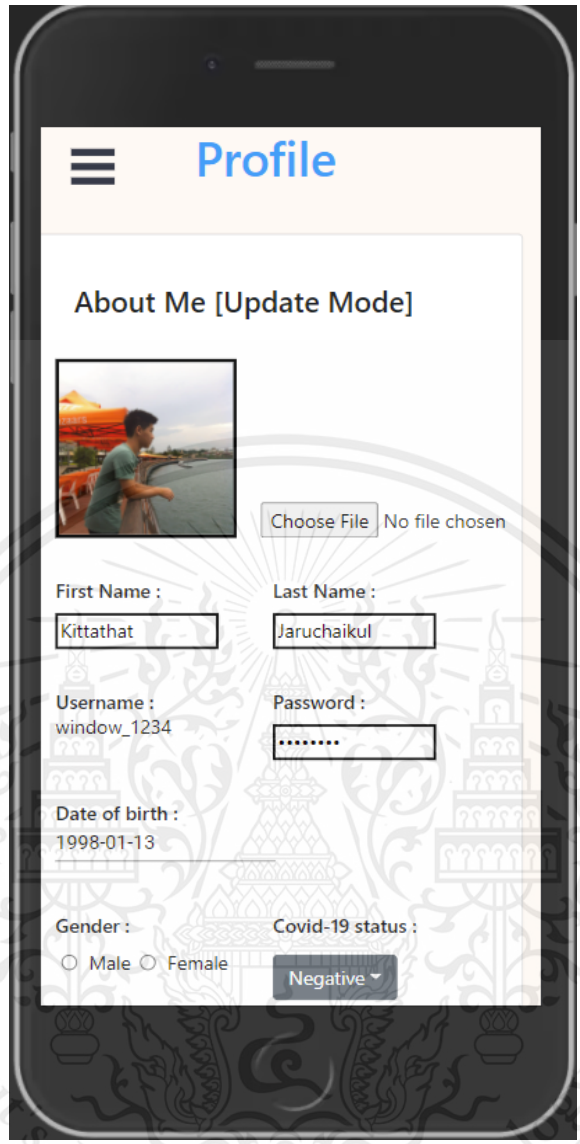

Figure 6.22: Profile page (2) in mobile version

Sidebar Part

Side bar is for the user to go around the website easily by clicking the name of the page. Moreover, users can see their compact information.

