Department of Computer Science

Graduation Project

Final Report

Project Title:

Indoor Positioning System Based on Bluetooth

Project Team:

Seif Mortada Metwally

Kirolos Melad Ramsis

Abdalla Saad Emam

Kirolos Sobhy Haleem

Mohamed Reda Kotob

Peter Sameh Rushdy

Omar Ezzat Abd El-Hai

Academic supervisor:

Dr. \ Tarek Salah Abd El-Azeem

Assisting supervisor:

Eng. \ Esraa Ezzat

July 2023

Abstract

This documentation describes the development of an indoor positioning system (IPS) using Bluetooth technology. The project aims to address the need for an accurate, easy-to-use IPS that leverages Bluetooth beacons to provide precise location data for mobile devices.

The documentation includes information on the hardware and software components of the system. The hardware component involves designing and building Bluetooth beacons that can be easily deployed throughout an indoor environment. The software component involves developing algorithms for accurately calculating the position of mobile devices based on the signals received from the Bluetooth beacons.

The system is designed to provide real-time location data with sub-meter accuracy, making it suitable for a wide range of location-based services in various settings, including retail, healthcare, and transportation. The system is also designed to be easy to integrate into existing mobile applications, allowing developers to quickly add indoor positioning capabilities to their applications.

The documentation includes technical specifications for the hardware and software components of the system, as well as instructions for deploying and configuring the system. Additionally, the documentation includes information on testing and validation procedures to ensure that the system meets its accuracy and performance requirements.

Overall, this documentation provides a comprehensive guide to the development of an indoor positioning system using Bluetooth technology. By following the instructions provided in this documentation, developers can build their own Bluetooth-based IPS and integrate it into their applications to provide accurate indoor navigation and location-based services.

Acknowledgements

This work would not have been possible without the academic education, directive support, and other factors that are too long to be mentioned by the department of Computer and Information Technology in El-Shorouk Academy. We are especially indebted to Dr. Tarek salah Abd El-Azeem, our supervisor and mentor, and Eng. Esraa Ezzat, our assistant supervisor, who have been supportive of our academic goals and who worked actively to provide us with the protected academic time to pursue those goals. We are grateful to all of those with whom we had the pleasure to work during this project and research phases. Both gave us all the help we needed to make this project a reality, and a paper will not be enough to show our gratitude towards both aforementioned factors of our success.

Table of Contents

Contents Page

Chapter 1 : Introduction and Planning

1.1 Problem Definition:

Indoor navigation and location-based services are becoming increasingly important in a variety of settings, including retail, healthcare, and transportation. However, current indoor positioning systems often rely on expensive hardware or require complex installation processes. Additionally, many existing systems have limited accuracy or are not able to provide real-time location data.

To address these challenges, a project is proposed to develop an indoor positioning system using Bluetooth technology. The system will leverage Bluetooth beacons placed throughout an indoor environment to provide precise location data for mobile devices. By using Bluetooth technology, the system can provide accurate location data without requiring complex installation processes or expensive hardware.

The proposed system will be designed to be highly accurate, providing real-time location data with sub-meter accuracy. Additionally, the system will be easy to integrate into existing mobile applications, allowing developers to quickly add indoor positioning capabilities to their applications.

The project will involve developing both hardware and software components. The hardware component will involve designing and building Bluetooth beacons that can be easily deployed throughout an indoor environment. The software component will involve developing algorithms for accurately calculating the position of mobile devices based on the signals received from the Bluetooth beacons.

Overall, the proposed project aims to address the need for an accurate, easy-to-use indoor positioning system that leverages Bluetooth technology. By providing real-time location data with sub-meter accuracy, the system can enable a wide range of location-based services in a variety of settings.

1.2 Project objectives

- Develop an IPS using Bluetooth technology that is low-cost and easy to deploy

- Leverage inexpensive Bluetooth beacons and open-source software to provide accurate indoor navigation and location-based services.
- Address the need for an affordable IPS that can be easily deployed in a variety of settings.
- Provide real-time location data with sub-meter accuracy, suitable for a wide range of location-based services.
- Develop both hardware and software components, including low-cost Bluetooth beacons and algorithms for calculating device position.
- Make indoor navigation and location-based services accessible to a wider range of businesses and organizations by providing an affordable IPS solution.

1.3 Expected Time Plan

Plans / Weeks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
System Analysis	■	■	■	■	■	■	■																	
Research	▨	▨	▨	▨	▨																			
Requirement						▨																		
UML Diagrams							▨																	
System Design								■	■	■	■													
System Architecture								▨																
UI Design									▨															
Software Design										▨	▨													
Implementation												■	■	■	■	■	■	■	■	■				
Front End												▨	▨	▨										
Mobile App												▨	▨	▨	▨	▨	▨							
Beacons												▨	▨	▨	▨									
Desktop App												▨	▨	▨	▨	▨								
System Integration																		▨	▨	▨				
System Testing																					■	■	■	■
Unit Test																					▨			
UI Test																						▨		
Integration Test																							▨	▨

1.4 Indoor Positioning System (IPS)

1.4.1 Definition

An indoor positioning system (IPS) is a network of devices used to locate people or objects where GPS and other satellite technologies lack precision or fail entirely, such as inside multistory buildings, airports, alleys, parking garages, and underground locations.

A large variety of techniques and devices are used to provide indoor positioning ranging from reconfigured devices already deployed such as smartphones, WiFi and Bluetooth antennas, digital cameras, and clocks; to purpose built installations with relays and beacons strategically placed throughout a defined space. Lights, radio waves, magnetic fields, acoustic signals, and behavioral analytics are all used in IPS networks. IPS can achieve position accuracy of 2 cm, which is on par with RTK enabled GNSS receivers that can achieve 2 cm accuracy outdoors. IPS use different technologies, including distance measurement to nearby anchor nodes (nodes with known fixed positions, e.g. WiFi / LiFi access points, Bluetooth beacons or Ultra-Wideband beacons), magnetic positioning, dead reckoning. They either actively locate mobile devices and tags or provide ambient location or environmental context for devices to get sensed. The localized nature of an IPS has resulted in design fragmentation, with systems making use of various optical, radio, or even acoustic technologies.

IPS has broad applications in commercial, military, retail, and inventory tracking industries. There are several commercial systems on the market, but no standards for an IPS system. Instead each installation is tailored to spatial dimensions, building materials, accuracy needs, and budget constraints.

For smoothing to compensate for stochastic (unpredictable) errors there must be a sound method for reducing the error budget significantly. The system might include information from other systems to cope for physical ambiguity and to enable error compensation. Detecting the device's orientation (often referred to as the compass direction in order to disambiguate it from smartphone vertical orientation) can be achieved either by detecting landmarks inside images taken in real time, or by using trilateration with beacons. There also exist technologies for detecting magneto metric information inside buildings or locations with steel structures or in iron ore mines.

1.4.2 Applicability and precision :

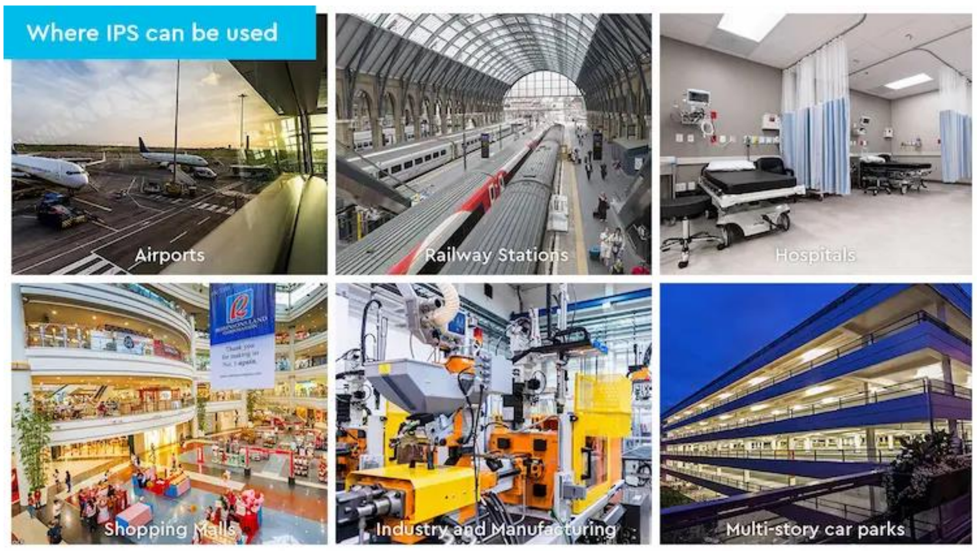
Due to the signal attenuation caused by construction materials, the satellite based Global Positioning System (GPS) loses significant power indoors affecting the required coverage for receivers by at least four satellites. In addition, the multiple reflections at surfaces cause multi-path propagation serving for uncontrollable errors. These very same effects are degrading all known solutions for indoor locating which uses electromagnetic waves from indoor transmitters to indoor receivers. A bundle of physical and mathematical methods are applied to compensate for these problems. Promising direction radio frequency positioning error correction opened by the use of alternative sources of navigational information, such as inertial measurement unit (IMU), monocular camera Simultaneous localization and mapping (SLAM) and WiFi SLAM. Integration of data from various navigation systems with different physical principles can increase the accuracy and robustness of the overall solution.

The U.S. Global Positioning System (GPS) and other similar Global navigation satellite systems (GNSS) are generally not suitable to establish indoor locations, since microwaves will be attenuated and scattered by roofs, walls and other objects. However, in order to make the positioning signals become ubiquitous, integration between GPS and indoor positioning can be made.

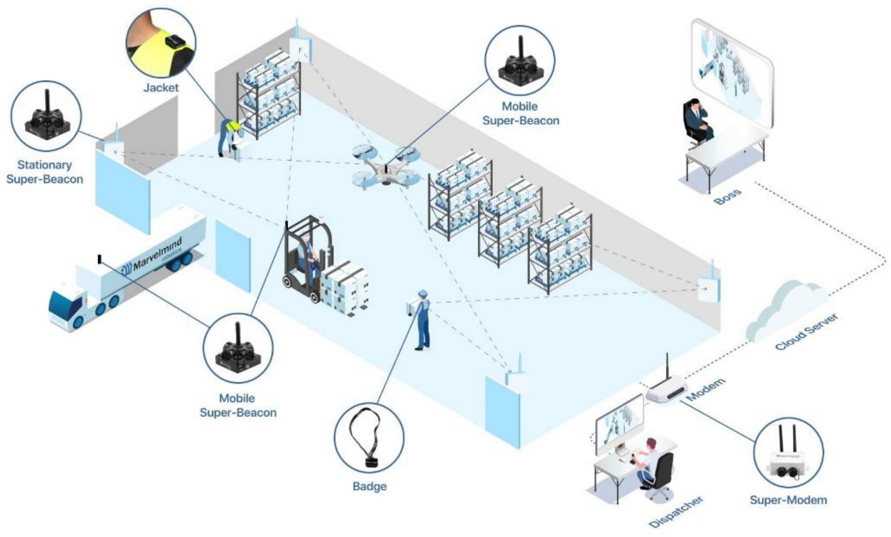
Currently, GNSS receivers are becoming more and more sensitive due to increasing microchip processing power. High Sensitivity GNSS receivers are able to receive satellite signals in most indoor environments and attempts to determine the 3D position indoors have been successful. Besides increasing the sensitivity of the receivers, the technique of A-GPS is used, where the almanac and other information are transferred through a mobile phone.

However, despite the fact that proper coverage for the required four satellites to locate a receiver is not achieved with all current designs (2008–11) for indoor operations, GPS emulation has been deployed successfully in Stockholm metro. GPS coverage extension solutions have been able to provide zone-based positioning indoors, accessible with standard GPS chipsets like the ones used in smartphones.

An indoor positioning system (IPS) is a network of devices used to locate people or objects where GPS and other satellite technologies lack precision or fail entirely, such as inside multistorey buildings, airports, alleys, parking garages, and underground locations.



A large variety of techniques and devices are used to provide indoor positioning ranging from reconfigured devices already deployed such as smartphones, Wi-Fi and Bluetooth antennas, digital cameras, and clocks; to purpose-built installations with relays and beacons strategically placed throughout a defined space.



Lights, radio waves, magnetic fields, acoustic signals, and behavioral analytics are all IPS can achieve a position accuracy of 2 cm, which is on par with RTK-enabled GNSS receivers that can achieve 2 cm accuracy outdoors.

IPS uses different technologies, including distance measurement to nearby anchor nodes (nodes with known fixed positions, e.g., Wi-Fi / Li-Fi access points, Bluetooth beacons, or Ultra-Wideband beacons), magnetic positioning, and dead reckoning.

They either actively locate mobile devices and tags or provide ambient location or environmental context for devices to get sensed.

The localized nature of an IPS has resulted in design fragmentation, with systems making use of various optical, radio, or even acoustic technologies.

Chapter 2 : Technology and Literature Survey

2.1 Similar Project :

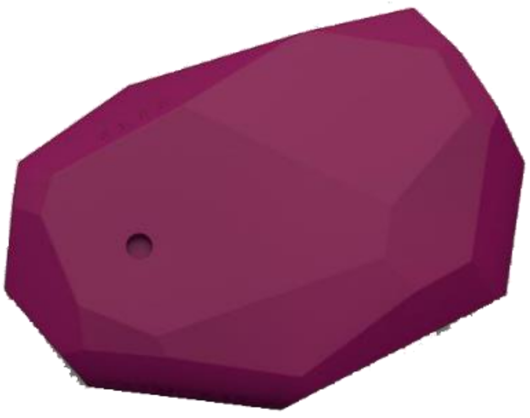
Project 1 : Estimote Indoor Location Technology

Features:

- Accurate indoor positioning and tracking using Bluetooth Low Energy (BLE) beacons
- Machine learning algorithms to improve accuracy over time
- Integration with other systems such as building management or security systems
- Compatibility with both iOS and Android devices
- Range of hardware options including beacons, stickers, and wearables
- Range of software tools and APIs for developers to build custom applications

Tools and technologies:

- Estimote Indoor Location SDK for iOS and Android
- Estimote Cloud for managing beacons and data
- Estimote Monitoring for real-time location tracking and event detection
- Machine learning algorithms for improving accuracy



Estimote Device

Project 2 : Aeroscout Wi-Fi Positioning

Features:

- Indoor positioning and tracking using Wi-Fi access points
- No additional hardware required, as most smartphones and mobile devices already have Wi-Fi capabilities
- Integration with other systems such as building management or security systems
- Compatibility with both iOS and Android devices
- Range of software tools and APIs for developers to build custom applications

Tools and technologies:

- Wi-Fi access points for location detection
- Wi-Fi fingerprinting for accurate location tracking
- Machine learning algorithms for improving accuracy

Aeroscout Device

Project 3 : Nanotron UWB Positioning

Features:

- Indoor positioning and tracking using Ultra-Wideband (UWB) technology
- High accuracy location tracking with centimeter-level precision
- Integration with other systems such as building management or security systems
- Compatibility with both iOS and Android devices
- Range of hardware options including tags, anchors, and gateways
- Range of software tools and APIs for developers to build custom applications

Tools and technologies:

- nanotron swarm bee family of tags and anchors
- nanotron swarm gateway for data collection and management
- UWB technology for high accuracy location tracking
- Machine learning algorithms for improving accuracy

Nanotron Device

Table For Comparison of similar projects

Project	Estimote Indoor Location Tech.	Aeroscout Wi-Fi Positioning	Nanotron UWB Positioning
Technology	BLE	Wi-Fi	UWB
Accuracy	High	Medium	High
Range	Short.	Medium	Short
Cost	Medium Beacons around 40$	High Access Points are expensive	High Anchors are expensive
Power Consumption	Low beacons can last up to 5 years on a single battery	High access points require power and network connectivity	Low anchors can last several years on a single battery
Infrastructure Required	Beacons (can be easily installed	Access Points (require power and network connectivity	Anchors (require installation and calibration)
Advantages	• Easy to install • low power consumption • high accuracy • good for small areas • can work with existing mobile devices.	• Can cover larger areas than BLE • Can work with existing Wi-Fi infrastructure.	• High accuracy, good for tracking moving objects • Can work in harsh environments
Disadvantage	• Limited range compared to Wi-Fi and UWB, may require a large number of beacons for larger areas	• Higher cost than BLE, may require additional infrastructure for larger areas.	• Higher cost than BLE and Wi-Fi, requires installation and calibration of anchors.

2.2 Similar Technologies :

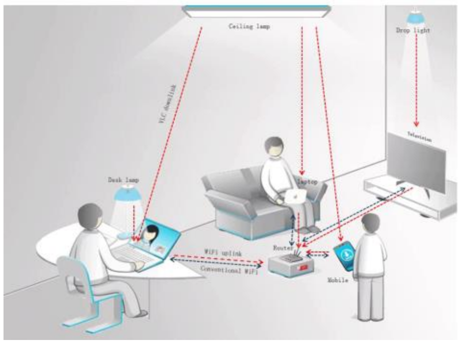
Several numbers of technologies have been imposed by the researchers for indoor positioning systems within past years. Among those technologies, there most common technologies have been identified such as visible light communication (VLC) that uses optical technologies, ultrasound which uses sound-based technologies, Wi-Fi, Bluetooth, Radio Frequency Identification (RFID) and ultra-wideband that uses radio frequency technologies. In this section, a brief explanation on the technologies has been reviewed.

1. Visible light communication (VLC)

Light signal can be classified in both infrared light and visible. Light signal is also an electromagnetic signal same as the RF signal, but its associated technologies might not be same. The advantages and challenges for optical technologies differ. For instance, line of sight is the main problem for optical signals.

Visible light communications (VLC), a technology in which the visible spectrum is modulated to transmit data. Due to the propagation distance of the light emitting diodes (LEDs), VLC is a short-range communication technology. VLC technology has been practiced for indoor positioning system purposes for the past few years. The system reuses current lighting infrastructures, thus can be deployed easily and fast inside buildings where LEDs are used for lighting.

Briefly, each light emitting diode has a different flicker encoding, thus the sensors held by the users will easily compare the modulation over the known encoding scheme. Then, it determines the presiding and associates the corresponding lamp.



There are few advantages of using VLC technology for indoor positioning systems such as lamps are extensively and homogenously available in buildings, as well as beacons can complement the positioning in indoor environment.

Furthermore, VLC Technology is considered as one of the efficient technologies to be implemented for the indoor positioning system because the positioning technology does not depend on batteries.

VLC Technology also can be considered precise because it is less than 1 meter and has a high range up to 8 meters. Studies conducted prove that all VLC projects have an accuracy below 20cm. Some commercial companies in light industry such AS Philips proposed that the accuracy of the system can be easily increased by adding more bulbs. The disadvantage of the system is the back channel and tracking is only possible with special hardware/app.

2. Ultrasound

Sound signal can be divided into both audible and ultrasonic. Location based systems that use ultrasonic sound need to have frequency of higher than the audible range to track and decide the location by determining the time taken for the ultrasound signal to travel from transmitter to receiver. Ultrasonic mainly uses triangulation algorithm to take reflective distance and to determine the position of the objects in indoor environments.

The advantage of this technique is low in cost ability of reflection in indoor obstacles. The disadvantage is multipath signals may disturb the measurement distance between the receivers and the emitters.

A remarkable ultrasound positioning system can be identified as Bat System. Research conducted in uses array of microphones and tags carried by the user to provide an active configuration on positioning. Using the information provided by the signal strength, it is possible to find out the position where the user is facing as well , The advantage of this system is it is easier to place more transmitter simultaneously and the precision is just a few centimeters. The drawback was the location information might be leaked and resulted in security breach.

3. Wi-Fi

One of the most known wireless technologies is Wi-Fi technology. Wireless Local Area Network (WLAN) can also be utilized for location estimation within the network. WLAN is also known as middle distance wireless technology. Past few years, it has been a very famous technology among public.

Its dominant local wireless networking standard is IEEE 802.11. Since WLAN infrastructure is quite famous among in most indoor environments, this approach might be suitable for indoor positioning system.

An important advantage of this technique is it is less cost, since most all indoor infrastructure has installed Wi-Fi enabled device. Apart from that WLAN also does not require Line of sight. There are three methods that can be implemented to apply WLAN technology for indoor positioning.

Firstly, distance calculation of a known base from a known antenna by using propagation model.

Secondly, the triangulation method where target position is calculated from signal strength information received from multiple locations.

Thirdly, fingerprinting method where mapping process takes place between signal strength of routers and the location information built in the offline database.

Accuracy of WLAN technology can be varied from 20m to 40m, but improvements can take place by deploying wireless routers or by hybrid together with another technology.

Power consumption is known as a major drawback of this technology. Small and battery power becomes the major constraint for the indoor localization. Signal attenuation from the indoor environment for example cupboards, cushion, and floors is the main limitation for this technology.

4. Bluetooth

Bluetooth technology classified into short distance wireless technology. It is also a personal area network standard. It is like Wi-Fi technology where it uses 2.4GHz and 5 GHz.

Bluetooth also can be explained as a communication technology that has been digitally embedded with information on radio frequency signal. Bluetooth was mainly introduced to exchange information between mobile devices by minimizing the usage of cables mainly to implement wireless technology within short distance.

Bluetooth is also known as lighter standard, supports many other networking services and it is highly ubiquitous. Bluetooth is a small size transceiver with unique ID in it. This technology is perfect to implement in small size infrastructure such as rooms and not suitable for a larger environment such a big building with multiple floors.

The advantage of Bluetooth technology is its device is small and encourages easy integration among mobile devices. The coverage of Bluetooth technology is 1-30 meters. Bluetooth technology is also low in cost and requires low power technology.

On the other hand, Bluetooth technology also has a few drawbacks. Each time Bluetooth performs finding, it repeats it finding steps and it resulted in latency and power consumption at the same time. It also provides low accuracy within 2m to 3m. This accuracy is considered very low compared to other technologies. Hence, this Bluetooth technology is not suitable for large localization infrastructure.