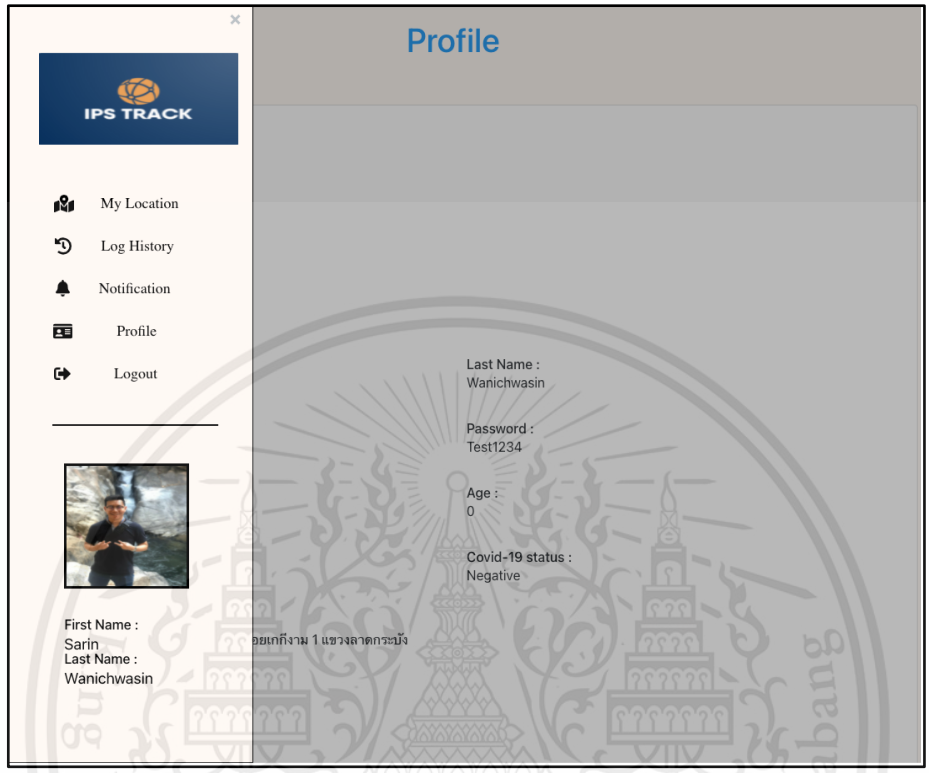

Figure 6.23: Sidebar in desktop version

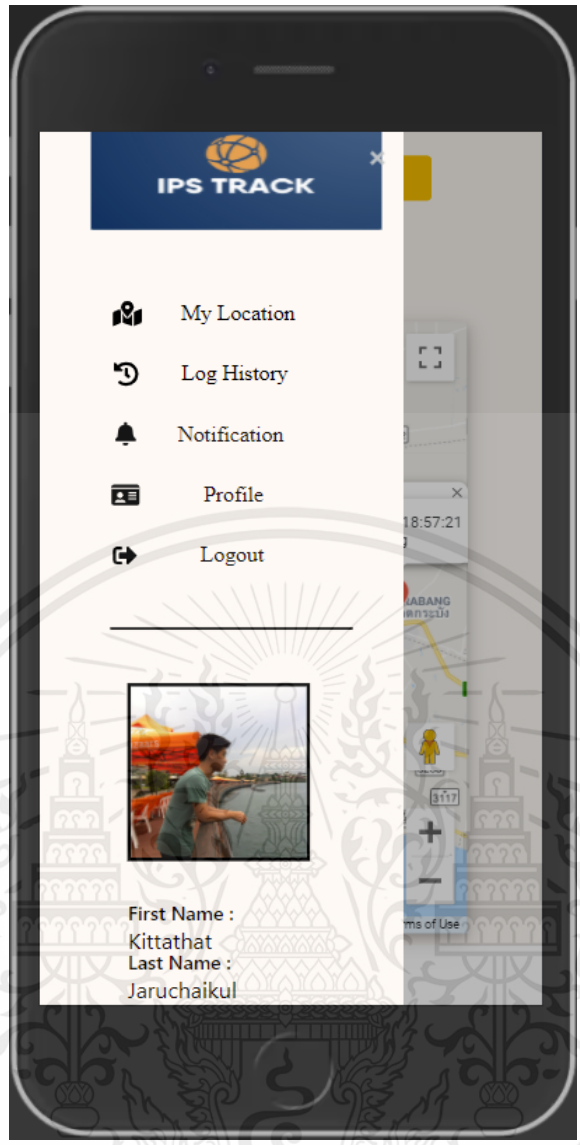

Figure 6.24: Sidebar in mobile version

Admin Page

Admin page works similarly to the outdoor map, but admin could access to see any user's information. Moreover, we intended to allow the admin to generate a timeline of contact of each user, to see who the user has been contacting to.

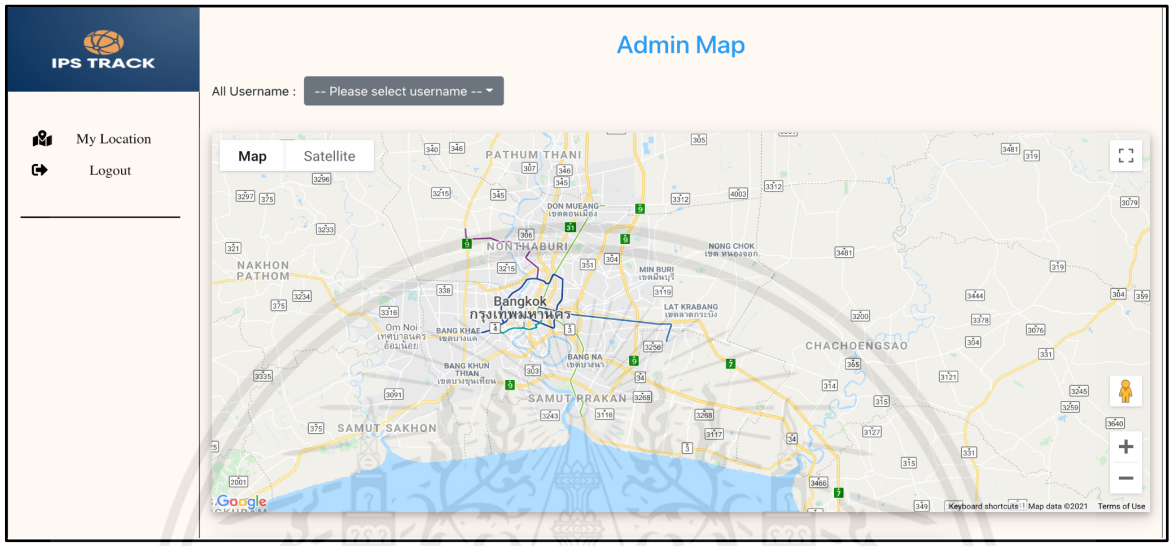

Figure 6.25: Admin Page (1) in desktop version

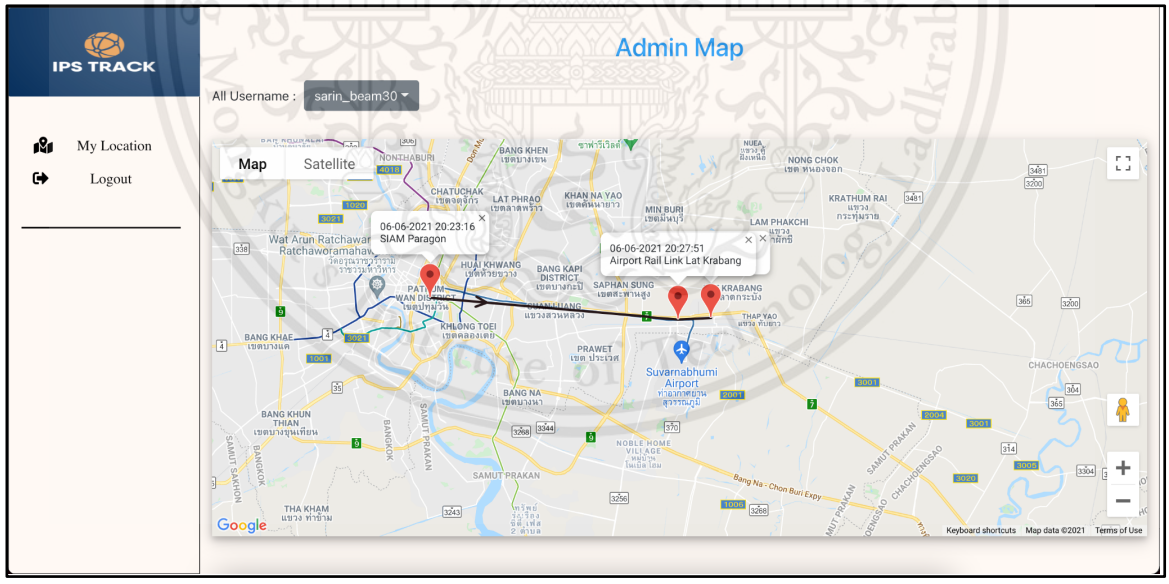

Figure 6.26: Admin Page (2) in desktop version

6.2.2 Backend part

6.2.2.1 Django Web Framework

Our Django web framework is deployed in a Heroku container for hosting. I used a docker container to set up the environment of the python version, all used-python package and Django web framework version. Furthermore, I have also used a Heroku Postgres with PostGIS extension as our database in the Django web framework.