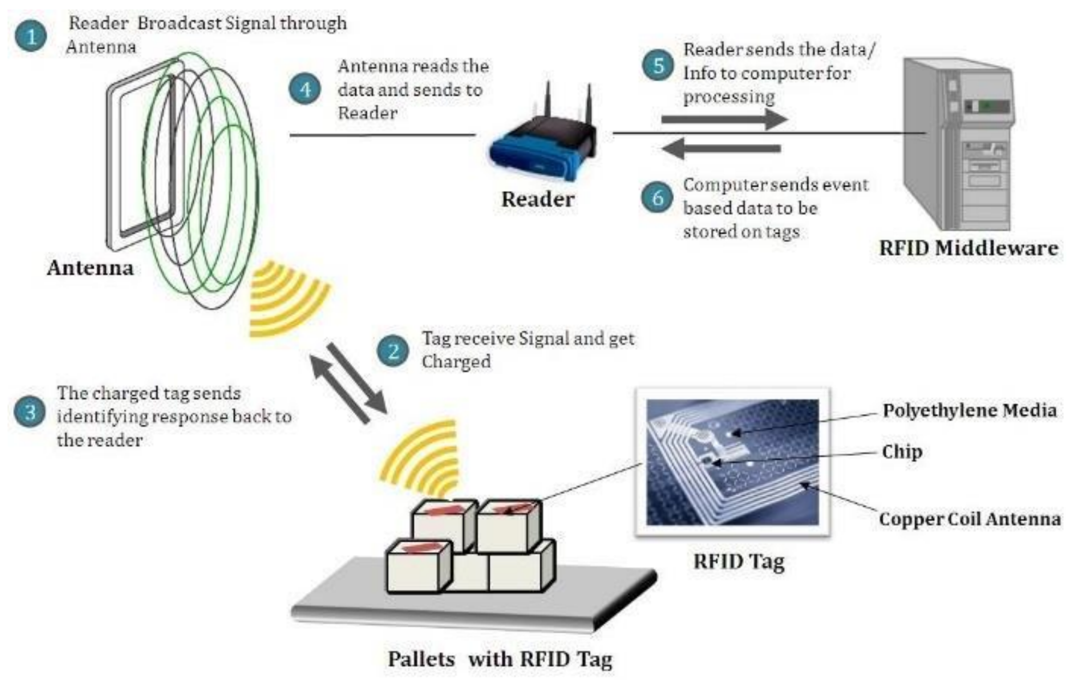
5. Radio frequency identification (RFID)

Radio Frequency Identification (RFID) holds a history of more than 40 years in history. This technology needs to be divided into two parts, which is reader localization and tags localization. RFID defines electromagnetic transmission to an RF integrated circuit by retrieving and storing data. The RFID reader reads the data obtained from the RFID tags.

RFID can be classified as either passive or active. Passive RFID tags do not require battery supply. It is small in volume and cost effective compared to the other tag. Passive RFID tags have a very limited range. The reading normally will be between 1-2m.

Active RFID tags are tags that actively send their ID. It is a small size transceiver. Advantages of active RFID tags are they have low range, and it is small thus it is very convenient. It is suitable for high unite value products in an assembly process. The radio waves signals have the ability of penetrating into solid objects. This makes the RFID technology work without the presence of Line of sight. The drawback is its accuracy is often limited the strength of signal depends on the density of the object. Frequency range in RFID technology can be divided into three parts such as Ultra-High Frequency (UHF) 860-960MHz, High Frequency (HF) at 13.56 MHz and Low Frequency (LF) at 125- 134MHz.

Furthermore, RFID technology is less cost effective, compact, with a high data rate and secured overall. The limitation of this technology is LF and HF have ability for short reading range and UHF have drawbacks in absorption of RF signal in Liquid state or presence. Figure 3 shows a scheme of working RFID technology.


1 Reader Broadcast Signal through Antenna
4 Antenna reads the data and sends to Reader
5 Reader sends the data/ Info to computer for processing
6 Computer sends event based data to be stored on tags
Reader
Antenna
RFID Middleware
2 Tag receive Signal and get Charged
3 The charged tag sends identifying response back to the reader
Polyethylene Media
Chip
Copper Coil Antenna
RFID Tag
Pallets with RFID Tag

6. Ultra-wide band (UWB)

UWB technology is the latest technology compared to other traditional technologies. UWB technology is usually implemented for its precise indoor positioning. This technology uses a nanosecond radio pulse to transmit data in a wide bandwidth range from greater than 500MHz.

Time of Arrival (TOA) and Time Difference of Arrival (TDOA) measures can be used together with this UWB technology to determine the distance between the target and reference point.

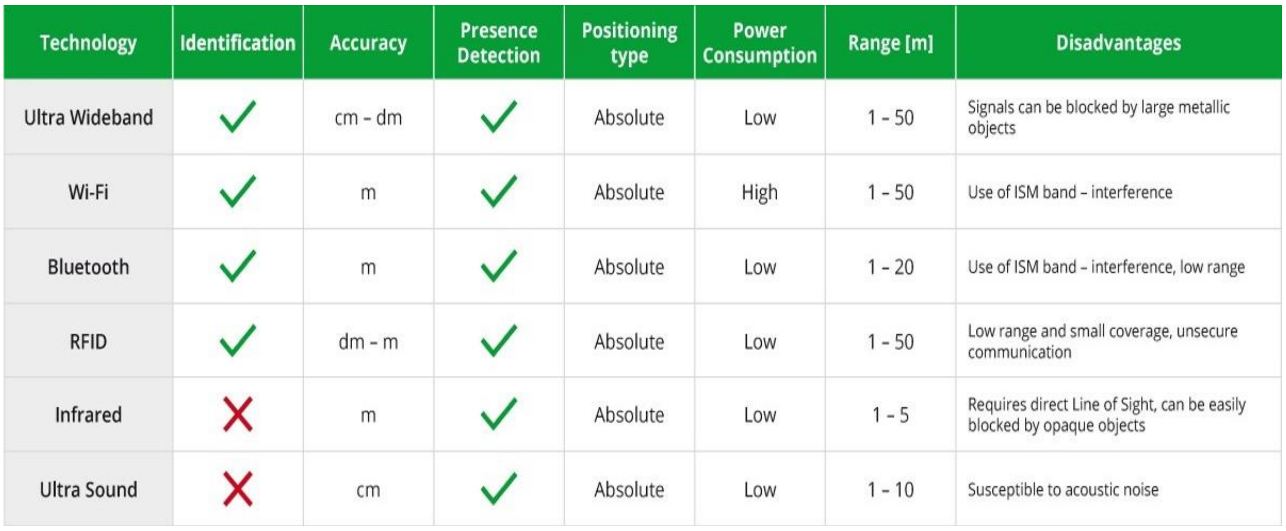
Studies conducted presented that Indoor Positioning System with UWB technology contains four transmitters and few mobile users.

The transmitters were synchronized using a time division scheme. The system proves an accuracy of 1meter when the UWB signal is at 528MHz. There are few clear plus points of implementing this UWB technology such as it has penetrating powers and it consumes very low power consumptions.

For instance, this technology uses small transmission power -41.4dBm/MHz which is limited by FCC, meaning the power consumption is low. Furthermore, it is immune to multipath effects and highly secured with less complex infrastructure. Due to this advantage of UWB technology, it is best to use in both stationery and moving indoor environment and it provides very accurate positioning accuracy.

2.2.2 Technologies comparison

A reliable Indoor Positioning System that benefits in every aspect can only be achieved by selecting the most suitable technologies to accomplish the whole system overall.



Technology	Identification	Accuracy	Presence Detection	Positioning type	Power Consumption	Range [m]	Disadvantages
Ultra Wideband	✓	cm – dm	✓	Absolute	Low	1 – 50	Signals can be blocked by large metallic objects
Wi-Fi	✓	m	✓	Absolute	High	1 – 50	Use of ISM band – interference
Bluetooth	✓	m	✓	Absolute	Low	1 – 20	Use of ISM band – interference, low range
RFID	✓	dm – m	✓	Absolute	Low	1 – 50	Low range and small coverage, unsecure communication
Infrared	✗	m	✓	Absolute	Low	1 – 5	Requires direct Line of Sight, can be easily blocked by opaque objects
Ultra Sound	✗	cm	✓	Absolute	Low	1 – 10	Susceptible to acoustic noise

Comparison among existing technologies for selecting suitable technologies for indoor positioning system is very crucial. For a detailed technology comparison, a few parameters were chosen such as accuracy, coverage cost, complexity, typical environment, and power consumption were explained in this table.

2.2.3 Bluetooth Technology

What is Bluetooth?

Bluetooth is a wireless communication technology that enables devices to transfer data and communicate with each other over short distances. It is designed to be a low-power and low-cost alternative to other wireless communication technologies.

History of Bluetooth:

Bluetooth was developed in the 1990s by Ericsson, a Swedish telecommunications company. It was originally intended to be a wireless alternative to RS-232 cables, which were commonly used for serial communication between devices at the time. The name "Bluetooth" comes from Harald Bluetooth, a 10th-century Danish king who united Denmark and Norway.

Frequency range of Bluetooth:

Bluetooth uses radio waves in the 2.4 GHz frequency range to transmit data between devices. This frequency range is divided into 79 channels, each with a bandwidth of 1 MHz.

Features of Bluetooth:

- Low power consumption
- Low cost
- Short-range communication (typically up to 10 meters)
- Simple pairing process
- Secure communication using encryption

Advantages of Bluetooth:

- Low power consumption for longer battery life in portable devices

- Low cost compared to other wireless communication technologies

- Short-range communication for improved security and privacy

- Simple pairing process for easy device setup

- Secure communication using encryption for improved security

Disadvantages of Bluetooth:

- Limited range compared to other wireless communication technologies

- Limited data transfer rate compared to other wireless communication technologies

- Interference from other devices operating in the same frequency range

Uses of Bluetooth:

- Wireless audio devices (e.g. headphones, speakers)

- Wireless input devices (e.g. keyboards, mice)

- Mobile devices (e.g. smartphones, tablets)

- Internet of Things (IoT) devices

- Smart home automation

Overall, Bluetooth is a widely used wireless communication technology that provides low-power and low-cost connectivity between devices over short distances. Its simple pairing process and secure communication make it ideal for a variety of applications, from wireless audio devices to smart home automation. However, its limited range and data transfer rate can be a disadvantage in certain applications.

2.3 IPS and GPS :

An indoor positioning system (IPS) can be thought of as a global positioning system (GPS) for indoor venues. While GPS is used outdoors, IPS can detect real-time locations to accurately determine the coordinates of people or assets inside a building. These coordinates are typically represented visually by a blue dot on a digital indoor map to provide additional context to users and features such as way finding.

If you're looking for a specific location, for example, you can pull up the digital mall map on your mobile device and get directions from your exact location. Similar to outdoor GPS, users are not required to enter a "start" destination and as they begin to follow along a route, the blue dot moves with them.

When trying to navigate indoors, often leveraging a GPS signal is out of the question as buildings limit access to GPS signals. With the absence of a GPS signal, devices must rely on other technologies such as indoor positioning to pinpoint the indoor location of a user. There are a variety of indoor positioning technologies that can be used, including beacons, Wi-Fi, RFID, and geomagnetic.

Whether leveraging a cost-effective infrastructure-free system, or a more accurate hardware system, these technologies can enable a user to locate themselves or enable employees to keep track of their assets within a building.

2.4 Important of Indoor Positioning System

- Locating and positioning :

While most current IPS are able to detect the location of an object, they are so coarse that they cannot be used to detect the orientation or direction of an object.

- Locating and tracking

One of the methods to thrive for sufficient operational suitability is "tracking". Whether a sequence of locations determined form a trajectory from the first to the most actual location. Statistical methods then serve for smoothing the locations determined in a track resembling the physical capabilities of the object to move. This smoothing must be applied, when a target moves and also for a resident target, to compensate erratic measures. Otherwise the single resident location or even the followed trajectory would compose of an itinerant sequence of jumps.

- Identification and segregation

In most applications the population of targets is larger than just one. Hence the IPS must serve a proper specific identification for each observed target and must be capable to segregate and separate the targets individually within the group. An IPS must be able to identify the entities being tracked, despite the "non-interesting" neighbours. Depending on the design, either a sensor network must know from which tag it has received information, or a locating device must be able to identify the targets directly.

The major consumer benefit of indoor positioning is the expansion of location-aware mobile computing indoors. As mobile devices become ubiquitous, contextual awareness for applications has become a priority for developers. Most applications currently rely on GPS, however, and function poorly indoors.

Indoor positioning systems (IPS) are technologies that enable the tracking of objects or people within a building or other enclosed space. IPS is achieved through the use of various sensors, such as Wi-Fi, Bluetooth, and GPS, which can be installed throughout the building to provide accurate location data.

There are several technologies that can be used in IPS, and each has its own advantages and limitations. Some of the most commonly used technologies include:

- **Wi-Fi-based IPS:** This technology uses Wi-Fi access points to track the location of devices within a building. The system works by measuring the strength of the Wi-Fi signal received by a device and using this information to triangulate its position.
- **Bluetooth-based IPS:** Bluetooth beacons can be used to provide indoor positioning data. These beacons transmit a unique identifier that can be picked up by a device, allowing its location to be determined.
- **Inertial-based IPS:** This technology uses sensors within a device, such as accelerometers and gyroscopes, to track its movement and determine its location.
- **Magnetic-based IPS:** This technology uses the Earth's magnetic field to determine the location of a device. By measuring the strength and direction of the magnetic field at various points within a building, the system can triangulate the position of a device.

Once the sensors have collected location data, it is processed by a software system that can provide real-time location information to
users.

Applications benefiting from indoor location include:

1. Accessibility aids for the visually impaired.
2. Augmented reality
3. School campus
4. Museum guided tours
5. Shopping malls, including hypermarkets.
6. Warehouses
7. Factory
8. Airports, bus, train and subway stations
9. Parking lots, including these in hypermarkets
10. Targeted advertising
11. Social networking service
12. Hospitals

13. Hotels
14. Sports
15. Cruise Ships
16. Indoor robotics
17. Tourism
18. Amusement Parks

2.5 Benefits of Indoor Positioning System

There are many benefits of using an indoor positioning system, including:

- **Improved safety and security:** IPS can improve safety and security within a building by enabling quick location tracking in case of an emergency. In case of a fire or other disaster, indoor positioning can be used to quickly locate people who may be in danger and guide them to safety.
- **Enhanced customer experience:** In retail environments, IPS can be used to enhance customer experience. By tracking the movements of customers within a store, retailers can gain valuable insights into their behavior and preferences, which can be used to improve product placement and marketing strategies.
- **Optimized workflow and efficiency:** In industrial and manufacturing environments, IPS can optimize workflow and efficiency. By tracking the location of equipment and personnel, businesses can identify bottlenecks and other areas for improvement, leading to increased productivity and profitability.
- **Improved patient care and safety:** In healthcare settings, IPS can improve patient care and safety. By tracking the location of medical equipment and personnel, hospitals can ensure that critical resources are always available when needed and that patients are receiving the care they require.
- **Enhanced user experience:** IPS can enhance the overall user experience in a variety of settings, from museums and theme parks to airports and public transportation systems. By providing real-time location data and other relevant information, indoor positioning can help visitors navigate complex environments more easily and efficiently.

2.6 Methods for Indoor Positioning System

There are several methods used in indoor positioning systems to calculate the location of a device or object within a building. Some of the most commonly used methods include:

1. Received signal strength (RSS) to calculate the indoor positioning

RSS or RSSI is a measure of the received radio signal power. Distance estimation using this measurement principle requires prior knowledge of the path loss models of RF transmitters obtained either empirically or statistically. Generally, propagation path loss increases with increase in distance and operating frequency. The accuracy of distance estimation using this approach depends strongly on the accuracy of the path loss model used. RSS-based ranging and positioning techniques are popular since they don't require high complexity hardware and are relatively cheaper and easier to implement25. However, RSS-based range estimation using RF signals is less accurate and less reliable in presence of obstacles. Multipath propagation and shadowing from obstacles cause challenges for RF wave propagation as RF electromagnetic waves are easily reflected and strongly attenuated by obstacles. Also, significant deviation in RSS values of RF waves can be seen due to human presence, as the direct Line-of-Sight (LOS) is blocked because of the absorption of electromagnetic waves in RF band by the human body. On the other hand, it is an extremely simple method to implement that requires few hardware resources.

The RSSI method involves converting the received signal strength (RSS) into an estimated distance using a calibration curve that maps RSS values to corresponding distances. The calibration curve can be represented mathematically as a function, such as a power law or exponential decay function. For example, a simple power law calibration curve might be represented as:

$$10*((\text{Measured Power} - \text{Instant RSSI}) / 10* N)$$

2. Time of Flight (TOF) or Time of Arrival (TAO) to calculate the indoor positioning

TOA measurement represents the exact time taken by the signal to travel from the transmitter to the receiver. In this case distance travelled by the signal can be calculated from TOA measurement and the propagation speed of signal (usually the speed of light). Most notable example of range estimation using TOA measurements is the GPS. In general, direct TOA based positioning results in two problems. First, all transmitters and receivers in the system must be precisely synchronized. Second, a timestamp must be labeled in the transmitting signal for the measuring unit to discern the distance the signal has traveled. To have a good precision you need wide bandwidth signals such spread-spectrum signals or very short pulses. Also, noise, reflections, multipath, and scattering of the signal cause errors in ranging. In multipath situations, it is difficult to identify the first path of the signal to obtain accurate TOA measurement. TOA measurements with Two-Way Ranging (TWR) does not require synchronization between transmitters and receivers as it considers bidirectional communication path of the signal, and the Round-Trip Time (RTT) can be calculated using offset time between transmitter and receiver.

3. Time difference of arrival (TDOA)

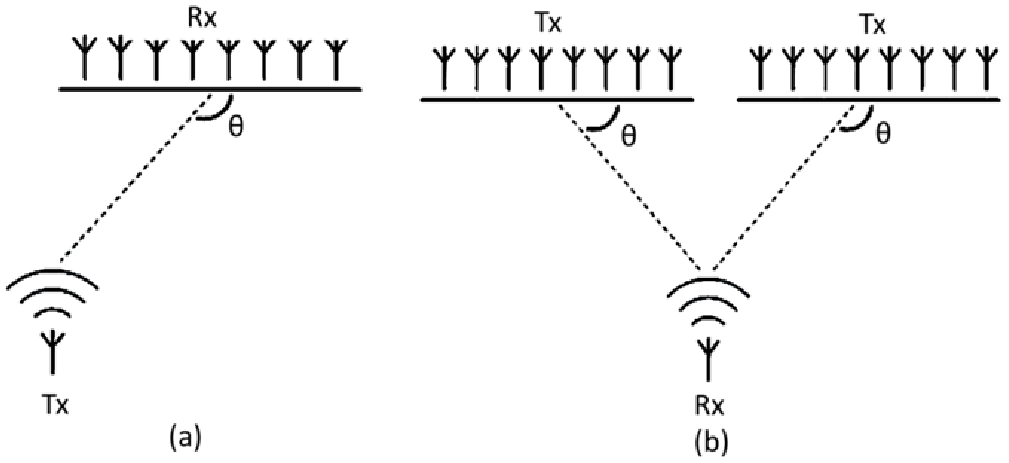
TDOA is a technique to determine relative position of the target transmitter based on the difference in time at which the signal arrives at multiple reference receivers instead of considering absolute arrival time used in TOA technique. The position can be obtained as the intersection of several hyperbolas corresponding to each reference receiver. This technique requires highly precise synchronization between the reference receivers. Precision of TDOA based distance estimation depends on criteria such as the characteristics of the signal (signal modulation and bandwidth), SNR, environment (multipath propagation and interference).

4. The phase difference of arrival (PDOA)

PDOA method translates difference in phase of arrival measurements of the multifrequency carrier signal into the distance between the transmitter and the receiver. This method is based on dual-frequency radar technique for range estimation. In PDOA technique, two continuous wave signals of different frequencies with frequency difference Δf between them are transmitted. At the receiver, the phase difference $\Delta\varphi$ is measured. The distance between the transmitter and the receiver is proportional to Δφ and inversely proportional to Δf. Because of the presence of extremely small signal bandwidth phase estimation error can be very small. This method is limited because of the requirement of the complexity of the hardware for phase of arrival measurements and the need of multifrequency carrier signal.

5. Angle of Arrival (AOA) and Angle of Departure (AOD) measurement to calculate the indoor positioning with high accuracy

AOA and AOD measurement principles rely of angle measurements to estimate the direction of RF transmitter and receiver, respectively. Phased array of antennas is deployed at the receiver and the transmitter respectively for direction finding using AOA and AOD. The accuracy of angle measurements is limited by multipath reflections, shadowing and directivity of the antennas. To achieve good accuracy in direction finding, accurate orientation information of stationary locators is required in addition to complex hardware providing accurate angle measurements. The Bluetooth core specification 5.1 provided by Bluetooth Special Interest Group (SIG) include a direction-finding feature that makes it possible to detect the direction of a Bluetooth signal. With Bluetooth direction finding feature, either AOA or AOD method can be used for direction estimation with the help of phased array antennas.



Schematic diagram of the calculation method by Angle of Arrival measurement

6. Channel Status Information (CSI)

CSI was introduced as an index to improve upon the limitations of the RSS. RSS provides only the information of strength of the signal, while the CSI provides more fine-grained physical layer information. In addition to signal strength information, phase shift information of the individual signal components is provided by CSI. Thus, main path of received signal can be easily distinguished from the reflections of the signal. In the case of fingerprinting, CSI provides more information compared to RSS for fingerprinting thereby improving positioning accuracy. CSI associated with different deep learning approaches allow to obtain more precise positioning results.

2.7 The different indoor location algorithms

1. Proximity detection

Proximity based positioning methods estimate the position of the target by determining whether the target object is in the proximity to the reference point or not. This is achieved with the help of sensing mechanism, placed either at the reference point or at the target, specifically aimed at detecting the presence. Proximity detection is the simplest positioning method used for applications that do not require very high accuracy.

2. Centroid Determination

Position estimation using centroid determination method involves the calculation of the geometric center using the positions of multiple reference nodes within the detection range of the target node. This low complexity positioning method is easy to implement but it offers low accuracy as it only takes the proximity data into account and simply performs averaging of the coordinates to obtain the position of the target. Weighted Centroid Localization (WCL) is an improvement to the centroid determination method in which centroid calculation is influenced by weights that can be functions of signal strength or distances of each node.

3. trilateration

trilateration is a position determination method using simultaneous distance measurements of the target from three or more reference nodes with known positions. trilateration based positioning method can utilize distance measurements from various ranging methods. After the distance measurements are available, the position of the target can be determined as the intersection of the circles or spheres formed with radii representing the respective distance measurement for each reference node and the geometric centers of the circles or spheres coinciding with the positions of reference nodes (Fig. 1). When the distance estimation is accurate, trilateration offers very high positioning accuracy.