Figure 6.27: Set-up environment of Django web framework by using Dockerfile

heroku.yml

```yaml
build:
  docker:
    web: Dockerfile
    worker: Dockerfile

run:
  web: daphne hello_django.asgi:application --port $PORT --bind 0.0.0.0 -v2
  worker: celery -A hello_django worker -B --pool=solo -l info

release:
  image: web
  command:
    - python manage.py collectstatic --noinput && python manage.py makemigrations && python manage.py migrate
```

Figure 6.28: Set-up environment of Django web framework by using heroku.yml

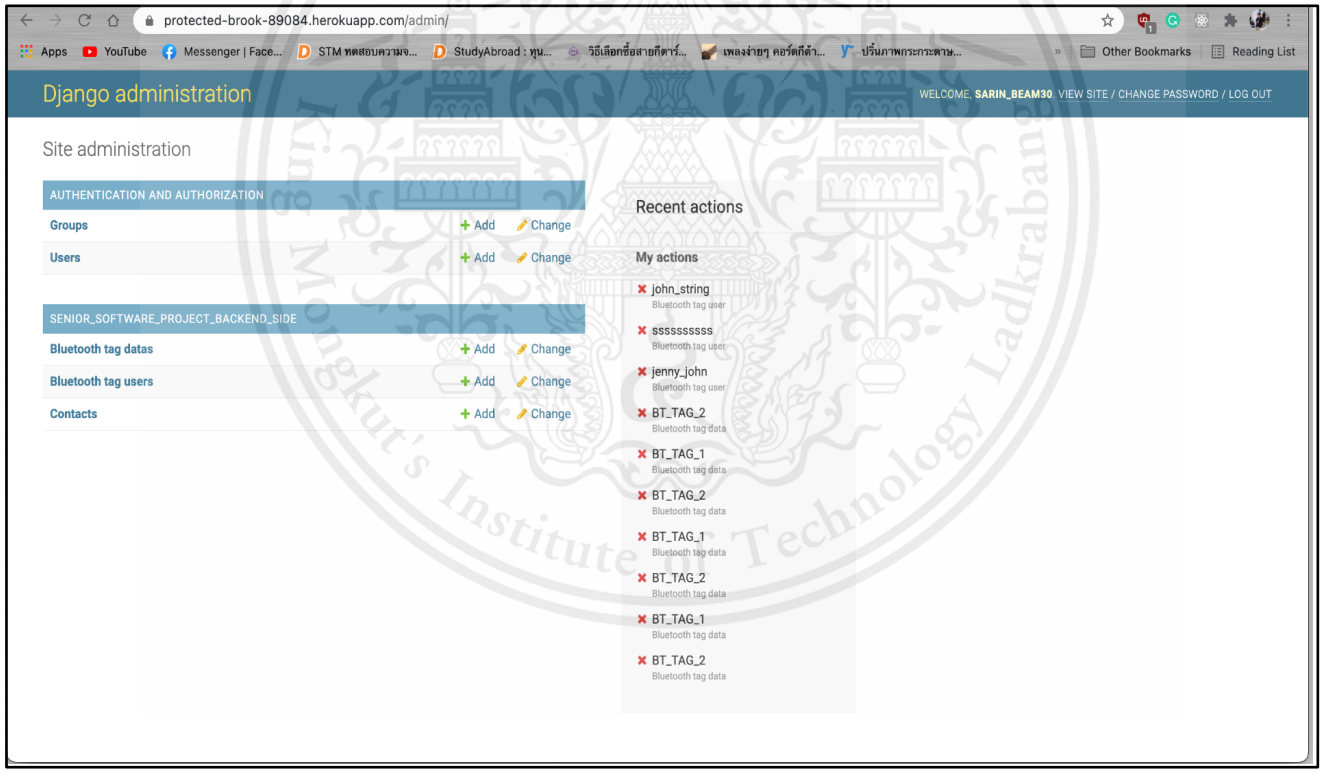

Figure 6.29: Django admin page on Heroku server

6.2.2.2 PostgreSQL

We have already set up PostgreSQL with PostGIS extension databases in the Heroku server for using contract tracing features, store and return the current position of the Bluetooth tag to the frontend part.

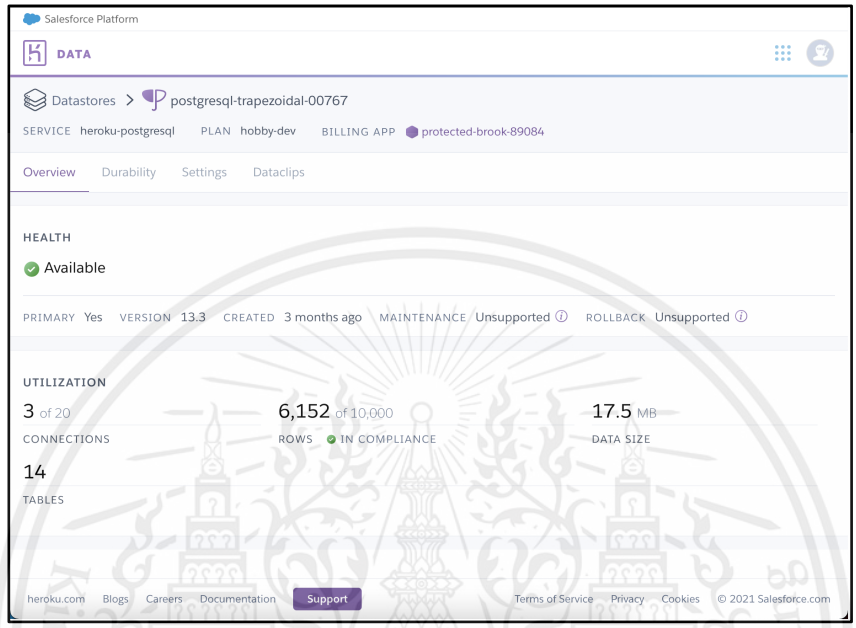

Figure 6.30: Heroku PostgreSQL database on the Django web framework

```
protected-brook-89084::DATABASE=> \dt;
                                List of relations
 Schema |                             Name                              | Type  |     Owner
--------+---------------------------------------------------------------+-------+----------------
 public | auth_group                                                    | table | iugcqkgnfuijoy
 public | auth_group_permissions                                        | table | iugcqkgnfuijoy
 public | auth_permission                                               | table | iugcqkgnfuijoy
 public | auth_user                                                     | table | iugcqkgnfuijoy
 public | auth_user_groups                                              | table | iugcqkgnfuijoy
 public | auth_user_user_permissions                                    | table | iugcqkgnfuijoy
 public | django_admin_log                                              | table | iugcqkgnfuijoy
 public | django_content_type                                           | table | iugcqkgnfuijoy
 public | django_migrations                                             | table | iugcqkgnfuijoy
 public | django_session                                                | table | iugcqkgnfuijoy
 public | senior_software_project_backend_side_bluetoothtagdata         | table | iugcqkgnfuijoy
 public | senior_software_project_backend_side_bluetoothtaguser         | table | iugcqkgnfuijoy
 public | senior_software_project_backend_side_contact                  | table | iugcqkgnfuijoy
 public | spatial_ref_sys                                               | table | postgres
(14 rows)
```

Figure 6.31: List all tables in the PostgreSQL database

```
protected-brook-89084::DATABASE=> \d+ senior_software_project_backend_side_bluetoothtagdata;
                                    Table "public.senior_software_project_backend_side_bluetoothtagdata"
       Column        |           Type           | Collation | Nullable | Default | Storage  | Stats target | Description
---------------------+--------------------------+-----------+----------+---------+----------+--------------+-------------
 BT_TAG_ID           | uuid                     |           | not null |         | plain    |              |
 BT_TAG_DEVICE_NAME  | character varying(20)    |           | not null |         | extended |              |
 BT_TAG_OWNER        | character varying(20)    |           | not null |         | extended |              |
 TIMESTAMP           | timestamp with time zone |           | not null |         | plain    |              |
 X_COORD             | double precision         |           | not null |         | plain    |              |
 Y_COORD             | double precision         |           | not null |         | plain    |              |
 LOCATION            | character varying(50)    |           | not null |         | extended |              |
 LATITUDE_LONGTITUDE | geography(Point,4326)    |           | not null |         | main     |              |
 FLOOR               | integer                  |           | not null |         | plain    |              |
 ROOM                | character varying(20)    |           | not null |         | extended |              |
Indexes:
    "senior_software_project_backend_side_bluetoothtagdata_pkey" PRIMARY KEY, btree ("BT_TAG_ID")
    "senior_software_project_backend_side_bluetoothtagdata_LATITUDE_" gist ("LATITUDE_LONGTITUDE")
Access method: heap
```

Figure 6.32: List all columns in the Bluetooth tag data table

```
protected-brook-89084::DATABASE=> \d+ senior_software_project_backend_side_bluetoothtaguser;
                                                                Table "public.senior_software_project_backend_side_bluetoothtaguser"
      Column      |          Type          | Collation | Nullable |                                  Default                                  | Storage  | Stats target |
Description
------------------+------------------------+-----------+----------+---------------------------------------------------------------------------+----------+--------------+-
------------
 id               | integer                |           | not null | nextval('senior_software_project_backend_side_bluetoothtaguser_id_seq'::regclass) | plain    |              |
 USER_ID          | text                   |           | not null |                                                                           | extended |              |
 IMAGE_PROFILE    | character varying(100) |           | not null |                                                                           | extended |              |
 DATE_OF_BIRTH    | date                   |           | not null |                                                                           | plain    |              |
 FIRST_NAME       | character varying(20)  |           | not null |                                                                           | extended |              |
 LAST_NAME        | character varying(20)  |           | not null |                                                                           | extended |              |
 GENDER           | character varying(20)  |           | not null |                                                                           | extended |              |
 HOME_ADDR        | text                   |           | not null |                                                                           | extended |              |
 USER_NAME        | character varying(20)  |           | not null |                                                                           | extended |              |
 PASSWORD         | character varying(20)  |           | not null |                                                                           | extended |              |
 COVID_19_LINEAGE | character varying(30)  |           | not null |                                                                           | extended |              |
 COVID_19_STATUS  | character varying(20)  |           | not null |                                                                           | extended |              |
Indexes:
    "senior_software_project_backend_side_bluetoothtaguser_pkey" PRIMARY KEY, btree (id)
Access method: heap
```

Figure 6.33: List all columns in the user table

```
protected-brook-89084::DATABASE=> \d+ senior_software_project_backend_side_contact;
                                                          Table "public.senior_software_project_backend_side_contact"
        Column         |           Type           | Collation | Nullable |                                  Default                                  | Storage  | Stats target |
 Description
------------------------+--------------------------+-----------+----------+---------------------------------------------------------------------------+----------+--------------+
-------------
 ID                     | integer                  |           | not null | nextval('"senior_software_project_backend_side_contact_ID_seq"'::regclass) | plain    |              |
 TIMESTAMP              | timestamp with time zone |           | not null |                                                                           | plain    |              |
 USER_1                 | text                     |           | not null |                                                                           | extended |              |
 USER_2                 | text                     |           | not null |                                                                           | extended |              |
 CONTACT_TRACING_LEVEL  | integer                  |           | not null |                                                                           | plain    |              |
 COVID_19_STATUS_USER_1 | text                     |           | not null |                                                                           | extended |              |
 COVID_19_STATUS_USER_2 | text                     |           | not null |                                                                           | extended |              |
 DISTANCE               | double precision         |           | not null |                                                                           | plain    |              |

Indexes:
    "senior_software_project_backend_side_contact_pkey" PRIMARY KEY, btree ("ID")
Access method: heap
```

Figure 6.34: List all columns in the contact table

6.2.2.3 PostGIS

Firstly, We use point-field type to store latitude and longitude of Bluetooth tag data in our database.