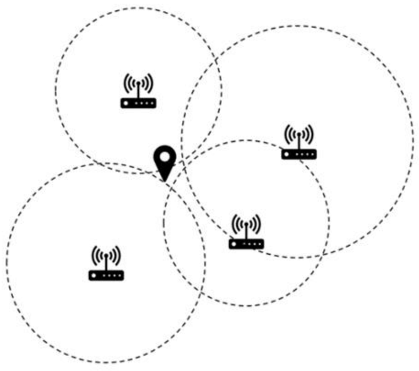


Figure 1 : trilateration method

4. Angulation

Angulation is similar positioning method to trilateration, except it determines the position of the target by using the angle measurements relative to multiple reference points (Fig. 2) instead of distance measurements. Phased array antennas with direction detection capabilities are popular for angle measurements in the case of wireless signals. The position of the target is estimated at the intersection point of the lines formed using the angle measurements and passing through respective reference points (Fig. 2).

Error in each angle measurement determines the error in positioning using this method as the lines formed using the angle measurements will not intersect at a single point (Fig. 2).

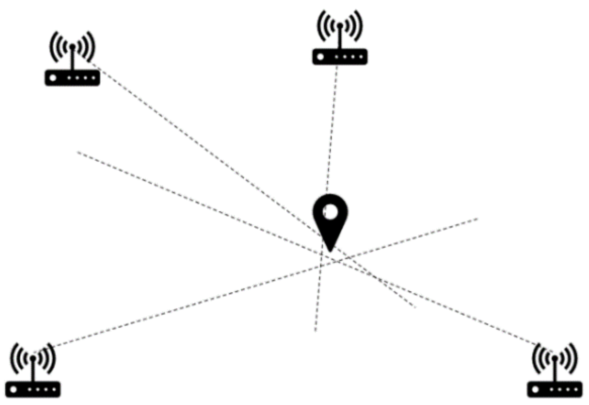


Figure 2: Tri-Angulation Method

5. Fingerprinting

Fingerprinting is a scene analysis-based positioning method which relies on the fingerprints obtained by the environmental survey. Fingerprinting is most often performed using Received Signal Strength Indicator (RSSI) of Radio Frequency (RF) signals5, although audio signals6, magnetic fields, visual information from images7 and combination of multiple observations from surroundings8 are also the quantities used for fingerprints. This method generally has two phases. First, is the calibration phase which is performed offline, meaning before deployment of the system. In this phase, a map is created using the measurements of the fingerprints over the operational area of the system. The second phase is online positioning phase. During this phase real time measurement of the fingerprint is compared with the offline measurements to predict the position of target. Buildings with an already existing infrastructure for technologies such as Wi-Fi and Bluetooth are examples where the fingerprint is interesting as a location method with low cost and low accuracy.

6. Dead Reckoning

Dead reckoning is the process of estimating position of the target with the help of previously known or estimated positions and known or estimated speeds and sometimes accelerations over the elapsed time. Inertial Measurement Units (IMU), with motion sensors and rotation sensors are generally used for dead reckoning. IMUs can determine the position and speed of the target without external inputs. Inertial sensors can be designed in a Microelectromechanical Sensors (MEMS) technology and thus can be embedded in lightweight devices such as smartphones. The inaccuracy of the dead reckoning process is cumulative, thus the estimated position drifts from the actual position with increasing deviation as the time increases. The term Pedestrian Dead Reckoning (PDR) is often used in the field of indoor applications to indicate that the accelerometers have been attached to the body of the user.

7. Map Matching

In the absence of positioning update due to reasons such as bad signal reception, the position estimated by an indoor navigation system, can be controlled, corrected, or updated using a map database. To find a correct representation of target's position, certain elements of the map are associated with the characteristics of the target's trajectory, which is called as Map Matching (MM). MM algorithms associate current positioning data with spatial map data to establish an accurate link between them, thereby improving positioning accuracy.

8. Combination of Positioning Methods

In the case of some positioning system architectures, more than one measuring principles or positioning techniques are used. For example, angulation can be used in combination with lateration to reduce the angle measurements required for estimating the position of the target or to improve the positioning accuracy. In addition to angle and distance measurements, other information such as proximity or various kinds of sensor data can be used to enhances the performance of a positioning system. Recently, positioning methods utilizing heterogeneous information1 and data fusion techniques21 are focus of research in the field of indoor positioning.

2.8 Firebase

Firebase is a mobile and web application development platform that provides a range of services to help developers build high-quality applications quickly and easily. It was acquired by

Google in 2014 and has since become one of the most popular backend-as-a-service (BaaS) platforms available.

Features of Firebase

Firebase offers a range of features that make it an attractive option for developers, including:

1. Real-time database: Firebase's real-time database allows developers to store and sync data in real-time across multiple clients. This makes it easy to build collaborative applications, such as chat apps and collaborative document editors.

2. Authentication: Firebase provides a secure authentication system that allows developers to easily add user authentication to their applications.

3. Hosting: Firebase's hosting service makes it easy to deploy web applications quickly and easily.

4. Cloud Functions: Firebase's cloud functions allow developers to write serverless functions that can be triggered by events in their applications.

5. Analytics: Firebase provides a range of analytics tools that allow developers to track user behavior and measure the performance of their applications.

6. Crash reporting: Firebase's crash reporting service allows developers to track and analyze application crashes in real-time.

7. Cloud messaging: Firebase's cloud messaging service allows developers to send push notifications to users across multiple platforms.

Advantages of Firebase

There are several advantages to using Firebase for application development, including:

1. Rapid development: Firebase's pre-built features and tools make it easy for developers to build high-quality applications quickly and easily.

2. Scalability: Firebase's cloud-based infrastructure allows applications to scale easily as user demand increases.

3. Real-time data synchronization: Firebase's real-time database allows developers to build real-time collaborative applications without having to manage complex synchronization logic.

4. Cross-platform support: Firebase supports a range of platforms, including iOS, Android, and web, making it easy for developers to build cross-platform applications.

5. Ease of use: Firebase's intuitive interface and documentation make it easy for developers of all skill levels to use.

Disadvantages of Firebase

There are also some disadvantages to using Firebase, including:

1. Vendor lock-in: Since Firebase is a proprietary platform, developers may be locked into using it if they build their application using its services.

2. Limited customization: While Firebase provides a range of pre-built features and tools, some developers may find that they need more customization options than what is available.

3. Limited control over infrastructure: Since Firebase is a cloud-based platform, developers have limited control over the underlying infrastructure.

How Firebase Works

Firebase works by providing a range of pre-built features and tools that developers can use to build high-quality applications quickly and easily. Developers can access these features through Firebase's web console or through its REST API.

When a developer creates an application using Firebase, they are given access to a range of services, including real-time database, authentication, hosting, cloud functions, analytics, crash reporting, and cloud messaging. These services can be used individually or in combination to build complex applications.

Firebase's real-time database allows developers to store and sync data in real-time across multiple clients. This makes it easy to build collaborative applications, such as chat apps and collaborative document editors. Firebase also provides a secure authentication system that allows developers to easily add user authentication to their applications.

Overall, Firebase is a powerful platform that provides a range of features and tools for building high-quality mobile and web applications quickly and easily. While there are some disadvantages to using Firebase, the advantages make it an attractive option for many developers.

2.9 SQL Server Database

A SQL database is a type of relational database that uses Structured Query Language (SQL) to manage and manipulate data. SQL is a programming language that allows users to interact with databases by writing queries that retrieve and manipulate data.

Advantages of SQL Databases

There are several advantages to using SQL databases, including:

- Ease of use: SQL is a widely used language, and there are many tools and resources available to help developers write and manage SQL databases.
- Scalability: SQL databases can scale easily as the amount of data stored in them increases.
- Data integrity: SQL databases enforce data integrity by ensuring that data is consistent and accurate.
- Security: SQL databases offer robust security features, such as access control and encryption, to protect data from unauthorized access.
- Flexibility: SQL databases can be used in a wide range of applications, from small personal projects to large enterprise systems.

Disadvantages of SQL Databases

There are also some disadvantages to using SQL databases, including:

- Complexity: SQL databases can be complex to set up and manage, particularly for large-scale applications.

- Cost: While open-source SQL databases are available, commercial SQL databases can be expensive to license and maintain.
- Performance: SQL databases can experience performance issues when dealing with very large datasets or complex queries.
- Limited scalability: While SQL databases can scale easily, there are limits to how much data they can store and how many concurrent users they can support.
- Relational model limitations: The relational model used by SQL databases can be limiting in some cases, particularly when dealing with unstructured or semi-structured data.

Overall, SQL databases are a powerful tool for managing and manipulating data. While they have some disadvantages, their ease of use, scalability, and robust security features make them a popular choice for a wide range of applications.

2.10 Kotlin

Kotlin is a statically-typed programming language that runs on the Java Virtual Machine (JVM). It was developed by JetBrains and was designed to be a more concise and expressive alternative to Java. Kotlin is interoperable with Java, which means that developers can use Kotlin and Java code together in the same project.

Features of Kotlin :

- Concise syntax: Kotlin has a more concise syntax than Java, which means that developers can write less code to achieve the same functionality.
- Null safety: Kotlin has built-in null safety features that help prevent null pointer exceptions in code.
- Interoperability with Java: Kotlin is interoperable with Java, which means that developers can use both languages together in the same project.
- Functional programming support: Kotlin has built-in support for functional programming concepts, such as lambdas and higher-order functions.
- Extension functions: Kotlin allows developers to add new functions to existing classes without having to inherit from them or use design patterns like decorators.

- Data classes: Kotlin has built-in data classes that make it easy to create classes that are used primarily to store data.
- Operator overloading: Kotlin allows developers to overload operators, which makes it easier to work with complex data types.
- Coroutines: Kotlin has built-in support for coroutines, which are a way to write asynchronous code in a more synchronous style.
- Smart casts: Kotlin has smart casts that allow developers to automatically cast variables to a more specific type based on runtime checks.
- Companion objects: Kotlin allows developers to define companion objects, which are objects that are tied to a class and can be used to store static methods and properties.

Advantages of Kotlin

There are several advantages to using Kotlin, including:

- Conciseness: Kotlin is a more concise language than Java, which means that developers can write less code to achieve the same functionality.
- Interoperability: Kotlin is interoperable with Java, which means that developers can use both languages together in the same project.
- Null safety: Kotlin has built-in null safety features, which can help prevent null pointer exceptions in code.
- Functional programming support: Kotlin has built-in support for functional programming concepts, such as lambdas and higher-order functions.
- Ease of learning: Kotlin has a relatively simple syntax and is easy to learn for developers who are familiar with Java.

Disadvantages of Kotlin

There are also some disadvantages to using Kotlin, including:

- Learning curve: While Kotlin is relatively easy to learn for developers who are familiar with Java, it still has a learning curve for developers who are new to the language.

- Limited tooling support: While Kotlin has good support in many IDEs, such as IntelliJ IDEA and Android Studio, some tools may not have full support for the language.
- Performance overhead: While Kotlin code can be compiled to run on the JVM, there is some performance overhead associated with using a higher-level language like Kotlin.
- Less community support: While Kotlin has a growing community of developers, it is still less popular than Java and may have less community support in some areas.

Overall, Kotlin is a powerful and expressive programming language that offers many advantages over Java. Its interoperability with Java, concise syntax, and built-in null safety features make it an attractive choice for many developers.

2.11 Flutter

Flutter is a mobile app SDK (Software Development Kit) developed by Google for building high-performance, high-fidelity, apps for iOS and Android, from a single codebase. It uses the Dart programming language, which is a modern, object-oriented language that compiles to native code for both iOS and Android. Flutter provides a rich set of customizable widgets, which makes it easy to build beautiful, responsive user interfaces that look and feel like native apps. With hot reload, developers can make changes to the code and see them reflected in the app's UI in real-time, which can speed up the development process. Flutter has a growing community of developers and provides extensive documentation and support.

Features:

- Flutter is a mobile app SDK for building high-performance, high-fidelity, apps for iOS and Android, from a single codebase.
- It uses the Dart programming language, which is a modern, object-oriented language that compiles to native code for both iOS and Android.
- Flutter provides a rich set of customizable widgets, which makes it easy to build beautiful, responsive user interfaces.

- It includes hot reload, which allows developers to quickly see changes they make to the code reflected in the app's UI in real-time.
- Flutter has a growing community of developers and provides extensive documentation and support.

Advantages:

- Single codebase: Flutter allows developers to build apps for both iOS and Android from a single codebase, which can save time and resources.
- High performance: Flutter compiles to native code for both iOS and Android, which means that apps built with Flutter can be faster and more responsive than those built with other cross-platform frameworks.
- Beautiful UI: Flutter provides a rich set of customizable widgets, which makes it easy to build beautiful, responsive user interfaces that look and feel like native apps.
- Hot reload: With hot reload, developers can make changes to the code and see them reflected in the app's UI in real-time, which can speed up the development process.
- Growing community: Flutter has a growing community of developers and provides extensive documentation and support.

Disadvantages:

- Limited libraries: While Flutter has a growing community and a rich set of customizable widgets, its library of third-party packages is still relatively limited compared to other programming languages.
- Limited platform support: While Flutter can be used to build apps for both iOS and Android, it currently does not support other platforms such as Windows or macOS.
- Learning curve: Dart is not as widely used as other programming languages such as Java or Swift, which means that developers may need to invest time in learning the language before they can start building apps with Flutter.

- Smaller talent pool: Because Flutter is a relatively new technology, there may be fewer developers who are experienced with the platform, which could make it harder to find talent for development projects.

2.12 Hardware Can Used in System

Indoor positioning systems are becoming increasingly popular for a variety of applications, from tracking the location of people and assets to providing indoor navigation. These systems rely on a network of sensors placed throughout a building to track the movement of objects or people. Microcontrollers like Arduino can be used to process the data received from these sensors and provide real-time location information , so we will explain the microcontroller :

What is a Microcontroller?

A microcontroller is a small computer on a single integrated circuit. It contains a processor core, memory, and input/output peripherals. It is designed to control small devices such as sensors, motors, and lights.

2.12.1 Arduino

What is Arduino?

Arduino is an open-source platform used for building electronics projects. It consists of a microcontroller board, software, and community support. The Arduino board is designed to make it easy to create interactive projects by providing a simple interface between the hardware and software , like (Arduino Nano) .

Features of Arduino:

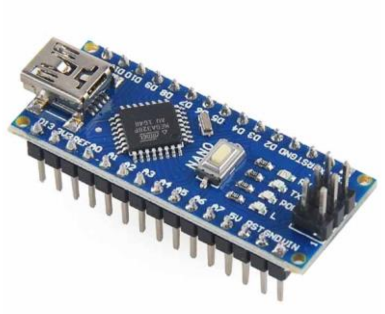
- Open-source platform

- Easy-to-use software

- Wide range of compatible sensors and modules

- Large community support

- Low cost


Arduino Nano

Advantages of Arduino:

- Easy to learn and use

- Low cost compared to other microcontrollers

- Wide range of compatible sensors and modules

- Large community support for troubleshooting and sharing projects

- High-level programming language similar to C++

Disadvantages of Arduino:

- Limited processing power compared to other microcontrollers

- Limited memory for storing programs and data

- Limited number of input/output pins

- Not suitable for high-performance applications

Uses of Arduino:

- Robotics

- Home automation

- Internet of Things (IoT) projects

- Wearable technology

- Musical instruments and sound equipment

I hope this information is helpful! Let me know if you have any further questions.

Sure, I'd be happy to provide you with some information about ESP32.

2.12.2 ESP32

What is ESP32?

ESP32 is a low-cost, low-power system on a chip (SoC) microcontroller with integrated Wi-Fi and Bluetooth capabilities. It is designed for a wide range of applications, from Internet of Things (IoT) devices to wearables and smart home automation.

Features of ESP32:

- Dual-core Tensilica LX6 processor

- Integrated 2.4 GHz Wi-Fi and Bluetooth capabilities

- Up to 240 MHz clock frequency

- Ultra-low power consumption

- Wide range of input/output peripherals, including analog-to-digital converters (ADCs), pulse width modulators (PWMs), and touch sensors

- Secure boot and flash encryption

Advantages of ESP32:

- Low cost compared to other microcontrollers with similar capabilities

- Integrated Wi-Fi and Bluetooth capabilities for easy connectivity

- Dual-core processor for improved performance and multitasking

- Ultra-low power consumption for longer battery life in portable devices

- Wide range of input/output peripherals for flexibility in designing applications

- Secure boot and flash encryption for improved security

Disadvantages of ESP32:

- Limited memory for storing programs and data compared to other microcontrollers

- Limited processing power compared to high-end microcontrollers

- Requires some programming knowledge to use effectively

Uses of ESP32:

- Internet of Things (IoT) devices

- Wearable technology

- Smart home automation

- Industrial automation and control systems

- Robotics

Overall, ESP32 is a versatile and cost-effective microcontroller with integrated Wi-Fi and Bluetooth capabilities that make it ideal for a wide range of applications. Its dual-core processor and low power consumption make it suitable for both high-performance and portable devices, while its wide range of input/output peripherals provide flexibility in designing custom applications.

2.12.3 Bluetooth Module

What is a Bluetooth module?

A Bluetooth module is a small electronic component that enables wireless communication between devices using Bluetooth technology. It is designed to be easy to integrate into electronic devices and provides a simple interface for wireless communication.

What is HM10?

HM10 is a Bluetooth module produced by Jinan Huamao Technology Co., Ltd. It is a low-cost, low-power module that provides wireless communication between devices using Bluetooth technology.

Features of HM10:

- Bluetooth 4.0 Low Energy (BLE) technology

- Low power consumption

- Small form factor for easy integration into electronic devices

- Simple interface for wireless communication

- Wide range of operating voltage (2.0V to 3.6V)

Advantages of HM10:

- Low cost compared to other Bluetooth modules with similar capabilities

- Low power consumption for longer battery life in portable devices

- Small form factor for easy integration into electronic devices

- Simple interface for wireless communication

- Wide range of operating voltage for flexibility in designing applications

Disadvantages of HM10:

- Limited range compared to other wireless communication technologies

- Limited data transfer rate compared to other wireless communication technologies

- Requires some programming knowledge to use effectively

Uses of HM10:

- Internet of Things (IoT) devices

- Wearable technology

- Smart home automation

- Industrial automation and control systems

- Robotics

Overall, HM10 is a low-cost and low-power Bluetooth module that provides wireless communication between devices using Bluetooth technology. Its small form factor and wide range of operating voltage make it easy to integrate into electronic devices, while its low power consumption makes it ideal for portable devices. Its simple interface and low cost make it a popular choice for hobbyists and makers looking to add wireless communication to their projects.

Chapter 3 : Requirement and Analysis

3.1 Requirement

Functional Requirements

1. Positioning requirements:

- The system should be able to detect the position of a Bluetooth-enabled device within an indoor environment.
- The system should be able to track the movement of a Bluetooth-enabled device in real-time.
- The system should be able to provide accurate location information with a minimum error rate of 1 meter.
- The system should be able to display the location of the device on a map or floor plan of the indoor environment.

2. Navigation requirements:

- The system should be able to provide directions or navigation assistance to the user based on their current location and destination.
- The system should be able to calculate the shortest or fastest route to the destination.
- The system should be able to provide information about nearby points of interest or amenities.

3. User requirements:

- The system should be able to identify and authenticate users based on their Bluetooth-enabled device.
- The system should be able to provide personalized recommendations or information based on user preferences or history.
- The system should be able to support multiple users simultaneously.

4. Compatibility requirements:

- The system should be compatible with a wide range of Bluetooth-enabled devices, including smartphones, tablets, and wearables.
- The system should be compatible with a variety of Bluetooth standards, including Bluetooth Low Energy (BLE) and Bluetooth Classic.

5. Integration requirements:

- The system should be able to integrate with other location-based services, such as mapping or geofencing services.
- The system should be able to integrate with third-party applications and services, such as social media or e-commerce platforms.

6. Data requirements:

- The system should be able to collect and store data about user location and movement for analysis and optimization.
- The system should be able to protect user privacy by only collecting and storing necessary data.

Non Functional Requirements :

1. Performance requirements:

- The system should be able to handle a large number of devices simultaneously without significantly impacting performance.
- The system should be able to provide location information in real-time with minimal latency.
- The system should be able to provide accurate location information with a minimum error rate of 1 meter.

2. Availability:

- The system should be reliable and have a high level of availability, with minimal downtime for maintenance or upgrades.

3. Extensibility:

- The system should be able to integrate with other systems such as building management or security systems.
- The system should be designed to support future updates and enhancements.

4. Portability:

- The system should be designed to work on a variety of hardware platforms and operating systems.
- The system should be able to adapt to changes in the underlying technology stack.

5. Reliability:

- The system should be secure and protect user privacy by only collecting and storing necessary data.
- The system should be designed to handle errors and exceptions gracefully.

6. Usability:

- The system should be easy to use and require minimal training for end-users.
- The system should provide clear and concise instructions for users.