Secondly, We use a PostGIS library to calculate the distance between point A and point B or Bluetoothtag A and Bluetoothtag B. Because the distance is the one of all important factors to make contact tracing features.

```
def calculateDistanceBetweenTwoTags(x1, y1, x2, y2):
    p1 = Point(x1, y1, srid=3857)
    p2 = Point(x2, y2, srid=3857)
    # print("DISTANCE : " + str(p1.distance(p2)))
    return p1.distance(p2)
```

Figure 6.35: The function for calculating the distance between tag A and tag B

6.2.2.4 SCADA System

We use a SCADA system to send the current real-time position of Bluetooth tag to Django web server by using WebSocket protocol. Before we can use the SCADA system, we must make a worker first because we will use the worker as a receiver to receive all information of the Bluetooth tag. After that, we need to create tags in the SCADA system to store and display only part of the data which you want. For example, the "IndoorPositioningSystem.BT_TAG.x_coord" will only store the X coordinate value from the Bluetooth tag. Then, we will display these data on the SCADA dashboard with the SCADA tags system. Furthermore, the SCADA system will help us to send data from Raspberry Pi to our web server by using real-time protocols which are python socket and WebSocket protocol in the backend part.

```
worker_1    | [2021-03-16 20:16:54,215: WARNING/ForkPoolWorker-6] **** CONNECT TO WEB SOCKET LEAW ****
worker_1    | [2021-03-16 20:16:54,220: WARNING/ForkPoolWorker-6] RESULT TYPE :
worker_1    | [2021-03-16 20:16:54,221: WARNING/ForkPoolWorker-6] <class 'str'>
worker_1    | [2021-03-16 20:16:54,221: WARNING/ForkPoolWorker-6] JSON_DATA :
worker_1    | [2021-03-16 20:16:54,221: WARNING/ForkPoolWorker-6] {"BT_TAG_DEVICE_NAME": "BT_TAG_1", "LOCATION": "ABC Building", "LATITUDE": 1.1111, "LONGTITUD
E": 2.2222, "FLOOR": 7, "ROOM": "ECC-804", "X_COORD": 3.046, "Y_COORD": 7.89}
worker_1    | [2021-03-16 20:16:54,222: WARNING/ForkPoolWorker-6] {"BT_TAG_DEVICE_NAME": "BT_TAG_1", "LOCATION": "ABC Building", "LATITUDE": 1.1111, "LONGTITUD
E": 2.2222, "FLOOR": 7, "ROOM": "ECC-804", "X_COORD": 3.046, "Y_COORD": 7.89}
worker_1    | [2021-03-16 20:16:54,222: WARNING/ForkPoolWorker-6] TYPE OF DATA :
worker_1    | [2021-03-16 20:16:54,223: WARNING/ForkPoolWorker-6] <class 'str'>
worker_1    | [2021-03-16 20:16:54,223: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,223: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_1.x_coord.(10)
worker_1    | [2021-03-16 20:16:54,225: WARNING/ForkPoolWorker-6] x_coord : 3.046
worker_1    | [2021-03-16 20:16:54,227: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,230: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_1.y_coord.(11)
worker_1    | [2021-03-16 20:16:54,232: WARNING/ForkPoolWorker-6] y_coord : 7.89
worker_1    | [2021-03-16 20:16:54,232: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,233: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_1.BT_TAG_DEVICE_NAME.(20)
worker_1    | [2021-03-16 20:16:54,240: WARNING/ForkPoolWorker-6] BT_TAG_DEVICE_NAME : BT_TAG_1
worker_1    | [2021-03-16 20:16:54,324: WARNING/ForkPoolWorker-6] **** CONNECT TO WEB SOCKET LEAW ****
worker_1    | [2021-03-16 20:16:54,328: WARNING/ForkPoolWorker-6] RESULT TYPE :
worker_1    | [2021-03-16 20:16:54,329: WARNING/ForkPoolWorker-6] <class 'str'>
worker_1    | [2021-03-16 20:16:54,330: WARNING/ForkPoolWorker-6] JSON_DATA :
worker_1    | [2021-03-16 20:16:54,331: WARNING/ForkPoolWorker-6] {"BT_TAG_DEVICE_NAME": "BT_TAG_2", "LOCATION": "AAA Building", "LATITUDE": 1.1111, "LONGTITUD
E": 2.2222, "FLOOR": 7, "ROOM": "ECC-809", "X_COORD": 5.545, "Y_COORD": 1.375}
worker_1    | [2021-03-16 20:16:54,332: WARNING/ForkPoolWorker-6] {"BT_TAG_DEVICE_NAME": "BT_TAG_2", "LOCATION": "AAA Building", "LATITUDE": 1.1111, "LONGTITUD
E": 2.2222, "FLOOR": 7, "ROOM": "ECC-809", "X_COORD": 5.545, "Y_COORD": 1.375}
worker_1    | [2021-03-16 20:16:54,333: WARNING/ForkPoolWorker-6] TYPE OF DATA :
worker_1    | [2021-03-16 20:16:54,333: WARNING/ForkPoolWorker-6] <class 'str'>
worker_1    | [2021-03-16 20:16:54,334: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,334: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_2.x_coord.(13)
worker_1    | [2021-03-16 20:16:54,339: WARNING/ForkPoolWorker-6] x_coord : 5.545
worker_1    | [2021-03-16 20:16:54,342: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,344: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_2.y_coord.(14)
worker_1    | [2021-03-16 20:16:54,347: WARNING/ForkPoolWorker-6] y_coord : 1.375
worker_1    | [2021-03-16 20:16:54,348: WARNING/ForkPoolWorker-6] TAG :
worker_1    | [2021-03-16 20:16:54,350: WARNING/ForkPoolWorker-6] IndoorPositioningSystem.BT_TAG_2.BT_TAG_DEVICE_NAME.(28)
worker_1    | [2021-03-16 20:16:54,352: WARNING/ForkPoolWorker-6] BT_TAG_DEVICE_NAME : BT_TAG_2
```