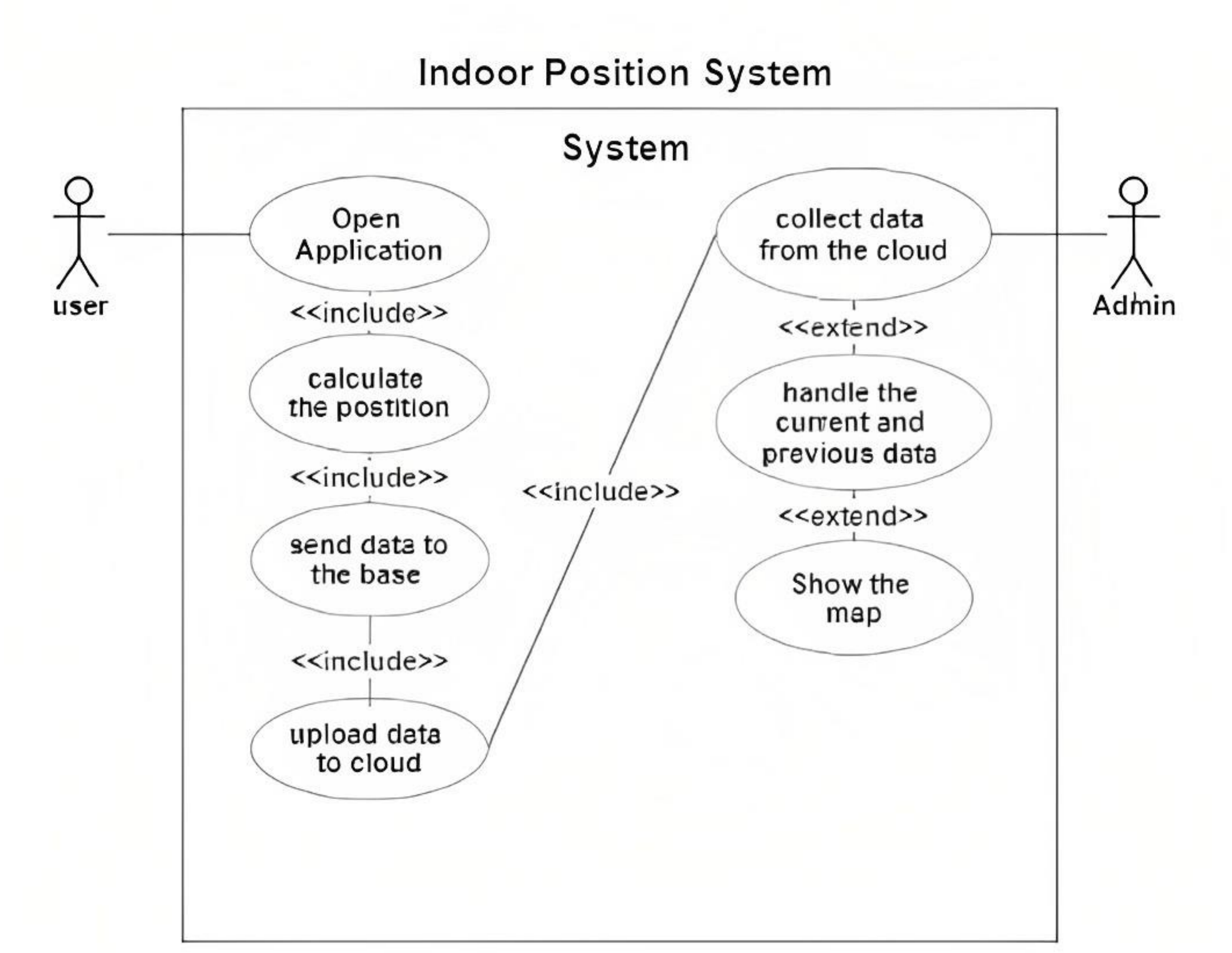
7. Correctness:

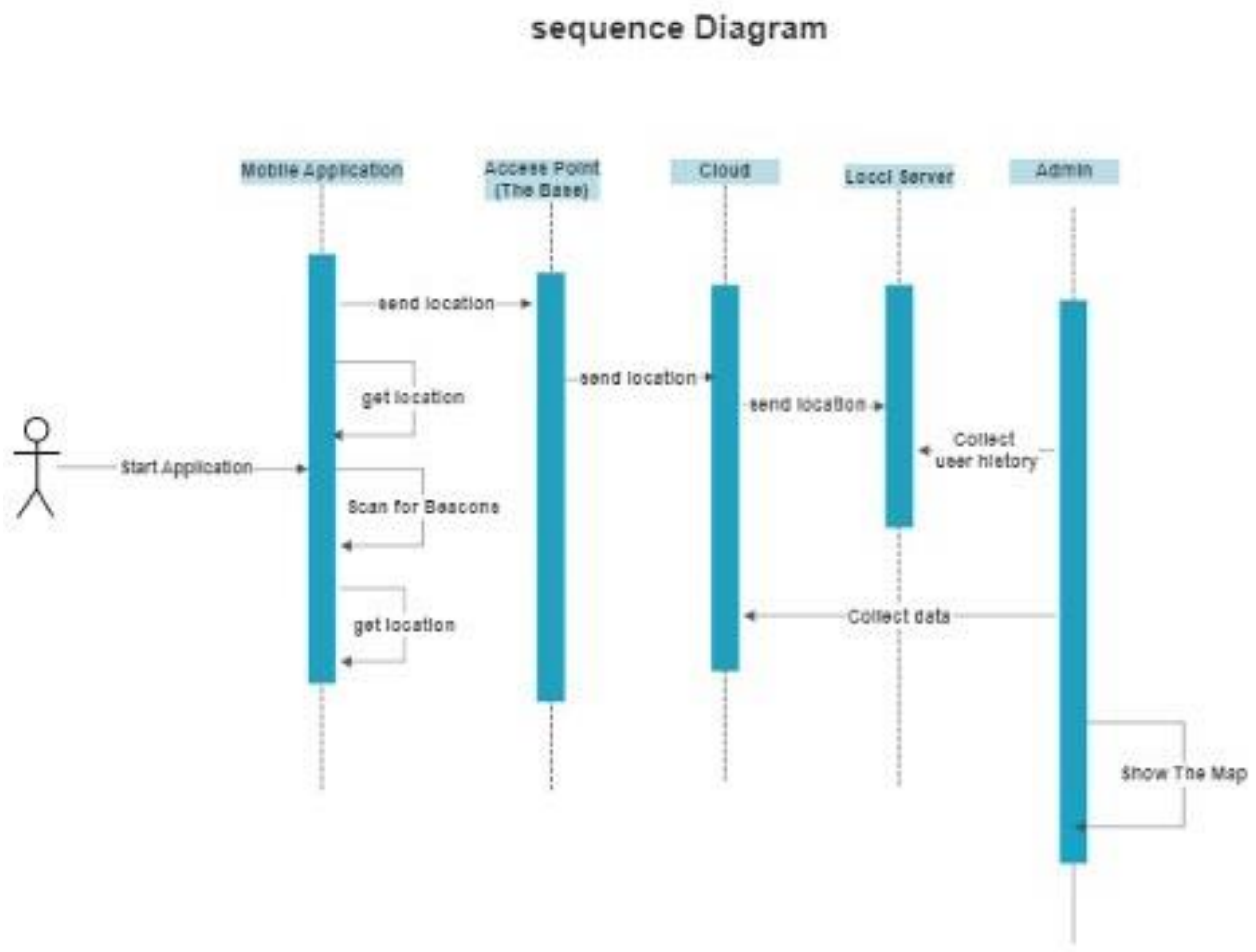
- The system should be designed to produce accurate and consistent results.
- The system should be able to detect and correct errors in location data.

8. Stability:

- The system should be able to operate in a variety of indoor environments, including large buildings, shopping malls, and airports.
- The system should be able to maintain its performance and accuracy over time.

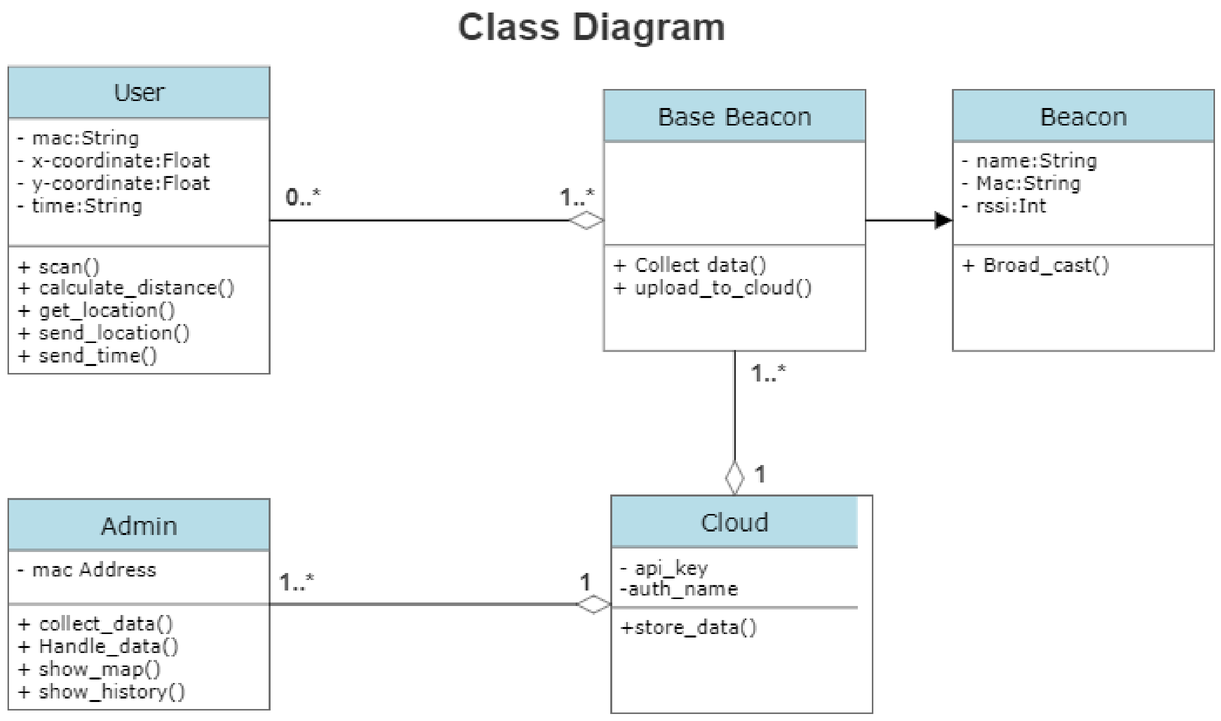
3.2 Use Cases


Indoor Position System
System
user
Open Application
<<include>>
calculate the postition
<<include>>
send data to the base
<<include>>
upload data to cloud
<<include>>
collect data from the cloud
<<extend>>
handle the current and previous data
<<extend>>
Show the map
Admin



3.3 Sequence Diagrams

3.4 Class Diagrams



3.5 Tools and Technologies

3.6 Project Development Methodology

Agile Methodology :

- Approach: Frequent stakeholder interaction
- Flexibility: High
- Requires: Team initiative and short-term deadlines

Agile methodology was developed as a response to Waterfall’s more rigid structure. As a result, it’s a much more fluid form of project management. A software development project can take years to complete, and technology can change significantly during that time. Agile was developed as a flexible method that welcomes incorporating changes of direction even late in the process, as well as accounting for stakeholders’ feedback throughout the process.

In Agile, the team will work on phases of the project concurrently, often with short-term deadlines. Additionally, the team, rather than a project manager, drives the project’s direction. This can empower the team to be motivated and more productive, but also requires a more self-directed team.

Pros	Cons
Short-term deadlines encourage productivity and efficiency	Because team members are working on multiple phases at a time, there is potential for overlap or unnecessary effort spent on later stages if an early phase needs to be modified
There is a lot of flexibility to change project direction and experiment with new directions	Deliverables are not a requirement to progress to the following phase. It can be harder to ensure the entire team is on the same page—especially if it’s a large team with different departments. It also means that work can get lost or miscommunicated between team members, especially when people leave and join the team in the middle of the projects.
The methodology is client-facing, which means that the team shares progress and incorporates feedback into the process	The project timeline is more difficult to determine from the start, and it is also more susceptible to change

Waterfall Methodology :

- Approach: Hands-off; goals and outcome established from the beginning
- Flexibility: Low
- Requires: Completing deliverables to progress to the next phase

Waterfall methodology is a linear form of project management ideal for projects where the end result is clearly established from the beginning of the project. The expectations for the project and the deliverables of each stage are clear and are required in order to progress to the next phase.

Pros	Cons
Provides a concrete plan of the project from start to finish	Because each project phase needs to be completed before progressing to the next stage, the process can take longer
The team establishes project requirements early on, which can save time	You might not realize an issue with a phase until you have already progressed to the next one. This would mean going back through each phase and checking where the mistake or error occurred, which can be a timely process.
Each phase of the project requires a deliverable to progress to the next phase, making the workflow more structured	The Waterfall methodology requires that you outline the project from start to finish before you begin. This doesn't allow for a lot of flexibility or change. Additionally, it can become problematic if the stakeholders disagree on the project’s vision and don’t find out until it is executed or in a later phase.

Agile and Waterfall Comparison Chart

	Waterfall	Agile
Timeline	Waterfall has a fixed timeline. The idea is that the start and finish of the project are already mapped out from the beginning.	Agile is a lot more flexible and accounts for experimenting with different directions. Rather than a fixed timeline, the schedule adapts as the project progresses. The Agile Manifesto, an online document released in 2001 by a group of software developers, says team members are expected to, “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.”
Client Involvement	Once the end goal is established, Waterfall does not involve the client or project owner during the process, apart from specific check-ins or for deliverables. The course of the project is outlined from the start, so incorporating client feedback is not an ongoing part of the process.	A fundamental part of Agile is including clients in the project development at every step. The Agile Manifesto states, “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.” Therefore, business owners are expected to be involved and give feedback to the software development team as they progress through the different phases of the project.
Flexibility	Waterfall is not as flexible as Agile because each phase needs to be fully completed before moving on to the next phase. The project is also planned out ahead of time, making this management system ideal for teams with a clear vision of where they are headed from start to finish.	Flexibility is built into the Agile method. Agile values short bursts of work, which are called sprints. The method welcomes adapting to different directions, incorporating new information even at a later stage of the project.
Budget	Fixed. The budget for projects using the Waterfall methodology is generally fixed. Because the project is determined from start to finish, there is less room to change the budget mid-project.	Flexible. Agile is open to adaptation, encourages experimentation and welcomes changes of direction, even in later phases of the project. Because of this, the budget tends to be more flexible.

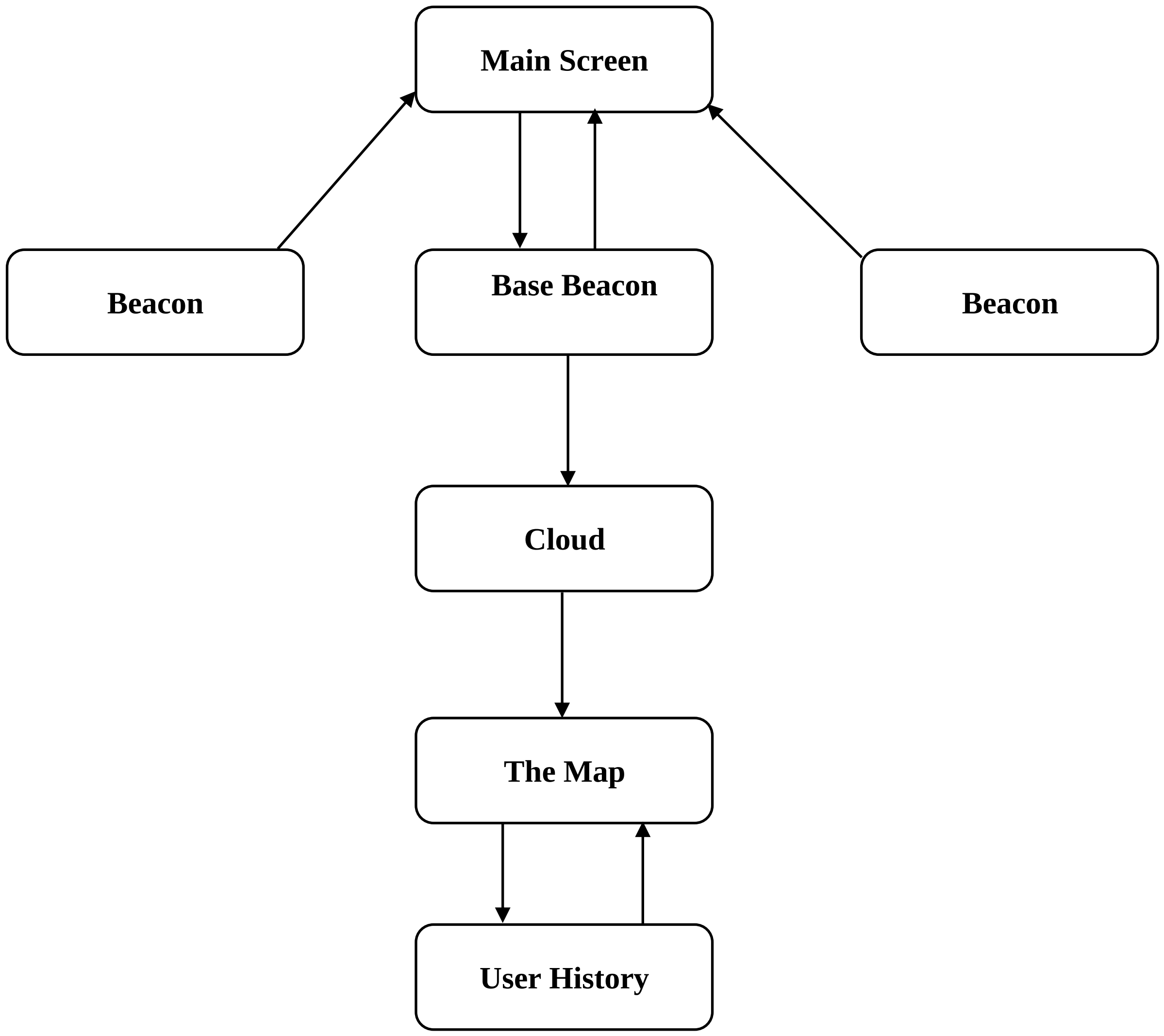
Waterfall is a linear project progression, so it’s best suited for projects with a defined end goal. If a project owner has a clear and specific vision of an app, for example, and is confident it will not change throughout the project development, Waterfall methodologies could be a good system to follow.

Meanwhile, Agile leaves a lot of room to adapt and change course as the project develops. It’s better suited for projects where the outcome may be dependent on more research or testing.

The budget for projects using Waterfall methodologies tends to be less flexible because the project is mapped out from the beginning. With Agile, there is more room to change direction as the project develops, so the budget is also subject to change. Similarly, the timeline with Waterfall is set from the start, while it’s more flexible with Agile and dependent on how the project develops.

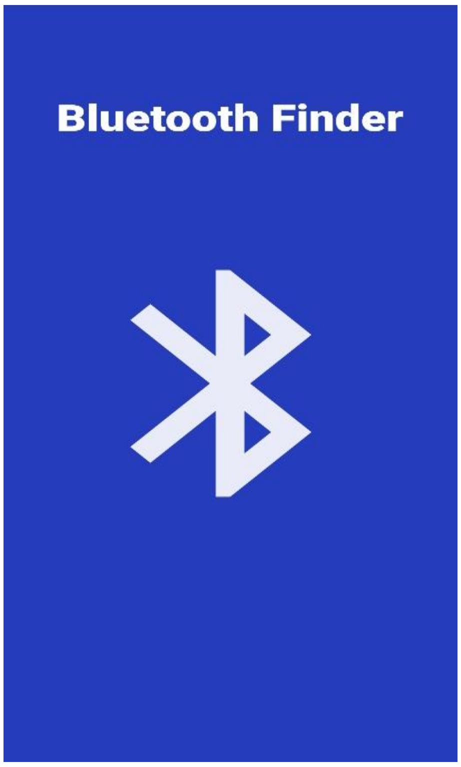
Chapter 4 : System Design

4.1 System Mapping



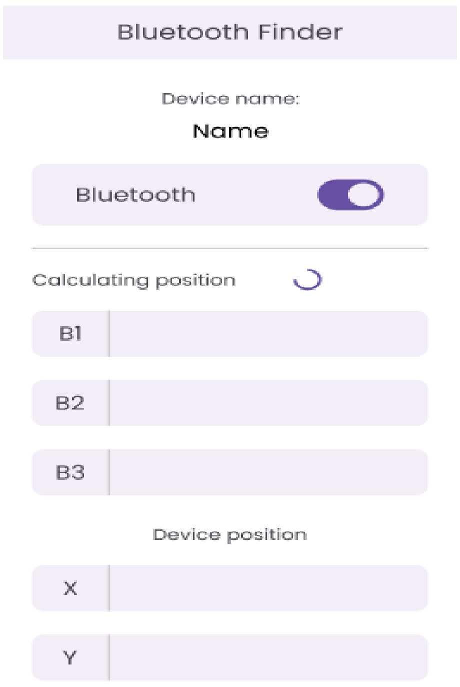
4.2. User Interface Design

4.2.1. Splash Screen

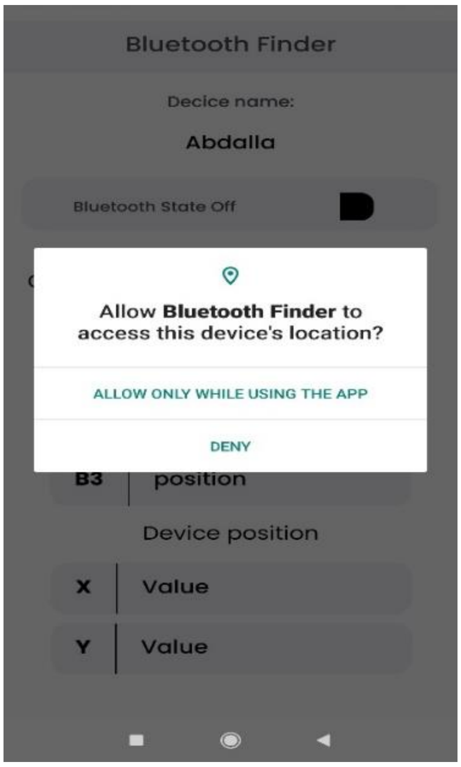


4.2.2. Main Screen Cases

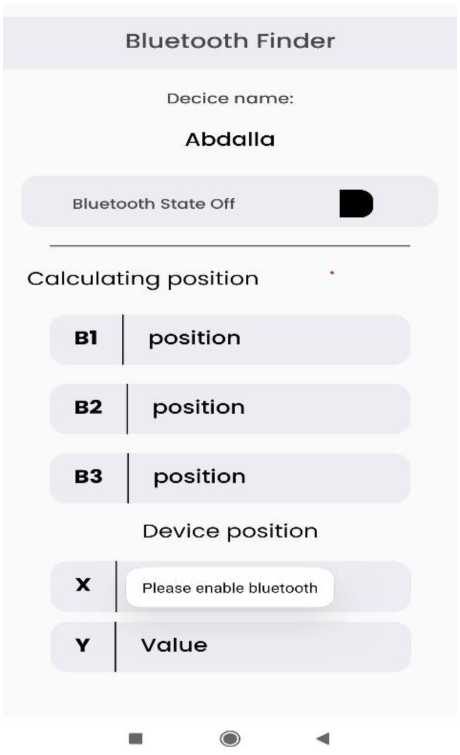
4.2.2.1 UI Design



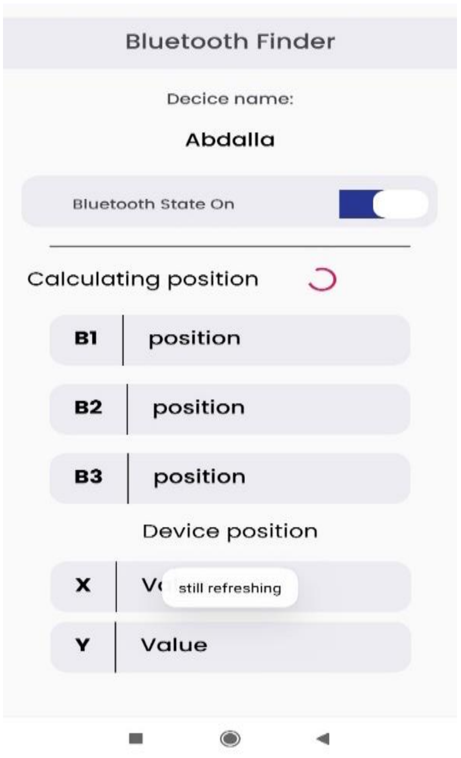
4.2.2.2 Giving permissions to the application