Figure 6.36: Celery python worker in the SCADA system

Django administration

WELCOME, -. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Tag › Tags

Select tag to change

ADD TAG +

Action: -------- Go 0 of 12 selected

	ID	DEVICE	NAME	FULL NAME	ADDRESS	VALUE TYPE	TYPE	LOGABLE	CONTI
☐	10	BT_TAG_1@192.168.4.150(5)	x_coord	IndoorPositioningSystem.BT_TAG_1.x_coord	X_COORD	str	X_COORD	✔	
☐	11	BT_TAG_1@192.168.4.150(5)	y_coord	IndoorPositioningSystem.BT_TAG_1.y_coord	Y_COORD	str	Y_COORD	✔	
☐	20	BT_TAG_1@192.168.4.150(5)	BT_TAG_DEVICE_NAME	IndoorPositioningSystem.BT_TAG_1.BT_TAG_DEVICE_NAME	BT_TAG_DEVICE_NAME	str	BT_TAG_DEVICE_NAME	✔	
☐	29	BT_TAG_1@192.168.4.150(5)	BT_TAG_OWNER	IndoorPositioningSystem.BT_TAG_1.BT_TAG_OWNER	BT_TAG_OWNER	str	BT_TAG_OWNER	✔	
☐	30	BT_TAG_1@192.168.4.150(5)	BT_TAG_LOCATION	IndoorPositioningSystem.BT_TAG_1.BT_TAG_LOCATION	LOCATION	str	LOCATION	✔	
☐	31	BT_TAG_1@192.168.4.150(5)	BT_TAG_LATITUDE	IndoorPositioningSystem.BT_TAG_1.BT_TAG_LATITUDE	LATITUDE	str	LATITUDE	✔	
☐	32	BT_TAG_1@192.168.4.150(5)	BT_TAG_LONGTITUDE	IndoorPositioningSystem.BT_TAG_1.BT_TAG_LONGTITUDE	LONGTITUDE	str	LONGTITUDE	✔	
☐	33	BT_TAG_1@192.168.4.150(5)	FLOOR	IndoorPositioningSystem.BT_TAG_1.FLOOR	FLOOR	str	FLOOR	✔	
☐	34	BT_TAG_1@192.168.4.150(5)	ROOM	IndoorPositioningSystem.BT_TAG_1.ROOM	ROOM	str	ROOM	✔	

Figure 6.37: Tags system in the SCADA system

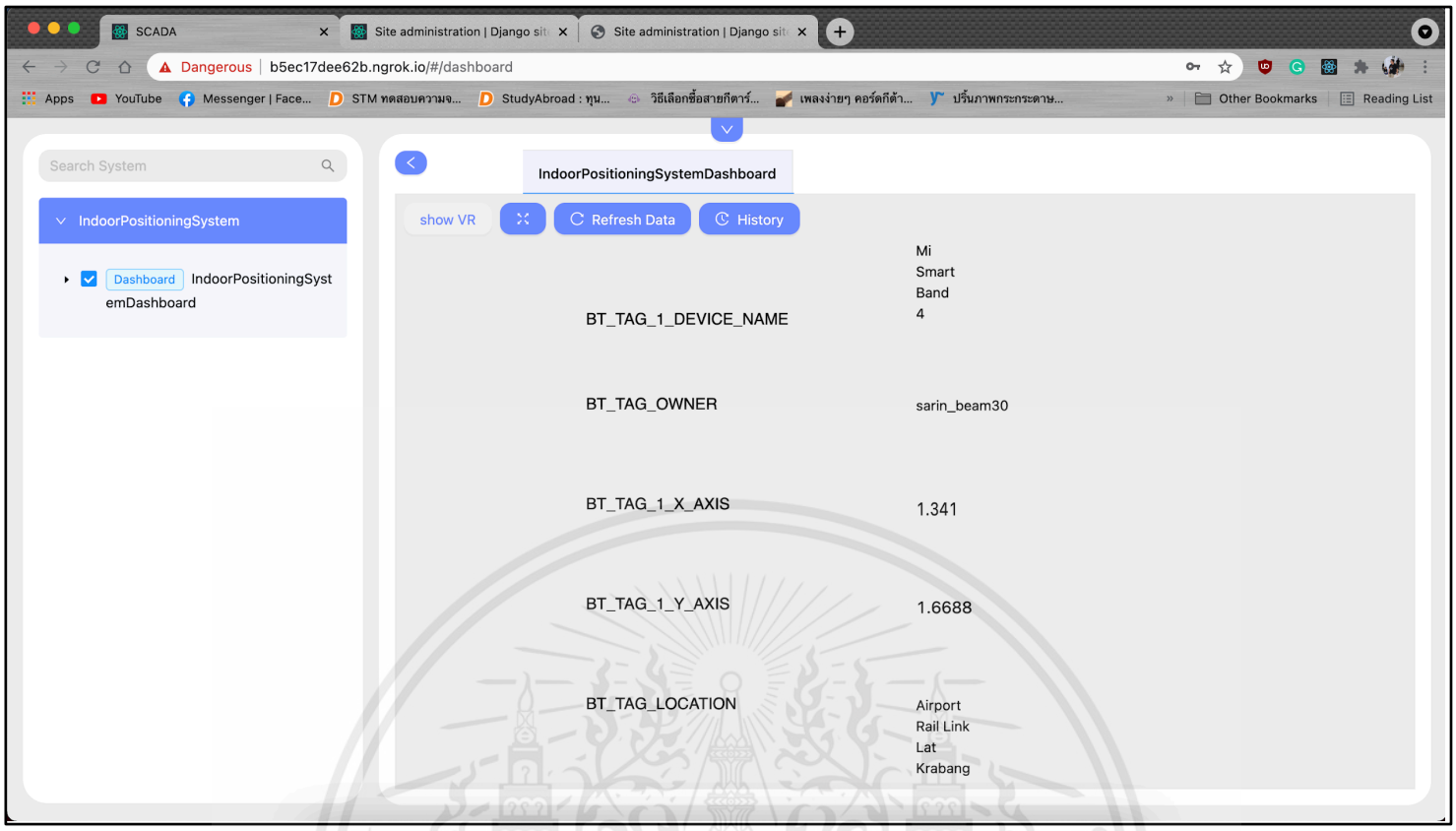

Figure 6.38: IPS dashboard in the SCADA system

6.2.2.5 List of all API paths and WebSocket paths

Path	HTTP Method	Action
API/location/	GET	Return all Bluetooth tag data in the database
API/contact/	GET	Return all contact data in the database
getAllCurrentUser/	GET	Return only active user(s) in the system
userRegistration/	POST	Register new user and save in the database
userLogin/	POST	Validate username and password. Then, return the result
userLogout/	POST	Clear active user's state from the system
userAndAdminInformation/	POST	Get user information which depend on user's token
userAndAdminInformation/	GET	Return all username which have a BluetoothTagData
userOutdoor/	POST	Return all out door location that rely on user token or username (Admin page)
updateUserInformation/	POST	Update user information

Figure 6.39: List of all API paths

Path	Method	Action
ws/location/	send	Return latest current location with given username
ws/contact/	send	Return only contact data with specificed username

Figure 6.40: List of all WebSocket paths

Chapter 7

Conclusion

We break down our conclusion into 3 parts: hardware, backend, and frontend.

Hardware

The Raspberry Pis have been configured to scan for the RSSI of multiple target devices. The scanned data can be used to calculate the (x, y) coordinates of the target devices and the data can be sent to the backend. However, consistency of the measured RSSI can still be improved. Currently, relying solely on RSSI is a viable method for indoor positioning if the concern isn't about absolute pinpoint accuracy. RSSI is highly unstable, since many factors, such as the invisible interfering radio signals on the same wavelength, the orientation of the transmitting/receiving sensor, direct line of sight, or signal reflecting off of surfaces, can easily cause it to fluctuate. With only 1 target, it is fairly reliable, with less than 25cm margin of error on average within a 3-meter enclosure. As the count of devices increases, the margin of error also increases (we observed to be almost a 50cm margin of error on average). If the margin of error below 1m is okay, then BLE and RSSI is enough for the use case. However, if more time is spent on fine tuning or trying different methods of filtering, compensating for the path loss, noise, other interferences, or when coupled with other technologies such as ultra-wideband or infrared, then RSSI would be more accurate indoor positioning.

Backend

For the backend part, I can send data or JSON data from Raspberry Pi to the SCADA server to show on the SCADA dashboard. For example, I can show the Bluetooth tag name, x-coordinate and y-coordinate on the SCADA dashboard. Furthermore, I currently have a separate Django server for analyzing the current position of the Bluetooth tag of every user in our system and returning all data which the Frontend needs.

Frontend

Front-end, we have implemented the website and deployed the website on GitHub. The website is able to display the A-frame map that will be used to show Bluetooth tags. We have also created connections between the server and client side (SCADA and website). After receiving the data, we can show the data on the map and test for its accuracy.

7.2 Future Work

Our plan for the future is to see if there's still a way to implement Bluetooth 5.1's direction finding as this would be a lot more accurate and consistent in location finding. Bluetooth 5.1 and 5.2 products have started to come out recently, and it could be a big improvement on the side of accuracy.

We will continue developing the data analysis aspects of our project in order to do contact tracing. If possible, we also planned to expand our project beyond the local scope and try using GPS for outdoor tracking.

The website will also be improved to show the target's location on the map in real time. A better method for tying each user to their own Bluetooth tag could also be implemented in the future.

Bibliography

[1] P. Sambu and M. Won. (2021, 6 March). *An Experimental Study on Direction Finding of Bluetooth 5.1: Indoor vs Outdoor* [Online]. Available: https://arxiv.org/pdf/2103.04121.pdf

[2] M. Brown, J Pinchin, and C. Hide. (2013, April). *Opening Indoors: The Advent of Indoor Positioning* [Online]. Available: https://www.researchgate.net/publication/258729318_Opening_Indoors_The_Advent_of_Indoor_Positioning

[3] J. Kunhoth, A. Karkar, S. AI-Maadeed and A. AI-Ali. (2020, May 02). Indoor positioning and wayfinding systems: a survey [Online]. Available: https://hcis-journal.springeropen.com/track/pdf/10.1186/s13673-020-00222-0.pdf

[4] C. Terawong, S. Choosakunwiboon, and S. Suttisirikul. (2018). Device-Context Based Indoor Localization. Available: International College King Mongkut's Institute of Technology Ladkrabang (KMITL-2018-IC-B-003-010).

[5] F. Shang, W. Su, Q. Wang, H. Gao and Q. Fu. (2014, August 31).
A Location Estimation Algorithm Based on RSSI Vector Similarity Degree [Online]. Available: https://journals.sagepub.com/doi/full/10.1155/2014/371350

[6] Mukund. (2016, October 7). How to Calculate Distance from the RSSI value of the BLE Beacon [Online]. Available: https://iotandelectronics.wordpress.com/2016/10/07/how-to-calculate-distance-from-the-rssi-value-of-the-ble-beacon/

[7] Cell Phone Trilateration Algorithm [Online]. (2019, March 29). Available: https://www.101computing.net/cell-phone-trilateration-algorithm/

[8] Jithin. (n.d.). Arduino BLE Example Explained Step by Step [Online]. Available: https://rootsaid.com/arduino-ble-example/

[9] M Afaneh. (2020, April 6). Bluetooth Addresses & Privacy in Bluetooth Low Energy [Online]. Available: https://www.novelbits.io/bluetooth-address-privacy-ble/

[10] B. Esme. (2009, March 26). Kalman Filter For Dummies [Online]. Available: http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies

[11] Umofomia. (2021, April 26). Kalman filter [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter

[12] N. Biswas. (2020, October 4). ReactJS Tutorial for Beginners -7 [Online]. Available: https://nabendu82.medium.com/reactjs-tutorial-for-beginners-7-5e2210b080e7

[13] W3C Community Group. (2021, May 17). Web Bluetooth Draft Community Group Report [Online]. Available: https://webBluetoothcg.github.io/web-Bluetooth/

[14] The PostGIS Project Steering Committee (PSC). (2021, March 05). PostGIS 2.4.10dev Manual [Online]. Available: https://postgis.net/stuff/postgis-2.4.pdf

[15] The Internals of PostgreSQL for database administrators and system developers [Online]. (n.d.). Available: https://www.interdb.jp/pg/pgsql01.html#:~:text=A%20PostgreSQL%20server%20runs%20on,store%20or%20to%20reference%20data.