4.2.2.3 Enable Bluetooth to search



4.2.2.4 Still Refresh to find beacons



Bluetooth Finder

Decice name:

Abdalla

Bluetooth State On

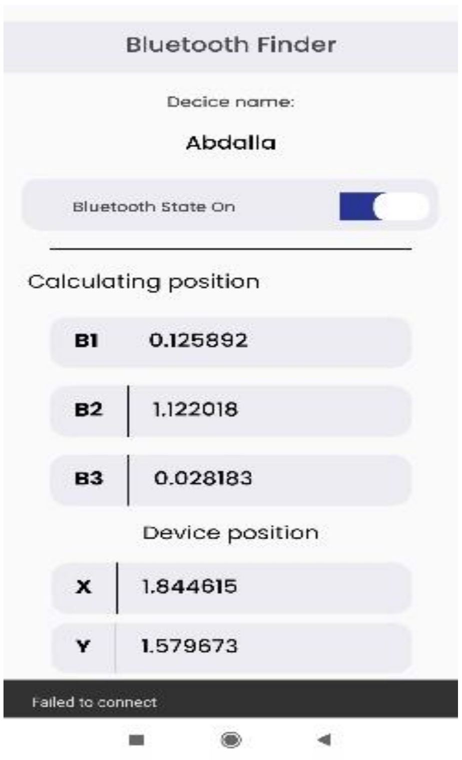
Calculating position

B1	position
B2	position
B3	position

Device position

X	Va still refreshing
Y	Value

4.2.2.5 failed to send data to the base beacon



Bluetooth Finder

Decice name:

Abdalla

Bluetooth State On

Calculating position

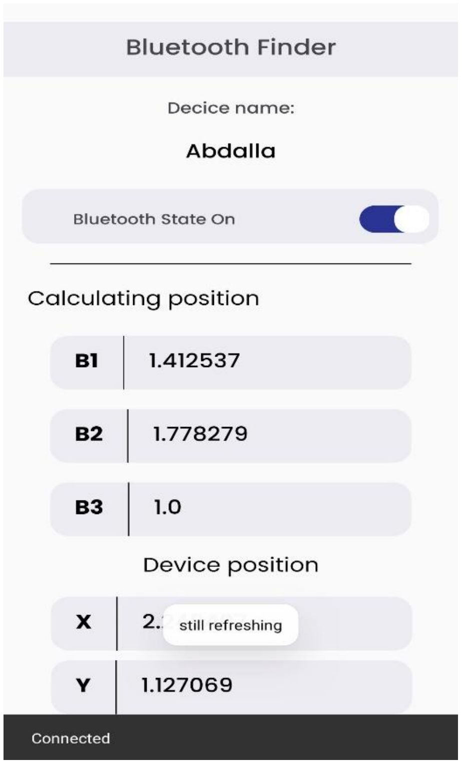
B1	0.125892
B2	1.122018
B3	0.028183

Device position

X	1.844615
Y	1.579673

Failed to connect

4.2.2.6 Connected Successfully and send data to base Beacon



Bluetooth Finder

Decice name:

Abdalla

Bluetooth State On

Calculating position

B1	1.412537
B2	1.778279
B3	1.0

Device position

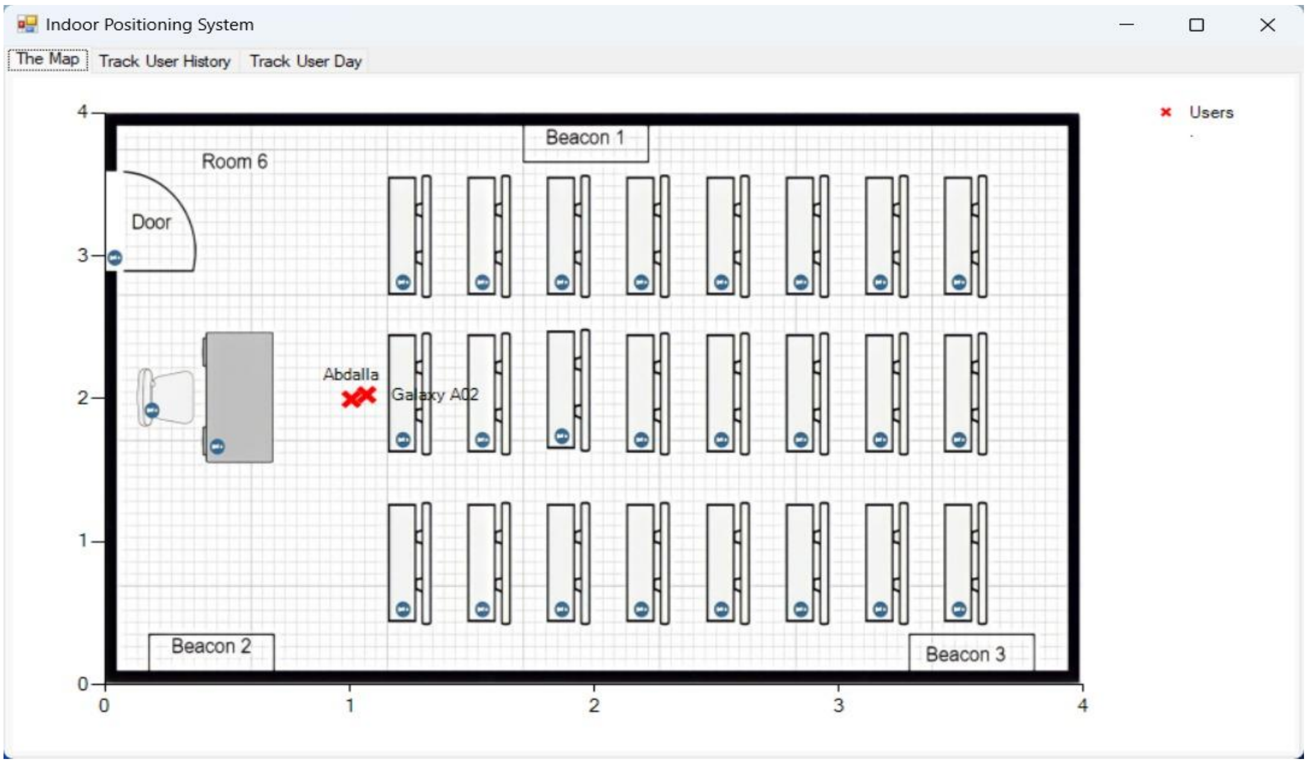
X	2.
Y	1.127069

still refreshing

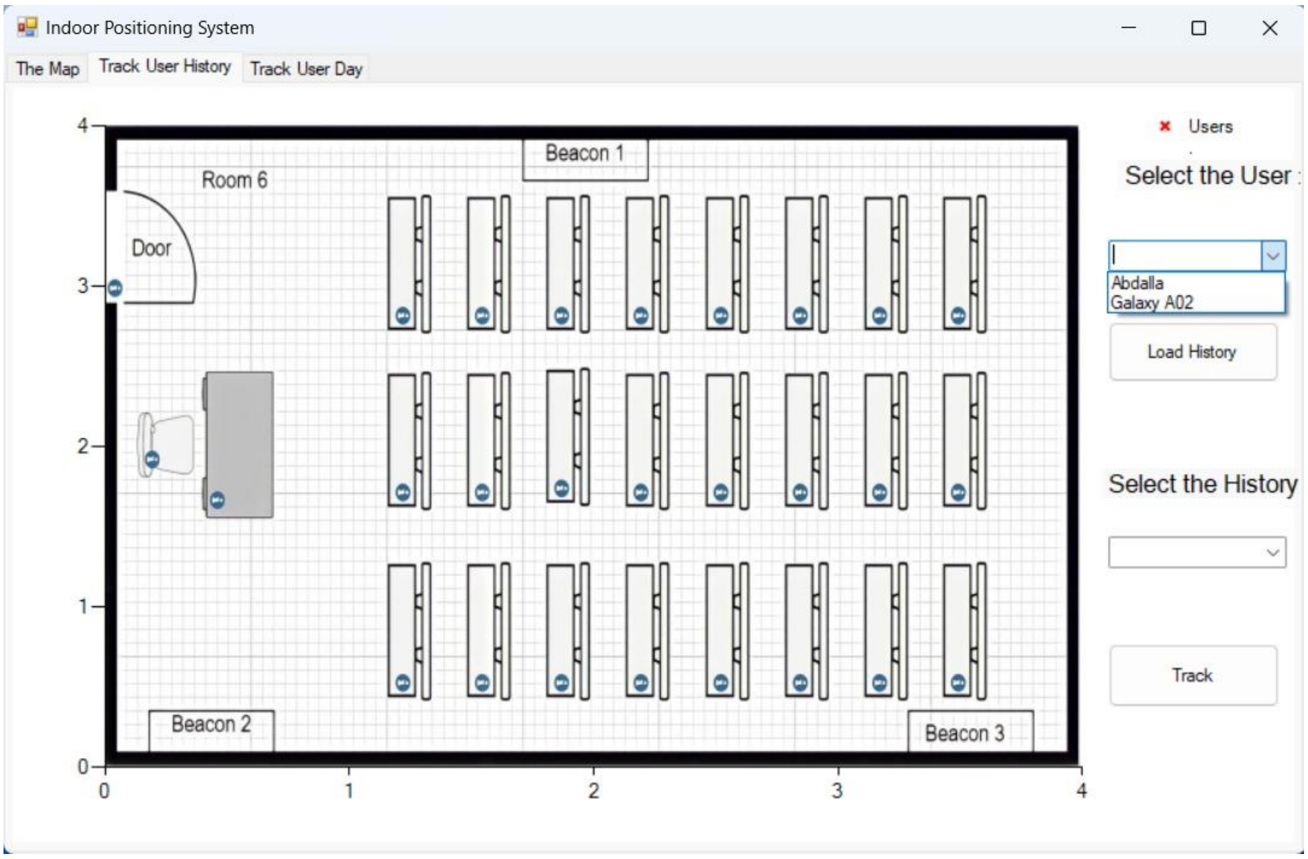
Connected

4.3 Admin Interface Desgin

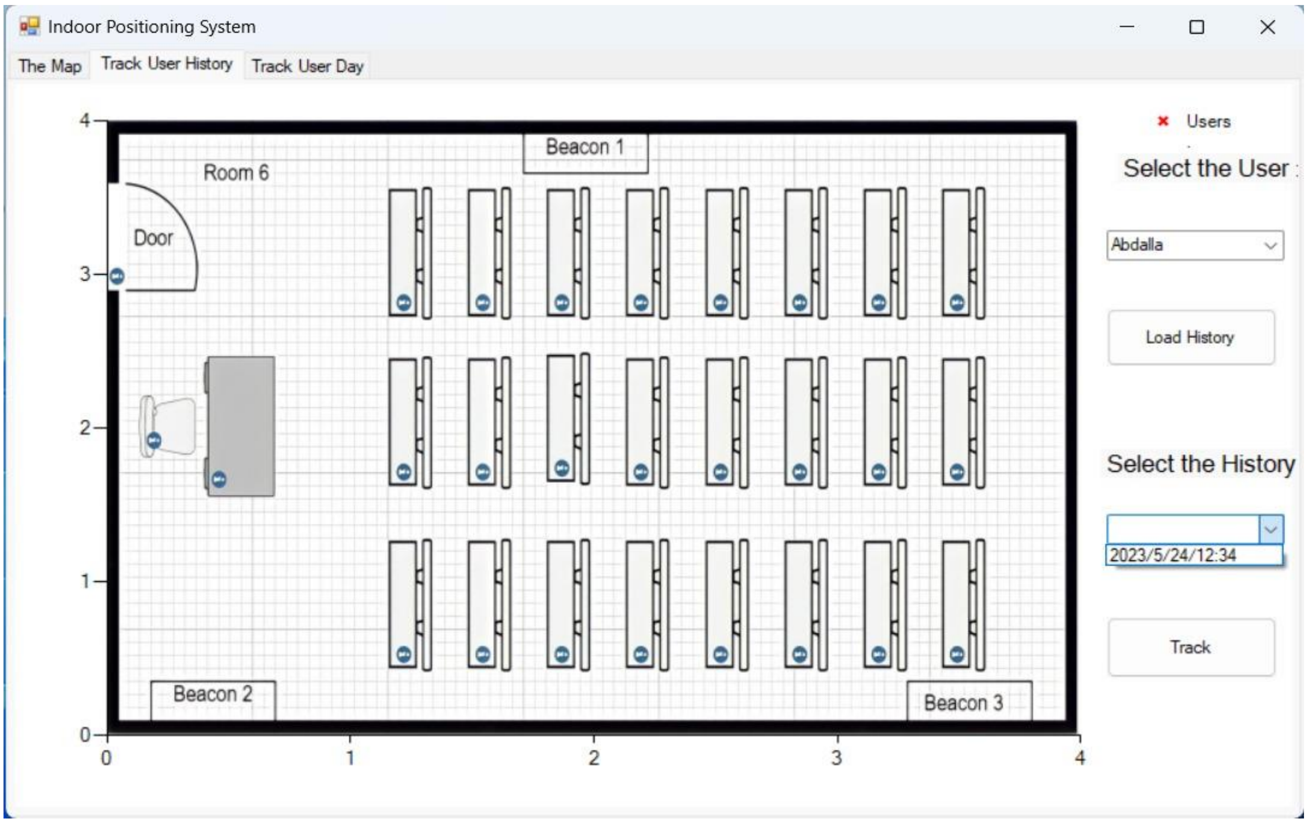
4.3.1 Main Screen that show map on desktop application



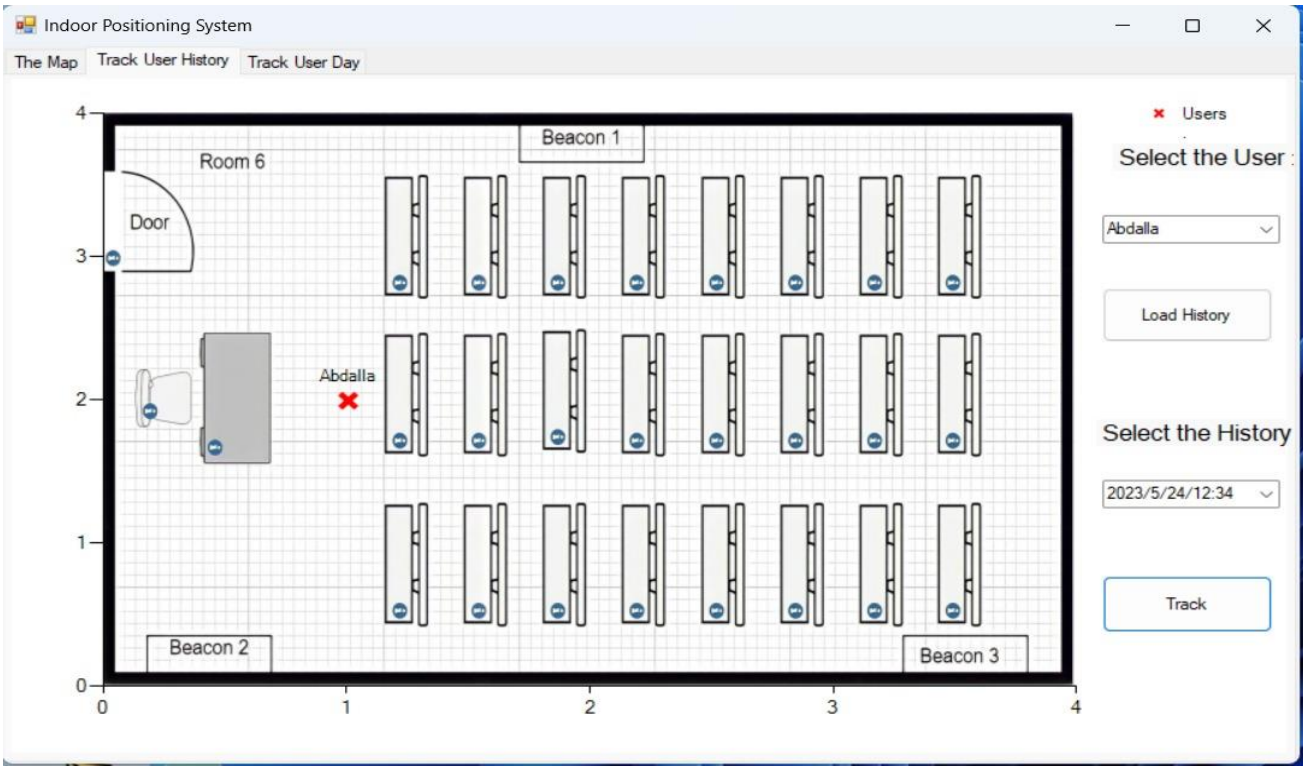
4.3.2 the second screen that shows the history of user tracking



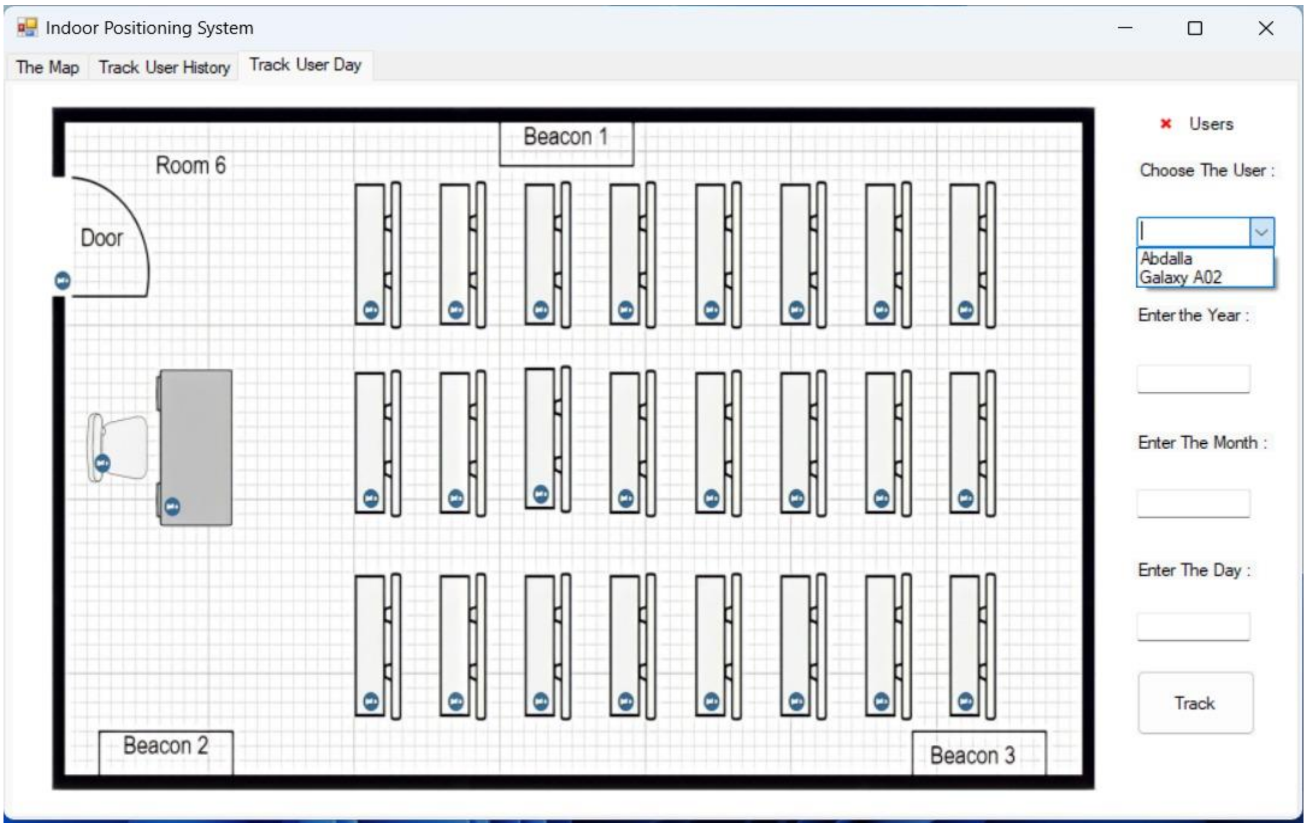
4.3.3 when choose the user and press on button load history



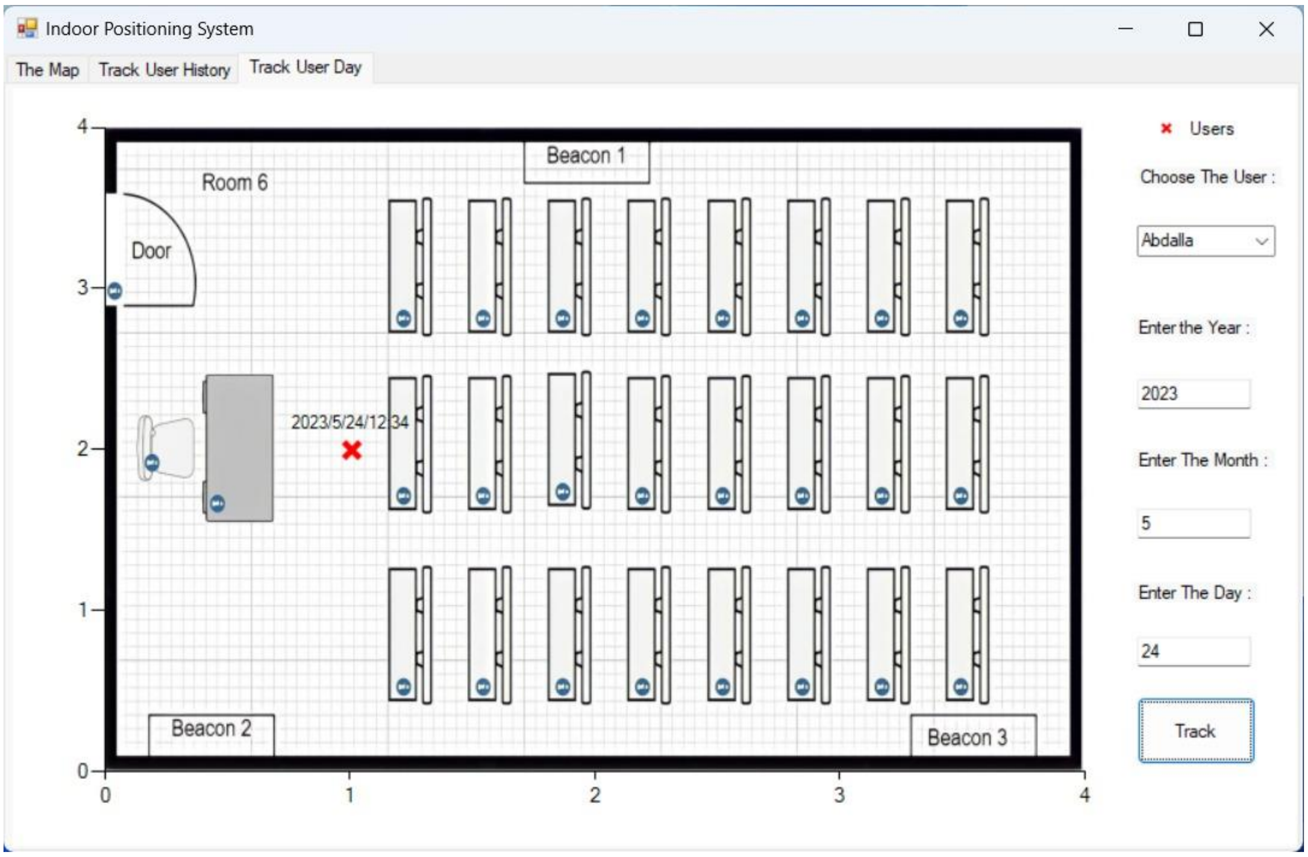
4.3.4 when choose the date and time and press button track



4.3.5 the third screen that shows the history of user in one day



4.3.6 when enter the year, month and day after choose the user



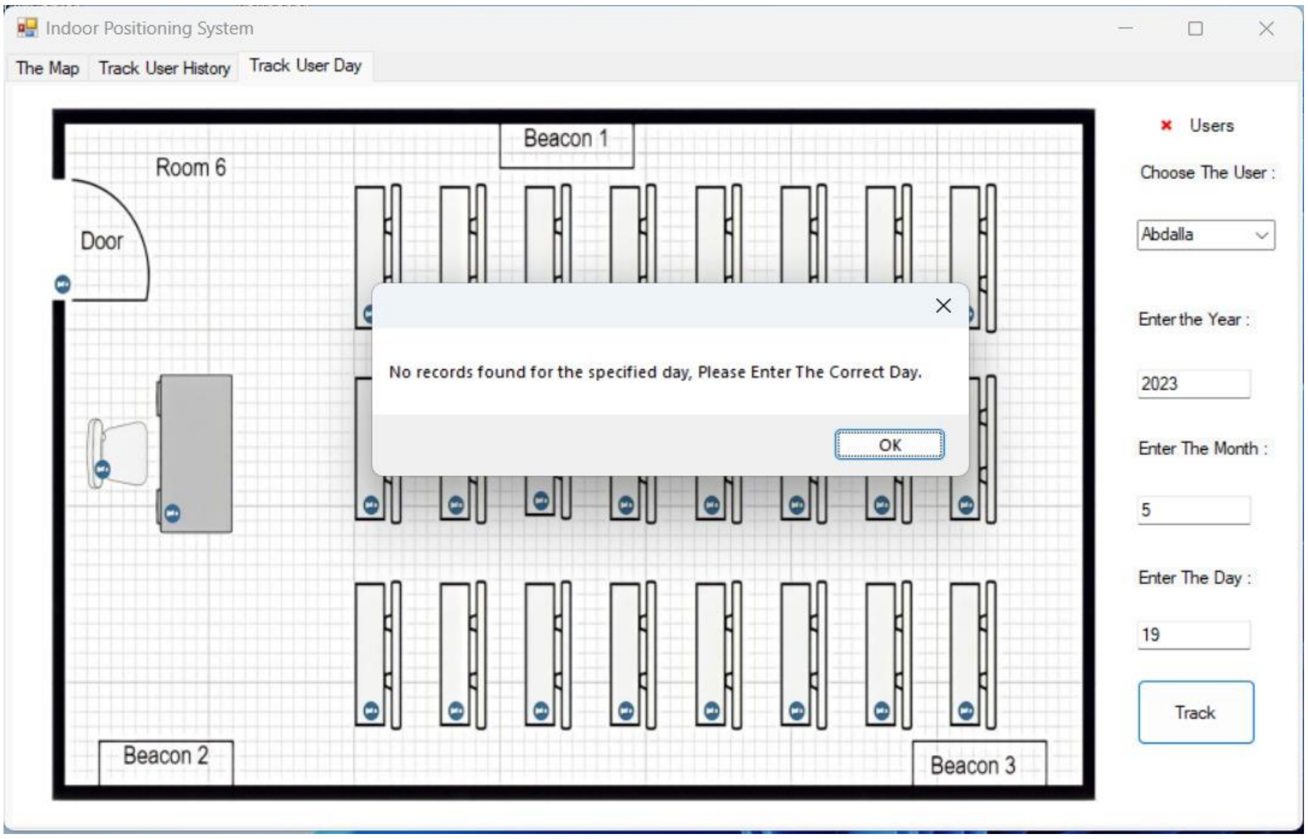
Choose The User : Abdalla

Enter the Year : 2023

Enter The Month : 5

Enter The Day : 24

4.3.7 Error when their no data in chosen day



No records found for the specified day, Please Enter The Correct Day.

Choose The User : Abdalla

Enter the Year : 2023

Enter The Month : 5

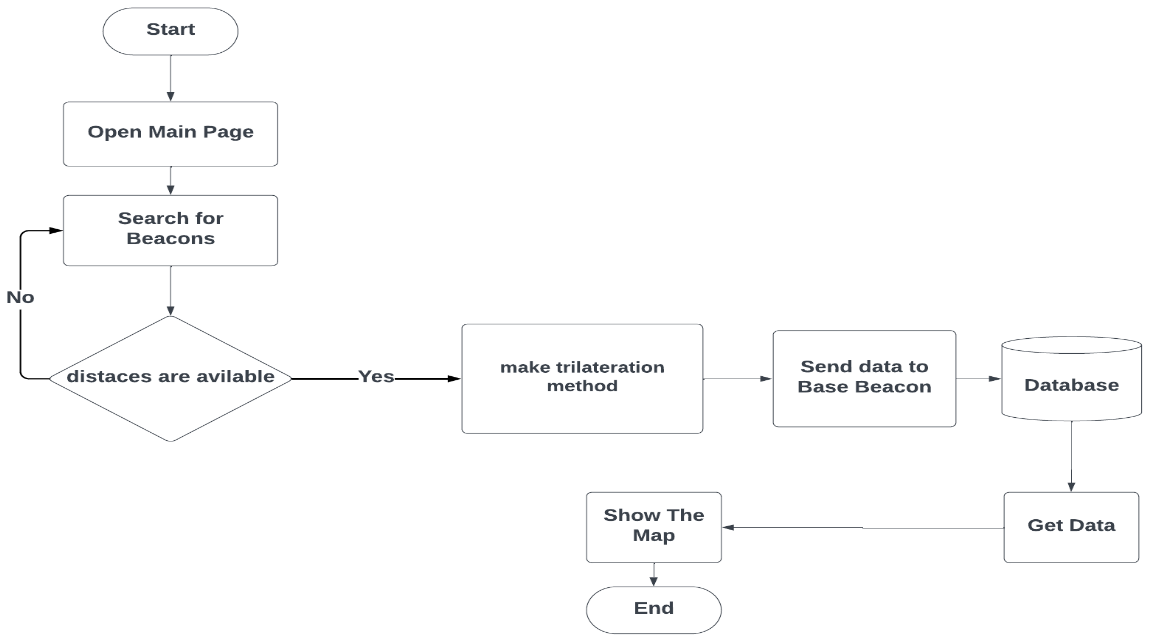
Enter The Day : 19

4.4 Flow Chart

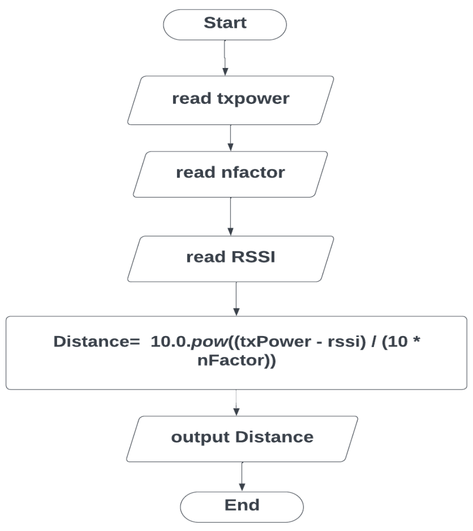
Flowchart:

A visual representation of a process or algorithm using symbols and arrows to illustrate the sequence of steps. It provides a clear and organized way to depict the flow of information or activities, aiding in understanding complex systems, identifying bottlenecks, and communicating ideas effectively. Flowcharts are widely utilized in software development, process analysis, problem-solving, and workflow documentation, facilitating streamlined and efficient workflows

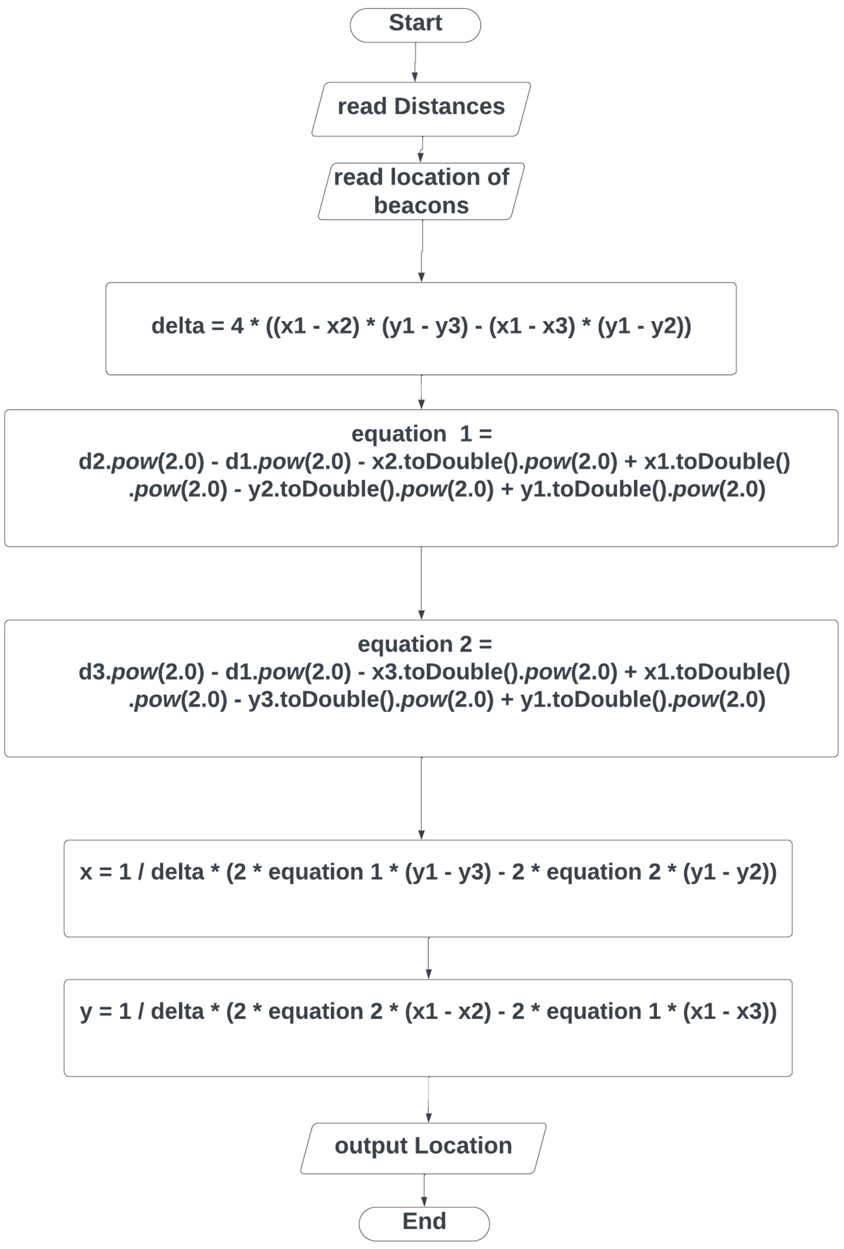
4.4.1 Flow Chart for our project



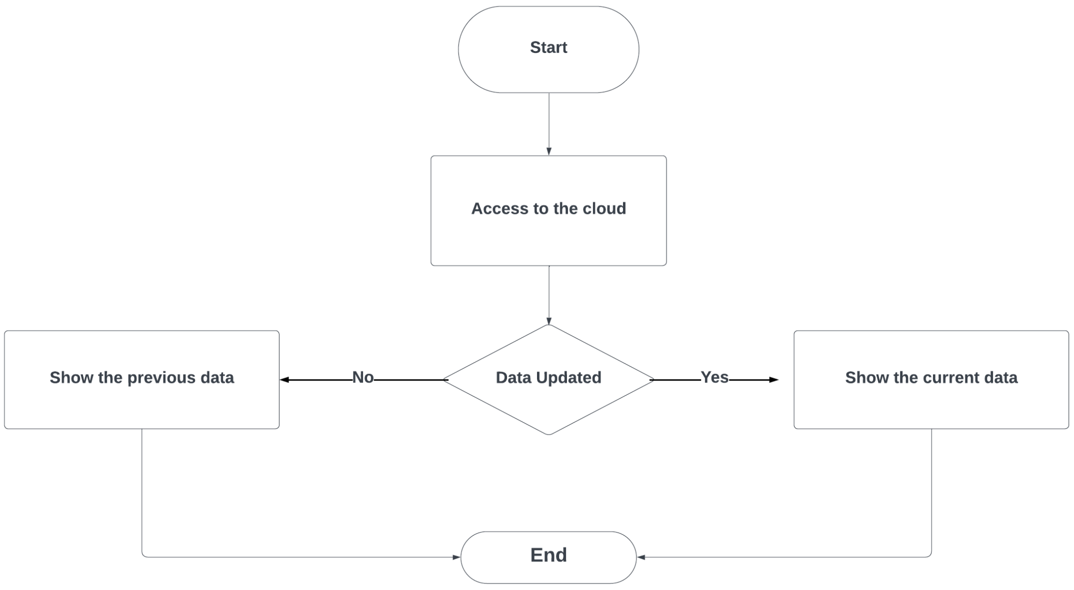
4.4.2 Flow Chart for covert RSSI to distance



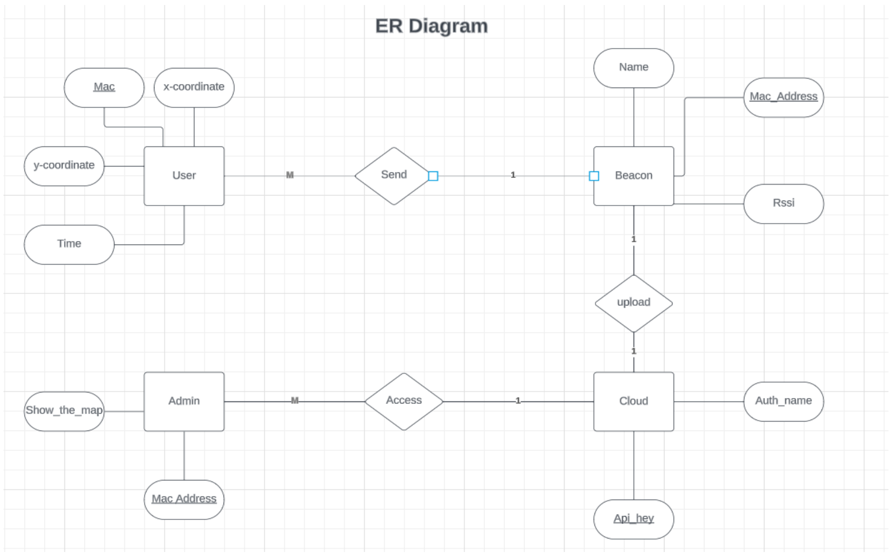
4.4.3 Flow Chart for trilateration method


Start
read Distances
read location of beacons
delta = 4 * ((x1 - x2) * (y1 - y3) - (x1 - x3) * (y1 - y2))
equation 1 = d2.*pow*(2.0) - d1.*pow*(2.0) - x2.toDouble().*pow*(2.0) + x1.toDouble() .*pow*(2.0) - y2.toDouble().*pow*(2.0) + y1.toDouble().*pow*(2.0)
equation 2 = d3.*pow*(2.0) - d1.*pow*(2.0) - x3.toDouble().*pow*(2.0) + x1.toDouble() .*pow*(2.0) - y3.toDouble().*pow*(2.0) + y1.toDouble().*pow*(2.0)
x = 1 / delta * (2 * equation 1 * (y1 - y3) - 2 * equation 2 * (y1 - y2))
y = 1 / delta * (2 * equation 2 * (x1 - x2) - 2 * equation 1 * (x1 - x3))
output Location
End

4.4.4 Flow Chart for Showing map by admin



4.5 Data Base Design



Chapter 5 : Implementation

5.1 Android Code

5.1.1 Android Manifest code :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="c.tlgbltcn.bluetoothfinder">
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/bluetooth_icon"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name="c.tlgbltcn.bluetoothfinder.MainActivity"
            android:exported="true">
        </activity>
        <activity
            android:name="c.tlgbltcn.bluetoothfinder.SplashActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".bluetooth.TryActivity"
            android:exported="true"/>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>
</manifest>
```

5.1.2 Spalsh screen Activity code :

```
package c.tlgbltcn.bluetoothhelper
import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.ImageView
import androidx.appcompat.app.AppCompatActivity

@SuppressLint("CustomSplashScreen")
```

```
class SplashActivity : AppCompatActivity() {
    private val SPLASH_TIME_OUT = 1000L // 1.5 seconds

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        // Set the splash screen layout as the content view
        setContentView(R.layout.activity_splash)
        // Get the ImageView reference
        val imageView = findViewById<ImageView>(R.id.splash_image)
        // Load the animation
        val animation = AnimationUtils.loadAnimation(this, R.anim.fade_rotate)
        // Set the animation listener
        animation.setAnimationListener(object : Animation.AnimationListener {
            override fun onAnimationStart(animation: Animation) {}

            override fun onAnimationEnd(animation: Animation) {
                // Delay for some time after the animation completes before moving to the main activity
                Handler().postDelayed({
                    startActivity(Intent(this@SplashActivity, MainActivity::class.java))
                    finish()            }
, SPLASH_TIME_OUT + animation.duration)         }
            override fun onAnimationRepeat(animation: Animation) {}
        })
        // Start the animation on the ImageView
        imageView.startAnimation(animation)
    }
    override fun onWindowFocusChanged(hasFocus: Boolean) {
        super.onWindowFocusChanged(hasFocus)
        if (hasFocus) {
            // Hide the status bar and enable full-screen immersive mode
            window.decorView.systemUiVisibility = (
                    android.view.View.SYSTEM_UI_FLAG_IMMERSIVE
                            or android.view.View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
                            or android.view.View.SYSTEM_UI_FLAG_FULLSCREEN
                            or android.view.View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                            or android.view.View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                            or android.view.View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                    )}}}
```

5.1.3 Splash screen XML code:

```
<?xml version="1.0" encoding="utf-8"?>
<!--suppress ALL -->
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FF253CBD">
    >
    <ImageView
        android:id="@+id/splash_image"
        android:layout_width="296dp"
        android:layout_height="339dp"
        android:layout_marginStart="77dp"
        android:layout_marginEnd="78dp"
        android:elevation="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@drawable/ic_bluetooth2_24dp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:contentDescription="Bluetooth finder"
         />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Bluetooth Finder"
        android:textColor="#ffffff"
        android:textSize="40sp"
        android:textStyle="bold"
        app:fontFamily="sans-serif-black"
        app:layout_constraintBottom_toTopOf="@+id/splash_image"
        app:layout_constraintEnd_toEndOf="@+id/splash_image"
        app:layout_constraintStart_toStartOf="@+id/splash_image"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="HardcodedText" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5.1.4 activity_main XML code :

```
<?xml version="1.0" encoding="utf-8"?>
<!--suppress ALL -->
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
    <data>
    </data>
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:id="@+id/constraintLayout2">
        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/recycler_view"
            android:layout_width="0dp"
            android:layout_height="0dp"
            android:layout_margin="16dp"
            android:visibility="gone"
            app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:listitem="@layout/item_bluetooth_device" />
        <Button
            android:id="@+id/add_xy"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:background="@drawable/button_background"
            android:text="add xy"
            android:visibility="gone"
            app:layout_constraintEnd_toEndOf="@id/recycler_view"
            app:layout_constraintStart_toStartOf="@id/recycler_view"
            app:layout_constraintTop_toBottomOf="@id/recycler_view" />
        <ProgressBar
            android:id="@+id/progressBar"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_marginStart="44dp"
            android:layout_marginTop="20dp"
            android:layout_marginEnd="107dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toEndOf="@+id/textView3"
            app:layout_constraintTop_toBottomOf="@+id/view" />
        <LinearLayout
            android:id="@+id/linearLayout"
            android:layout_width="match_parent"
            android:layout_height="56dp"
            android:layout_marginTop="20dp"
            android:background="#146750A4"
            android:gravity="start|center_vertical"
            android:orientation="horizontal"
            android:paddingLeft="4dp"
            android:paddingTop="8dp"
            android:paddingRight="4dp"
            android:paddingBottom="8dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent">
            <TextView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:fontFamily="@font/poppins_medium"
                android:lineHeight="22sp"
                android:text="Bluetooth Finder"
                android:textAlignment="center"
                android:textColor="#FF49454F"
                android:textSize="22sp" />
        </LinearLayout>
        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:fontFamily="@font/poppins_medium"
            android:text="Decice name:"
            android:textAlignment="center"
            android:textColor="#FF49454F"
            android:textSize="16sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/linearLayout" />
        <TextView
            android:id="@+id/device_name2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="14dp"
            android:fontFamily="@font/poppins_medium"
            android:text="Name"
            android:textAlignment="center"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
```

```
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="360dp"
    android:layout_height="56dp"
    android:layout_marginStart="25dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="26dp"
    android:background="@drawable/rounded_background"
    android:gravity="start|center_vertical"
    android:orientation="horizontal"
    android:paddingLeft="4dp"
    android:paddingTop="8dp"
    android:paddingRight="4dp"
    android:paddingBottom="8dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/device_name2">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/blueState"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:fontFamily="@font/poppins_medium"
            android:lineHeight="20sp"
            android:paddingStart="40dp"
            android:text="Bluetooth"
            android:textAlignment="center"
            android:textColor="#FF49454F"
            android:textSize="15sp" />
        <Switch
            android:id="@+id/enable_disable_switch"
            android:layout_width="77dp"
            android:layout_height="45dp"
            android:layout_marginStart="90dp"
            android:checked="true"
            android:minHeight="48dp"
            android:thumb="@drawable/custom_switch_thumb"
            android:track="@drawable/custom_switch_track"
            tools:ignore="RtlSymmetry,UseSwitchCompatOrMaterialXml,TouchTargetSizeCheck" />
    </LinearLayout>
</LinearLayout>
<View
    android:id="@+id/view"
    android:layout_width="312dp"
    android:layout_height="1dp"
    android:layout_gravity="center_horizontal"
    android:layout_marginStart="24dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="24dp"
    android:background="#FF000000"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout2" />
<TextView
    android:id="@+id/textView3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
```

```
        android:layout_marginTop="20dp"
        android:layout_marginEnd="175dp"
        android:fontFamily="@font/poppins_regular"
        android:text="Calculating position"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/view" />
    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="119dp"
        android:layout_marginTop="15dp"
        android:layout_marginEnd="119dp"
        android:fontFamily="@font/poppins_regular"
        android:text="Device position"
        android:textColor="#000000"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout5" />
    <LinearLayout
        android:id="@+id/linearLayout3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="24dp"
        android:background="@drawable/rounded_background_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView3">
        <TextView
            android:id="@+id/textView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_bold"
            android:text="B1"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />
        <View
            android:layout_width="1dp"
            android:layout_height="match_parent"
            android:background="#FF000000" />
        <TextView
            android:id="@+id/pos1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_medium"
            android:maxLines="1"
            android:maxLength="8"
            android:text="position1"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="24dp"
        android:background="@drawable/rounded_background_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout3">
        <TextView
            android:id="@+id/textView10"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_bold"
            android:text="B2"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />
        <View
            android:layout_width="1dp"
            android:layout_height="match_parent"
            android:background="#FF000000" />
        <TextView
            android:id="@+id/pos2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:maxLength="8"
            android:fontFamily="@font/poppins_medium"
            android:maxLines="1"
            android:text="position2"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="24dp"
        android:background="@drawable/rounded_background_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout4">
        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_bold"
            android:text="B3"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />
        <View
            android:layout_width="1dp"
            android:layout_height="match_parent"
            android:background="#FF000000" />
        <TextView
            android:id="@+id/pos3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_medium"
            android:maxLines="1"
            android:maxLength="8"
            android:text="position3"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="14dp"
        android:layout_marginEnd="49dp"
        android:background="@drawable/rounded_background_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView5">
        <TextView
            android:id="@+id/textView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_bold"
            android:text="X"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />
        <View
            android:layout_width="1dp"
            android:layout_height="match_parent"
            android:background="#FF000000" />
        <TextView
            android:id="@+id/numX"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:maxLength="8"
            android:fontFamily="@font/poppins_medium"
            android:maxLines="1"
            android:text="Value"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout8"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="49dp"
        android:background="@drawable/rounded_background_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout7">
        <TextView
            android:id="@+id/textView8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_bold"
            android:text="Y"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/textView3" />
        <View
            android:layout_width="1dp"
            android:layout_height="match_parent"
            android:background="#FF000000" />
        <TextView
            android:id="@+id/numY"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="21dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="21dp"
            android:layout_marginBottom="10dp"
            android:fontFamily="@font/poppins_medium"
            android:maxLength="8"
            android:maxLines="1"
            android:text="Value"
            android:textColor="#000000"
            android:textSize="20sp"
            app:layout_constraintTop_toBottomOf="@+id/textView" />
        </LinearLayout>
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

5.1.5 Main Activity Code :

```
package c.tlgbltcn.bluetoothfinder
import android.annotation.SuppressLint
import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothDevice
import android.os.Build
import android.os.Bundle
import android.os.Handler
import android.util.Log
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import c.tlgbltcn.bluetoothfinder.bluetooth.BluetoothDeviceModel
import c.tlgbltcn.bluetoothfinder.bluetooth.BluetoothListAdapter
import c.tlgbltcn.library.BluetoothHelper
import c.tlgbltcn.library.BluetoothHelperListener
import android.view.View
import android.view.Window
import androidx.annotation.RequiresApi
import androidx.constraintlayout.widget.ConstraintLayout
import androidx.lifecycle.lifecycleScope
import c.tlgbltcn.bluetoothfinder.bluetooth.MyBluetoothManager
import c.tlgbltcn.bluetoothhelper.R
import c.tlgbltcn.bluetoothhelper.databinding.ActivityMainBinding
import com.google.android.material.snackbar.Snackbar
import kotlinx.coroutines.*
import java.lang.Runnable
import java.time.LocalDateTime
import kotlin.collections.ArrayList
@Suppress("DEPRECATION")
@SuppressLint("SetTextI18n", "MissingPermission")
class MainActivity : AppCompatActivity(), BluetoothHelperListener {
   companion object {
      private const val TAG = "MainActivity"
   }
   //esp MacAddress
   private val espAddress = "CC:DB:A7:15:72:FA"
   // getting the current time
   @RequiresApi(Build.VERSION_CODES.O)
   val currentTime = LocalDateTime.now()
   //BLUETOOTH VARIABLES
```

```
private lateinit var bluetoothHelper: BluetoothHelper
private lateinit var binding: ActivityMainBinding
private lateinit var handler: Handler
private lateinit var refreshRunnable: Runnable
private lateinit var viewAdapter: RecyclerView.Adapter<*>
private lateinit var bluetoothAdapter: BluetoothAdapter
private lateinit var bluetoothManager: MyBluetoothManager
private val xyObservable = ValueObservable<Pair<Double, Double>>()
private lateinit var layout: ConstraintLayout
private var itemList = ArrayList<BluetoothDeviceModel>()
private var distanceFromB1: Double = 0.0
private var distanceFromB2: Double = 0.0
private var distanceFromB3: Double = 0.0
private var distance1Check: Double = 0.0
private var distance2Check: Double = 0.0
private var distance3Check: Double = 0.0
private var x: Double = 0.0
private var y: Double = 0.0
private var deviceName = ""
private val methods = Methods()
private val coroutineScope = lifecycleScope // or viewModelScope if using ViewModel
@RequiresApi(Build.VERSION_CODES.O)
@SuppressLint("SetTextI18n", "MissingPermission")
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //Hiding Navigation bar
    supportRequestWindowFeature(Window.FEATURE_NO_TITLE)
    window.decorView.systemUiVisibility = View.SYSTEM_UI_FLAG_HIDE_NAVIGATION
    // Initializing the binding
    binding = ActivityMainBinding.inflate(layoutInflater)
    val view = binding.root
    //Set the view to the layout
    setContentView(view)
    layout = findViewById(R.id.constraintLayout2)
    // Initializing the bluetoothHelper Library , The Bluetooth Manager and the Bluetooth Adapter
    bluetoothHelper =
        BluetoothHelper(this@MainActivity, this@MainActivity).setPermissionRequired(true)
            .create()
    bluetoothManager = MyBluetoothManager(espAddress)
   bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
   //Handler to schedule and run code on a specific thread
    handler = Handler()
    //refreshRunnable for the schedule of bluetooth scanning processes
    refreshRunnable = object : Runnable {
        override fun run() {

            if (bluetoothHelper.isBluetoothScanning()) {
                bluetoothHelper.stopDiscovery()
            } else {
                bluetoothHelper.startDiscovery()
            }
            // Schedule next click after 1 second
            handler.postDelayed(this, 1000)
       }       }
    // Check the current state of Bluetooth and update the Switch state
    if (bluetoothAdapter.isEnabled) {
        binding.enableDisableSwitch.isChecked = true
        binding.blueState.text = "Bluetooth State On"
        handler.post(refreshRunnable)
    } else {
        binding.enableDisableSwitch.isChecked = false
        binding.blueState.text = "Bluetooth State Off"
        Toast.makeText(this@MainActivity, "Please enable bluetooth", Toast.LENGTH_LONG).show()
  }
   //BINDING
    xyObservable.addObserver { (newX, newY) ->
        sendToEsp(currentTime, deviceName, newX, newY)
        binding.numX.text = newX.toString()
        binding.numY.text = newY.toString()
        binding.progressBar.visibility = View.GONE
    }
    binding.enableDisableSwitch.setOnCheckedChangeListener { buttonView, isChecked ->
        if (isChecked) {
            // Turn on Bluetooth
            bluetoothAdapter.enable()
            binding.blueState.text = "Bluetooth State Off"
            handler.post(refreshRunnable)


        } else {
            // Turn off Bluetooth
            bluetoothAdapter.disable()
            binding.blueState.text = "Bluetooth State On"
       }       }
    deviceName = bluetoothAdapter.name
    binding.deviceName2.text = deviceName
    viewAdapter = BluetoothListAdapter(itemList)
    binding.recyclerView.apply {
        layoutManager = LinearLayoutManager(this@MainActivity)
        adapter = viewAdapter        }    }
//The Formatted time which will be send with the XY values to the esp
@RequiresApi(Build.VERSION_CODES.O)
fun formatTime(): String {
    val now = LocalDateTime.now()
    val year = now.year
    val months = now.month.value
    val days = now.dayOfMonth
    var hours = now.hour
    val minutes = now.minute
    var hours2 = 0
    if (hours > 12) {
        hours2 = hours.minus(12)
        return "$year/$months/$days/$hours2:$minutes"
    } else {
        return "$year/$months/$days/$hours:$minutes"
    }  }
//Method of filtering the devices according to it's rssi signal and device name from the recyclerView
// , then calculates the distance using trilateration method
@SuppressLint("NotifyDataSetChanged")
override fun getBluetoothDeviceList(device: BluetoothDevice?, rssi: String) {
    val distance = methods.calculateDistance(rssi.toInt())
    if (distance < 3) {
        when (device?.name) {
            "Beacon1" -> {
                itemList.add(
                    BluetoothDeviceModel(
                        "Device Name :" + device.name,
                        "Device Mac : " + device.address,
                        "Distance : $distance "
                    )
            )
            viewAdapter.notifyDataSetChanged()
            val distance1 = methods.calculateDistance(rssi.toInt())
            distanceFromB1 = distance1
            distance1Check = distance1
            binding.pos1.text = "$distance"
            Log.i(TAG, "Device name: ${device.address}  RSSI: $rssi  Distance:$distance")
          }
          "Beacon2" -> {
            itemList.add(
              BluetoothDeviceModel(
                "Device Name :" + device.name,
                "Device Mac : " + device.address,
                "Distance : $distance "
              )
            )
            viewAdapter.notifyDataSetChanged()
            val distance2 = methods.calculateDistance(rssi.toInt())
            distanceFromB2 = distance2
            distance2Check = distance2
            binding.pos2.text = "$distance"
          }
          "Beacon3" -> {
            itemList.add(
              BluetoothDeviceModel(
                "Device Name :" + device.name,
                "Device Mac : " + device.address,
                "Distance : $distance "                    )                    )
            viewAdapter.notifyDataSetChanged()
            val distance3 = methods.calculateDistance(rssi.toInt())
            distanceFromB3 = distance3
            distance3Check = distance3
            binding.pos3.text = "$distance"
            Log.i(TAG, "Device name: ${device.address}  RSSI: $rssi  Distance:$distance")
          }
        }
        if ((distance1Check == 0.0 || distance2Check == 0.0 || distance3Check==0.0))
        {
          Toast.makeText(this, "still refreshing", Toast.LENGTH_LONG).show()
        } else {
          x = methods.trilateration(distanceFromB1, distanceFromB2, distanceFromB3).first
          y = methods.trilateration(distanceFromB1, distanceFromB2, distanceFromB3).second
          xyObservable.setValue(x to y)
          distance1Check = 0.0
          distance2Check = 0.0
          distance3Check = 0.0
          binding.pos1.text = ""
          binding.pos2.text = ""
          binding.pos3.text = ""
        }     }   }
  // Method of sending the values to the esp using the BluetoothManager class
  @RequiresApi(Build.VERSION_CODES.O)
  private fun sendToEsp(time: LocalDateTime, deviceName: String, x: Double, y: Double) {
    coroutineScope.launch {
      try {
        if (bluetoothManager.connect()) {
          Snackbar.make(layout, "Connected", Snackbar.LENGTH_SHORT).show()
          val newTime = formatTime()
          bluetoothManager.sendData("$deviceName,")
          bluetoothManager.sendData("$x,")
```

```
                bluetoothManager.sendData("$y,")
                bluetoothManager.sendData("$newTime,")
                Toast.makeText(this@MainActivity, "Sent successfully", Toast.LENGTH_LONG).show()

                bluetoothManager.disconnect()
            } else {
                Snackbar.make(layout, "Failed to connect", Snackbar.LENGTH_SHORT).show()
            }
        } catch (e: Exception) {
            // Handle any exceptions that occur during the coroutine execution
            Toast.makeText(this@MainActivity, "Error: ${e.message}", Toast.LENGTH_SHORT).show()
        }      }   }
    // This method is called when the Bluetooth discovery process starts,
    // It resets the distances from Beacon1, Beacon2, and Beacon3 to 0.0
    override fun onStartDiscovery() {
        distanceFromB1 = 0.0
        distanceFromB2 = 0.0
        distanceFromB3 = 0.0
    }
    // This method is called when the Bluetooth discovery process is finished. It clears the item list.
    override fun onFinishDiscovery() {
        itemList.clear()
    }
    // This method is called when the Bluetooth state is enabled. It updates the UI to indicate that the
Bluetooth state is "On".
    override fun onEnabledBluetooth() {
        binding.blueState.text = "Bluetooth State On"
    }
    // This method is called when the Bluetooth state is disabled. It updates the UI to indicate that the
Bluetooth state is "Off".
    override fun onDisabledBluetooh() {
        binding.blueState.text = "Bluetooth State Off"

    }
    // This method is called when the activity resumes. It calls the super method and registers for
Bluetooth state change notifications.
    override fun onResume() {
        super.onResume()
        bluetoothHelper.registerBluetoothStateChanged()
    }
    // This method is called when the activity is stopped. It calls the super method and unregisters the
Bluetooth state change notifications.
    override fun onStop() {
        super.onStop()
        bluetoothHelper.unregisterBluetoothStateChanged()
    }
}
```

5.1.6.1 Methods Class (calculate distance Method):

```
package c.tlgbltcn.bluetoothfinder
import android.annotation.SuppressLint
import android.bluetooth.BluetoothAdapter
import android.bluetooth.BluetoothDevice
import android.bluetooth.BluetoothSocket
import kotlin.math.pow
import android.content.Context
import android.net.wifi.WifiManager
import android.util.Log
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.withContext
import java.io.IOException
import java.io.OutputStream
import java.util.*
class Methods {
   //VARIABLES
   private val TAG = "Methods"
   private val espAddress="CC:DB:A7:15:72:FA"
   private lateinit var bluetoothDevice: BluetoothDevice
   private lateinit var outputStream: OutputStream
   private lateinit var bluetoothAdapter: BluetoothAdapter
   private lateinit var bluetoothSocket: BluetoothSocket
   //METHODS
   fun calculateDistance(rssi: Int, txPower: Int = -59, nFactor: Double = 2.0): Double {
      return 10.0.pow((txPower - rssi) / (10 * nFactor))
   }
```

5.1.6.2 Methods Class (trilateration Method):

```
fun trilateration(d1: Double, d2: Double, d3: Double): Pair<Double, Double> {
   var x = 0.0
   var y = 0.0
   val x1 = 0.0f
   val y1 = 0.0f
   val x2 = 4.0f
   val y2 = 0.0f
   val x3 = 2.0f
   val y3 = 4.0f
   val delta = 4 * ((x1 - x2) * (y1 - y3) - (x1 - x3) * (y1 - y2))
   val a: Double =
      d2.pow(2.0) - d1.pow(2.0) - x2.toDouble().pow(2.0) + x1.toDouble()
         .pow(2.0) - y2.toDouble().pow(2.0) + y1.toDouble().pow(2.0)
   val b: Double =
      d3.pow(2.0) - d1.pow(2.0) - x3.toDouble().pow(2.0) + x1.toDouble()
         .pow(2.0) - y3.toDouble().pow(2.0) + y1.toDouble().pow(2.0)

   x = 1 / delta * (2 * a * (y1 - y3) - 2 * b * (y1 - y2))
   y = 1 / delta * (2 * b * (x1 - x2) - 2 * a * (x1 - x3))
   return Pair(x, y)
}
```

5.2.1 Beacons Broadcast Code

```
#include <SoftwareSerial.h>
SoftwareSerial BTSerial(0,1); // RX | TX
void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600); // Bluetooth module baud rate
}
void loop() {
  if (BTSerial.available()) {
    String data = BTSerial.readString();
    Serial.println(data);  }  }
```

5.2.2 Base Beacon Code

```
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <BluetoothSerial.h>
#define FIREBASE_HOST "esp22-31c31-default-rtdb.firebaseio.com"
```

```
#define FIREBASE_AUTH "a7cKOot2Fr2vqOiQOPibMmzd2cNCOs6CtBp1vKmz"
#define WIFI_SSID "Khalidd"
#define WIFI_PASSWORD "khaledsaad249"
// Define FirebaseESP32 data object
FirebaseData firebaseData;
// Define BluetoothSerial object
BluetoothSerial SerialBT;
bool bluetoothDataReceived = false;
void setup()
{
  Serial.begin(115200);
 SerialBT.begin("ESP32_BT");
  // Connect to Wi-Fi
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  // Connect to Firebase
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  Firebase.reconnectWiFi(true);
  // Wait for connection to be established
  while (!Firebase.ready())
  {
    delay(1000);
  }
  Serial.println("------------------------------------");
  Serial.println("Connected...");
}
void loop()
{
  String data = "";
  String nodeName="";
  String xValue="";
  String yValue="";
  String timer="";
 // Check if Bluetooth data has been received
  if (SerialBT.available() && !bluetoothDataReceived)
  {
    data = SerialBT.readString();
    Serial.println(data);
// Set the flag to indicate Bluetooth data has been received
bluetoothDataReceived = true;
    // Turn off Bluetooth
    SerialBT.end();
  }
  // Check if Bluetooth data has been received and turn off Bluetooth
  if (bluetoothDataReceived)
  {
    int firstCommaIndex  = data.indexOf(',');
    int secondCommaIndex = data.indexOf(',', firstCommaIndex + 1);
    int thirdCommaIndex  = data.indexOf(',', secondCommaIndex + 1);
    int fourthCommaIndex = data.indexOf(',', thirdCommaIndex+1);
    if (firstCommaIndex != -1 && secondCommaIndex != -1 && thirdCommaIndex != -1&&fourthCommaIndex==-1)
    {
      nodeName = data.substring(0, firstCommaIndex);
      xValue = data.substring(firstCommaIndex + 1, secondCommaIndex);
      yValue = data.substring(secondCommaIndex + 1, thirdCommaIndex);
      timer = data.substring(thirdCommaIndex + 1);
      if (nodeName == "Abdalla")
      {       // Update x and y values in the existing node
        Firebase.setString(firebaseData, "/user1/mac", nodeName);
        Firebase.setString(firebaseData, "/user1/x", xValue);
        Firebase.setString(firebaseData, "/user1/y", yValue);
        Firebase.setString(firebaseData, "/user1/time", timer);  }
  else
  {   Firebase.setString(firebaseData, "/user2/mac", nodeName);
    Firebase.setString(firebaseData, "/user2/x", xValue);
    Firebase.setString(firebaseData, "/user2/y", yValue);
    Firebase.setString(firebaseData, "/user2/time", timer);       }         }
ESP.restart();  {  delay(1000);      }
```

5.3 Desktop Application Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using FireSharp;
using FireSharp.Config;
using FireSharp.Interfaces;
using FireSharp.Response;
using Newtonsoft.Json;
using System.Data.SqlClient;
using System.Windows.Forms.DataVisualization.Charting;
using System.Threading;
namespace firebase
{
public partial class Form1 : Form
{
```

5.3.1 Setup firebase Connection

```
private readonly IFirebaseConfig con;
private readonly FirebaseClient client;
public String conString = "Data Source=.;Initial Catalog=position;Integrated Security=True";
public Form1()
{
InitializeComponent();
con = new FirebaseConfig
{
            AuthSecret = "a7cKOot2Fr2vqOiQOPibMmzd2cNCOs6CtBp1vKmz",
            BasePath = "https://esp22-31c31-default-rtdb.firebaseio.com/"
            //AuthSecret = "fldVBZMkSZ8sMIKFUy3YZDSUmERKFmWMhQWi4GDc",
            //BasePath = "https://first-database-88a8c-default-rtdb.firebaseio.com/"

};
client = new FirebaseClient(con);
}
```

5.3.2 Get number of users on firebase

```
public async Task<int> GetChildCount(string nodeName)
{
var data = await client.GetTaskAsync($"/{nodeName}");
if (data == null)
{
            return 0;
}
else
{
            dynamic obj = Newtonsoft.Json.JsonConvert.DeserializeObject(data.Body);
            Dictionary<string, object> dict = obj.ToObject<Dictionary<string, object>>();
            return dict.Count;
}             }
```