[16] B. Garner. (2019, January 26). Deploying Django to Heroku: Connecting Heroku Postgres [Online]. Available: https://blog.usejournal.com/deploying-django-to-heroku-connecting-heroku-postgres-fcc960d290d1

[17] M. Herman. (2020, September 10). Deploying Django to Heroku With Docker [Online]. Available: https://testdriven.io/blog/deploying-django-to-heroku-with-docker/

Appendix

Comparison with other related works

Feature / Technology	Here	TI	COVID-Alert Application	Our Project
Wi-Fi	Yes	No	No	No
Bluetooth 5.1	No	Yes	No	No
Bluetooth Beacon (Bluetooth 4.0)	Yes	No	Yes	Yes
AoA (Angle of Arrival) technology	Not tell	Yes	No	No
Antenna	No	Yes	No	No
Indoor Map Mobile Application	Yes	No	No	No
Indoor Map Web Application	No	No	No	Yes
Contact Tracing Feature	No	No	Yes	Yes

Table 1: Our Project vs Related Companies

Calibration Results

3T Calibration	Mode (dBm)	Mean (dBm)	Kalman (dBm)
1	-71	-71.75	-72.79718751
2	-71	-71.55	-71.32422653
3	-70	-72.35	-72.95267772
4	-71	-72.05	-71.24104042
5	-74	-72	-71.03274129
6	-71	-71.68	-71.69683056
7	-71	-71.2	-71.71002751
8	-74	-72.45	-72.88827464
9	-71	-71.65	-71.73870703
10	-71	-71.95	-72.30413876

Table 2: BLE Peripheral Calibration: (OnePlus) 3T

X50 Calibration	Mode (dBm)	Mean (dBm)	Kalman (dBm)
1	-74	-77.47058823...	-76.08621879...
2	-75	-75.111...	-75.65245128...
3	-78	-76.05	-76.94000887...
4	-78	-75.9	-75.99233388...
5	-75	-75.52631578...	-75.30593673...
6	-78	-75.85	-76.36538196...
7	-75	-75.52631578...	-75.74205653...
8	-73	-75.5	-74.82939403...
9	-78	-75.36842105...	-74.00747668...
10	-75	-75.36842105...	-74.66347425...

Table 3: BLE Peripheral Calibration: (Realme) X50-5G

Mi4 Calibration	Mode (dBm)	Mean (dBm)	Kalman (dBm)
1	-66	-66.47610947...	-66.75898839...
2	-66	-67	-67.89440437...
3	-65	-67.333...	-67.72617685...
4	-65	-66.30952328...	-66.54502389...
5	-69	-67.5	-68.20834119...
6	-66	-67.65853658...	-67.79388999...
7	-65	-66.72093023...	-65.82668594...
8	-65	-66.55813953...	-66.50849104...
9	-69	-67.28571429	-67.30029996...
10	-65	-66.1818..	-66.39314645...

Table 4: BLE Peripheral Calibration: Mi Smart Band 4

Trilateration Results

	Tag1 = X50	
Real Location	x = 1.5	y = 1.5
Test no.	Calculated x	Calculated y
1	1.371	1.1275
2	1.3413	1.2629
3	1.486	1.3538
4	1.4659	1.2163
5	1.5807	1.5467
Average Difference	0.0861m	0.2248m

Table 5: 1 Peripheral Location Test Results

	T1 = MiBand		T2 = 3T	
Real Location	x = 0	y = 1	x = 2	y = 2
Test no.	Calculated x	Calculated y	Calculated x	Calculated y
1	-0.18	1.25	1.5417	1.984
2	-0.9812	1.229	2.111	2.3427
3	-0.2299	1.1887	1.977	1.7595
4	-1.132	1.093	1.5621	1.712
5	-0.2905	1.2717	2.348	2.1532
Average Difference	0.362m	0.206m	0.2763m	0.2085m

Table 6: 2 Peripherals Location Test Results

	T1 = X50		T2 = 3T		T3 = MiBand	
Real Location	x = 1	y = 1	x = 2	y = 1	x = 1	y = 2
Test no.	Calc. X	Calc. Y	Calc. X	Calc. Y	Calc. X	Calc. Y
1	0.574	1.5644	2.139	1.1688	1.5717	1.3609
2	-0.1352	0.1775	1.9252	0.4038	1.1693	1.5635
3	1.8622	0.6986	2.4069	0.8374	1.2184	1.483
4	1.0913	1.0709	2.858	1.8827	1.5667	1.5873
5	1.16	0.6015	1.8839	1.3208	1.409	1.6702
Average diff	0.4822m	0.3045m	0.3216m	0.4263m	0.3867m	0.468m

Table 7: 3 Peripherals Location Test 1 Results

	T1 = X50		T2 = 3T		T3 = MiBand	
Real Location	x = 0.5	y = 0.5	x = 1	y = 1	x = 3	y = 3
Test no.	Calc. X	Calc. Y	Calc. X	Calc. Y	Calc. X	Calc. Y
1	-0.6722	0.312	1.0852	1.1932	2.5745	2.6867
2	0.8295	-0.1496	0.8789	1.5545	2.9481	2.5251
3	0.503	1.0273	1.2087	-0.1905	2.3049	2.8493
4	0.2304	0.7673	1.5823	0.8903	2.6402	2.7015
5	0.1724	0.5713	1.8569	0.5994	2.7908	2.6082
Average diff	0.4199m	0.3393m	0.3733m	0.2880m	0.3521m	0.3782m

Table 8: 3 Peripherals Location Test 2 Results

	T1 = X50		T2 = 3T		T3 = MiBand	
Real Location	x = 1	y = 0	x = 1	y = 3	x = 3	y = 1.5
Test no.	Calc. X	Calc. Y	Calc. X	Calc. Y	Calc. X	Calc. Y
1	1.2307	-0.1755	1.6964	3.4077	2.9104	1.9956
2	0.4575	0.7673	1.2429	3.8002	2.4399	2.2803
3	0.8126	-0.4658	0.7338	3.2488	2.6571	1.8837
4	1.4066	0.3292	0.9044	3.2953	2.5696	1.4303
5	1.7267	-0.3676	0.6448	3.5391	2.3829	1.9246
Average Diff	0.4213m	0.4175m	0.332m	0.4538m	0.4085m	0.4289m

Table 9: 3 Peripherals Location Test 3 Results