5.3.3 Collect the data from firebase then draw the map and store the data to sql data base

```
public async void drawPointsAndSaveToDatabase()
{
try
{
            int count = await GetChildCount("/");
            string timecom = "";
            for (int i = 0; i < count; i++)
            {
            int j = i + 1;
            string t = "user" + j;
            var rs = await client.GetTaskAsync("user" + j + "");
            Data result = rs.ResultAs<Data>();
            string mac = result.mac;
            string time = result.time;
            double x = result.x;
            double y = result.y;
            if (TableExists(t) == 0)
            {
                        createDatabase(t);
                        string query = $"INSERT INTO {t} (mac, x, y, time1) VALUES ('{mac}', '{x}', '{y}', '{time}')";
                        // Create a connection to the database
                        using (SqlConnection connection = new SqlConnection(conString))
                        {
                        // Create a command object with the query and connection
                        using (SqlCommand command = new SqlCommand(query, connection))
                        {
                        // Open the connection
                        connection.Open();
                        // Execute the query
                        int result2 = command.ExecuteNonQuery();
                        }
                        }
                        chart1.Series["Users"].Points.AddXY(Convert.ToDouble(x), Convert.ToDouble(y));
                        chart1.Series["Users"].Points[i].Label = mac;
            }
            else if (TableExists(t) == 1)
            {
                        string query = $"INSERT INTO {t} (mac, x, y, time1) VALUES ('{mac}', '{x}', '{y}', '{time}')";
                        // Create a connection to the database
                        using (SqlConnection connection = new SqlConnection(conString))
                        {
                        // Create a command object with the query and connection
                        using (SqlCommand command = new SqlCommand(query, connection))
                        {
                        // Open the connection
                        connection.Open();
                        // Execute the query
                        int result2 = command.ExecuteNonQuery();
```

```
                    }     }
                    chart1.Series["Users"].Points.AddXY(Convert.ToDouble(x), Convert.ToDouble(y));
                    chart1.Series["Users"].Points[i].Label = mac;
            }
            timecom = time;
            }
            addUsersInComboBox();                       }
    catch (Exception ex)
    {              }                    }
```

5.3.4 Timer for refreshing the map

```
    private void timer1_Tick(object sender, EventArgs e)
    {  chart1.Series["Users"].Points.Clear();
            drawPointsAndSaveToDatabase();     }
```

5.3.5 Check if the user existed in database

```
    public int TableExists(string tableName)
    {
    // Connection string for the SQL Server database
    // Create a connection to the database
    using (SqlConnection connection = new SqlConnection(conString))
    {
            // Open the connection
            connection.Open();
            // Create a SQL query to check if the table exists
            string query = $"SELECT COUNT(*) FROM information_schema.tables WHERE table_name =
            '{tableName}'";
            // Create a command object with the query and connection
            using (SqlCommand command = new SqlCommand(query, connection))
            {
            // Execute the query and get the result
            int result = (int)command.ExecuteScalar();
            // Check if the result is greater than zero, which means the table exists
            if (result > 0)
            {
                    return 1;
            }
            else
            {
                    return 0;
            }   }   }   }
```

5.3.6 Create table in database for new users

```
    public void createDatabase(String table) {
    // SQL query to create the table
    string query = $"CREATE TABLE {table} (id INT IDENTITY(1, 1),mac VARCHAR(50) ,x VARCHAR(50),y
    VARCHAR(50),time1 VARCHAR(50))";
    // Create a connection to the database
    using (SqlConnection connection = new SqlConnection(conString))
    {
            // Open the connection
            connection.Open();
            // Create a command object with the query and connection
            using (SqlCommand command = new SqlCommand(query, connection))
            {
            // Execute the query
            command.ExecuteNonQuery();
            }
    }
    }
```

5.3.7 Loading the users in combobox for tracking their history

```
    public void addUsersInComboBox()
    {
    comboBox1.Items.Clear();
    comboBox3.Items.Clear();
    string query = "SELECT COUNT(*) FROM information_schema.tables WHERE table_type = 'BASE TABLE'";
    int tableCount;
    using (SqlConnection connection = new SqlConnection(conString))
    {
            connection.Open();
            using (SqlCommand command = new SqlCommand(query, connection))
            {
            tableCount = Convert.ToInt32(command.ExecuteScalar());
            }
connection.Close();   }
    for (int i = 0; i <tableCount ; i++)
    {
            int j = i + 1;
            string query1 = "SELECT mac FROM user" + j + "";
            using (SqlConnection connection = new SqlConnection(conString))
            {
            connection.Open();

            using (SqlCommand command = new SqlCommand(query1, connection))
            {
                    string name = (string)command.ExecuteScalar();
                    comboBox1.Items.Add(name);
                    comboBox3.Items.Add(name);
            }
            connection.Close();
            }
    }
    }
```

5.3.8 Adding all history of a specific user in a combobox to choose from them

```
    public void addTimesToComboBox()
    {
    if (comboBox1.SelectedIndex == -1)
    {
            MessageBox.Show("Please Choose The User !");
    }
    else {
    comboBox2.Items.Clear();
            string user = comboBox1.SelectedItem.ToString();
            string timeCompare = "";
            int j = 0;
            string query = "SELECT COUNT(*) FROM information_schema.tables WHERE table_type = 'BASE
            TABLE'";
            int tablesCount;
            int tableCount;
            using (SqlConnection connection = new SqlConnection(conString))
            {
                    connection.Open();

                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                    tablesCount = Convert.ToInt32(command.ExecuteScalar());
                    }
                    connection.Close();
            }
            for (int i = 0; i < tablesCount; i++)
            {
                    j = i + 1;
                    int y = i + 1;
                    string query1 = "SELECT mac FROM user" + j + "";
                    string name;
                    using (SqlConnection connection = new SqlConnection(conString))
                    {
                    connection.Open();
                    using (SqlCommand command = new SqlCommand(query1, connection))
                    {
                    name = (string)command.ExecuteScalar();
                    }
                    connection.Close();
                    }
                    if (user == name)
                    {
                    break;
                    }
            }
            string query2 = "SELECT COUNT(*) FROM user" + j + "";
            using (SqlConnection connection = new SqlConnection(conString))
            {
                    connection.Open();
                    using (SqlCommand command = new SqlCommand(query2, connection))
                    {
                    tableCount = Convert.ToInt32(command.ExecuteScalar());
                    }
                    connection.Close();
            }
            for (int p = 0; p < tableCount; p++)
            {
                    int c = p + 1;
                    string query4 = "SELECT time1 FROM user" + j + " WHERE id = " + c + "";
                    using (SqlConnection connection = new SqlConnection(conString))
                    {
                    connection.Open();
                    using (SqlCommand command = new SqlCommand(query4, connection))
                    {
                    string name = (string)command.ExecuteScalar();
                    if (name == timeCompare)
                    {
                    }
                    else
                    {
                    comboBox2.Items.Add(name);
                    timeCompare = name;
                    }
            }
                    connection.Close();
                    }
            }
    }

    }
```

Chapter 6 : Testing

6.1 Testing Approach

Criteria	Black Box Testing	White Box Testing
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	**White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.**
Levels Applicable To	Mainly applicable to higher levels of testing	**Mainly applicable to lower levels of testing**
Responsibility	Generally, independent Software Testers	**Generally, Software Developers**
Programming Knowledge	Not Required	**Required**
Implementation Knowledge	Not Required	**Required**
Basis for Test Cases	**Requirement Specifications**	**Detail Design**

6.1.1 Functional Testing

Functional Testing is a testing technique that is used to test the features/functionality of the

system or Software, should cover all the scenarios including failure paths and boundary cases.

Functional Testing Techniques:

1- **White Box Testing :**

White box testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

Advantages :

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Efficient in finding errors and problems
- Required knowledge of internals of the software under test is beneficial for thorough testing
- Allows finding hidden errors
- Programmers introspection
- Helps optimizing the code
- Due to required internal knowledge of the software, maximum coverage is obtained

Disadvantages

- Expensive as one has to spend both time and money to perform white box testing.
- In-depth knowledge about the programming language is necessary to perform white box testing.
- Requires high level knowledge of internals of the software under test
- Requires code access

2- Black Box Testing :

Black-box testing is a method of software testing that examines the functionality of an application based on the specifications. It is also known as Specifications based testing. Independent Testing Team usually performs this type of testing during the software testing life cycle. This method of test can be applied to each and every level of software testing such as unit, integration, system and acceptance testing.

Advantages

- Efficient when used on large systems.

- The tester and developer are independent of each other; testing is balanced and unprejudiced.
- Tester can be non-technical.
- There is no need for the tester to have detailed functional knowledge of system.
- Tests will be done from an end user's point of view, because the end user should accept the system. (This testing technique is sometimes also called Acceptance testing.)
- Testing helps to identify vagueness and contradictions in functional specifications.
- Test cases can be designed as soon as the functional specifications are complete.

Disadvantages

- Test cases are challenging to design without having clear functional specifications.
- It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
- It is difficult to identify all possible inputs in limited testing time. As a result, writing test cases may be slow and difficult.
- There are chances of having unidentified paths during the testing process.
- There is a high probability of repeating tests already performed by the programmer

6.1.2 Unit Testing

We wanted to test all pieces of the software components specially the basic components of the software to be sure that all the units of these components have been tested and they are working as expected. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Unit testing ensures that each component of the software is working correctly and as expected but this happens separately. There is no any integration among these components. The main aim is to isolate each unit of the system to identify, analyse and fix the defects. Technique used is White Box Testing.

6.1.3 Integration Testing

Integration testing aims to see how two or more components are going to work together to check if they are producing the expected output or not, it aims to exposes any defects in the interactions between the components, also to discover if any issues in the interface that gathers the integrated components. Progressively larger groups of tested software components corresponding to elements of architectural design are integrated and tested until the software works as a system. It also verifies that the system is integrated to any external or thirdparty systems defined in the system requirements. Upon completion of unit testing, the units or modules are to be integrated which gives raise to integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated. Integration Strategies used is Bottom up Integration .

Bottom up Integration :

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.

6.1.4 System Testing

We can consider that the system testing is a BIG integration testing after integrating all the components and the modules of the system together to verify that this system meets its requirements. System Testing (ST) is a black box testing technique performed to evaluate the complete system the system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective. System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and non-functional testing. Technique used is Black Box Testing

6.1.5 Acceptance Testing

Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is having met the required criteria for delivery to end users.

6.1.6 Interface Testing

Interface Testing is performed to evaluate whether systems or components pass data and control correctly to one another. It is to verify if all the interactions between these modules are working properly and errors are handled properly. Handle network failures between Web site and application server.

6.1.7 Regression Testing

Regression testing is a black box testing technique that consists of reexecuting those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

6.1.8 Non-Functional Testing

Non-Functional testing is a software testing technique that verifies the attributes of the system such as memory leaks, performance or robustness of the system. Non-Functional testing is performed at all test levels.

6.1.9 Performance Testing

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Performance Testing Techniques :

1- **Load testing**

It is the simplest form of testing conducted to understand the behavior of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.

2- Stress testing

It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected maximum.

Stress testing a Non-Functional testing technique that is performed as part of performance testing. During stress testing, the system is monitored after subjecting the system to overload to ensure that the system can sustain the stress. The recovery of the system from such phase (after stress) is very critical as it is highly likely to happen in production environment.

Reasons for conducting Stress Testing :

- It allows the test team to monitor system performance during failures.
- To verify if the system has saved the data before crashing or NOT.
- To verify if the system prints meaning error messages while crashing or did it print some random exceptions.
- To verify if unexpected failures do not cause security issues. Stress Testing - Scenarios
- Monitor the system behavior when maximum number of users logged in at the same time.
- All users performing the critical operations at the same time.
- All users accessing the same file at the same time.
- Hardware issues such as database server down or some of the servers in a server park crashed.

3- Soak testing

Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.

4- Spike testing

Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

5- Load Testing:

Load testing is performance testing technique using which the response of the system is measured under various load conditions. The load testing is performed for normal and peak load conditions.

Objectives of Load Testing:

- Response time
- Throughput
- Resource utilization
- Maximum user load

6- Endurance Testing

Endurance testing also known as Soak Testing is performed to determine if the application under test can sustain the continuous loads. Endurance testing, non-functional testing involves examining the system if it can withstand a huge load for a longer period of time and thereby measuring the system's reaction parameters.

7- Volume Testing

Volume testing is a Non-functional testing that is performed as part of performance testing where the software is subjected to a huge volume of data. It is also referred as flood testing.

Volume Testing Characteristics :

- During development phase, only small amount of data is tested.
- The performance of the software deteriorates over time as there is enormous amount of data overtime.
- Test cases are derived from design documents.
- Test data is usually generated using a test data generator.
- Test data need not be logically correct but the data is to assess the system performance.
- Upon completion of testing, results are logged and tracked to bring it to closure.

Volume Testing – Checklist :

- Verify if there is any data loss.
- Check the system's response time.
- Verify if the data is stored incorrectly.
- Check if the data is overwritten without any notification

8- Usability Testing

Usability testing is a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

- Level of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.
- The measure of increase in user productivity if any.
- Assessment of a user's attitude towards using the software.

9- Security Testing

Security testing is a testing technique to determine if an information system protects data and maintains functionality as intended. It also aims at verifying 6 basic principles as listed below:

- Confidentiality
- Integrity
- Authentication
- Authorization
- Availability
- Non-repudiation

10- Recovery Testing

Recovery testing is a type of non-functional testing technique performed in order to determine how quickly the system can recover after it has gone through system crash or hardware failure. Recovery testing is the forced failure of the software to verify if the recovery is successful.

11- Resilience Testing

Resilience testing confirms that the system can recover from expected or unexpected events without loss of data or functionality. Recovery testing confirms that the affected system can be restarted successfully after an outage.

12- Scalability Testing

Scalability is a performance testing parameter that investigates a system's ability to grow by increasing the workload per user, or the number of concurrent users, or the size of a database.

13- Compatibility Testing

Compatibility testing is a non-functional testing conducted on the application to evaluate the application's compatibility within different environments. It can be of two types - forward compatibility testing and backward compatibility testing.

6.1.10 Other Testing

1) **Database Testing**

Database testing involves the retrieved values from the database by the web or desktop application. Data in the User Interface should be matched as per the records are stored in the database.

2) **Thread Testing**

A thread is the smallest unit of work that a system can execute. Thread testing, a software testing technique used during early integration testing phase to verify the key functional capabilities that carry out specific task. These kinds of techniques are very helpful if an application is of type that uses client server architecture. Performing Thread testing on valid business transaction through the integrated client, server and network is very critical. Threads are integrated and tested incrementally as subsystems and then performed as a whole system.

3) **Concurrency Testing**

Concurrency testing is also known as multi-user testing, performed to identify the defects in an application when multiple users login to the application. It helps in identifying and measuring the problems in system parameters such as response time, throughput, locks/dead locks or any other issues associated with concurrency.

4) **Stability Testing**

Stability testing is a software testing technique adopted to verify if application can continuously perform well within or just above the acceptable period. It is a Non-functional Testing technique conducted as part of performance testing often referred as load or Endurance testing.

5) **Reliability Testing**

Software reliability testing is a testing technique that relates to testing the ability of software to function given environmental conditions consistently that helps uncover issues in the software design and functionality.

6) **Dependency Testing**

Dependency Testing, a testing technique in which an application's requirements are pre-examined for existing software, initial states in order to test the proper functionality. The impacted areas of the application are also tested when testing the new features or existing features.

7) **Portability Testing**

Portability testing is a process of testing with ease with which the software or product can be moved from one environment to another. It is measured in terms of maximum amount of effort required to transfer from one system to another system. The portability testing is performed regularly throughout the software development life cycle in an iterative and incremental manner.

8) **Test Automation**

Software Test automation makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. Usually, regression tests, which are repetitive actions, are automated. Testing Tools not only helps us to perform regression tests but also helps us to automate data set up generation, product installation, GUI interaction, defect logging, etc. Automation tools are used for both Functional and Non-Functional testing.

9) **Vulnerability Testing**

Vulnerability testing, a software testing technique performed to evaluate the quantum of risks involved in the system in order to reduce the probability of the event.

6.2 Testing Results

Hardware Used

- Arduino ESP32 board
- Bluetooth module
- 3 beacons

Method Used

- RSSI (Received Signal Strength Indication) and trilateration

Testing Environment

- A room with dimensions of 5m x 5m x 3m (L x W x H)
- Beacons were placed in fixed positions at three corners of the room
- The ESP32 board was carried around the room at different locations

Technology

Problem: Interference from other wireless signals in the environment affecting the accuracy of the RSSI values.

Success: Implementation of signal filtering techniques such as averaging or smoothing to reduce the impact of interference on the RSSI values, as well as use of directional antennas or frequency-hopping techniques to avoid interference.

Problem: Limited range of Bluetooth signals affecting the coverage area of the system.

Success: Use of multiple beacons to extend the coverage area, as well as implementation of signal boosting techniques such as power amplifiers or signal repeaters.

Manufacturing

Problem: Inconsistent performance of the hardware components affecting the accuracy and reliability of the system.

Success: Quality control measures such as testing and calibration of the hardware components before and after assembly, as well as use of high-quality components with low failure rates.

Problem: Difficulty in integrating the hardware and software components of the system.

Success: Collaboration between hardware and software engineers to ensure compatibility and seamless integration of the components, as well as use of standardized communication protocols and interfaces.

User Experience

Problem: Difficulty in using the Android application to calculate and display the position of the ESP32 board.

Success: User testing and feedback to improve the usability and user interface of the application, as well as providing clear instructions and tutorials for users.

Problem: Inaccurate or delayed position updates affecting the usability of the system.

Success: Optimization of the trilateration algorithm and Bluetooth signal processing to improve the accuracy and speed of position updates, as well as use of real-time data visualization to provide immediate feedback to users.

Overall, testing an indoor positioning system with Bluetooth using RSSI and trilateration methods, and using Arduino ESP32 as hardware and an Android application to calculate the position, requires careful consideration of various factors such as hardware performance, software integration, and user experience. However, with effective problem-solving strategies and collaboration between different stakeholders, these challenges can be overcome to create a successful and reliable system.

Chapter 7 : Conclusion

7.1 Summary of Achieved results

In conclusion, the indoor positioning system using Bluetooth, ESP32, and a mobile application with Kotlin programming language, and using RSSI and trilateration methods to calculate distance and position is a highly effective and accessible solution for tracking the position of objects or people in indoor environments. The following points summarize the benefits of this system:

- Accuracy and reliability: The use of RSSI and trilateration methods provides accurate distance and position calculation, making this system useful for a wide range of applications.
- User-friendliness: The Android application with Kotlin programming language provides real-time tracking of the receiver's position and an intuitive user interface, making it accessible to users of all technical backgrounds.
- Low cost: The use of low-cost hardware components such as the ESP32 board and Bluetooth module makes this system affordable for individuals and organizations with limited budgets.

This project demonstrates the potential for innovative technology solutions to improve our daily lives. By using open-source software and low-cost hardware components, this system can be implemented in various indoor environments to track the position of objects or people. Its success is attributed to its accuracy, reliability, user-friendliness, and affordability. In conclusion, the indoor positioning system using Bluetooth, ESP32, and a mobile application with Kotlin programming language, and using RSSI and trilateration methods to calculate distance and position is a valuable solution for indoor positioning that can benefit many industries and individuals.

Why choose Bluetooth IPS among the 8 indoor positioning system?

Positioning system has penetrated into all aspects of our life, among which satellite navigation and geographic data technology is the most mature positioning technology. However, these technologies are more suitable for use in the open outdoor environment due to the block of indoor building materials, which results in poor signal, that’s why the indoor positioning system emerged. There has been a variety of indoor positioning systems, each of which has its own suitable application industry and Bluetooth is used in a variety of industries and is considered one of the futures of IoT IPS systems. In this article, you will get to learn what IPS is, what kinds of the indoor positioning system is commonly used, why you should choose Bluetooth IPS, and how you can implement them.

Indoor positioning system refers to the equipment network that can locate people or objects in the indoor environment. It usually consists of two distinct elements: anchor and position tags. Anchor is a device, such as a beacon or a relay, strategically placed in a defined space. Tags are carried by people or things. Anchor either actively locates the mobile device and tag, or provides an environment location or context for the device to be aware of.

The advantages of Bluetooth include the following:

- It removes the problem of radio interference by using a technique called Speed Frequency Hopping. This technique utilizes 79 channels of the particular frequency band, with each device accessing the channel for only 625 microseconds, i.e. the device must toggle between transmitting and receiving data from one-time slot to another.
- This implies the transmitters change frequencies 1,600 times every second, meaning that more devices can make full use of a limited slice of the radio spectrum. This ensures that the interference won't take place as each transmitter will be on different frequencies.
- The power consumption of the chip (consisting of a transceiver) is low, at about 0.3mW, which makes it possible for the least utilization of battery life.
- It guarantees security at the bit level. The authentication is controlled using a 128- bit key.
- It is possible to use Bluetooth for both transferring data and verbal communication as

Bluetooth can support data channels of up to 3 similar voice channels.

- It overcomes the constraints of the line of sight and one-to-one communication as in other modes of wireless communication like infrared.

Hardware Used in our Project :

Arduino Nano:

- It is a microcontroller board based on the ATmega328P chip.
- It has 14 digital input/output pins, 6 analog inputs, a 16 MHz quartz crystal, a USB connection, and a power jack.
- It can be programmed using the Arduino IDE, which is a free software development environment based on C++.

ESP32:

- It is a low-cost, low-power system on a chip (SoC) microcontroller with built-in Wi-Fi and Bluetooth capabilities.
- It has 34 GPIO pins, 18 analog inputs, and supports various communication protocols such as SPI, I2C, and UART.
- It can be programmed using the Arduino IDE or the ESP-IDF (Espressif IoT Development Framework).

Bluetooth Module HM-10 :

- It is a small, low-cost Bluetooth module that provides wireless communication between devices.
- It supports the Bluetooth 2.0 specification and can be used in both master and slave modes.
- It can be easily integrated with microcontrollers such as Arduino Uno and ESP32 to enable wireless communication between devices.

There is Compare Between Our project and Similar Project :

Project	Estimote Indoor Location Tech.	Aeroscout Wi-Fi Positioning	Nanotron UWB Positioning	Our Project
Technology	BLE	Wi-Fi	UWB	Bluetooth
Accuracy	High	Medium	High	High
Range	Short.	Medium	Short	Medium
Cost	Medium Beacons around 40$	High Access Points are expensive	High Anchors are expensive	Low Under 30 $
Power Consumption	Low beacons can last up to 5 years on a single battery	High access points require power and network connectivity	Low anchors can last several years on a single battery	Low it can run with adapter 5 V
Infrastructure Required	Beacons (can be easily installed	Access Points (require power and network connectivity	Anchors (require installation and calibration)	Beacons (ESP 32 Arduino)
Advantages	• Easy to install • low power consumption • high accuracy • good for small areas • can work with existing mobile devices.	• Can cover larger areas than BLE • Can work with existing Wi-Fi infrastructure.	• High accuracy, good for tracking moving objects • Can work in harsh environments	• Low cost, easy setup • low power consumption, • no additional infrastructure required
Disadvantage	• Limited range compared to Wi-Fi and UWB, may require a large number of beacons for larger areas	• Higher cost than BLE, may require additional infrastructure for larger areas.	• Higher cost than BLE and Wi-Fi, requires installation and calibration of anchors.	• Limited Range • Limited Accuracy

References

1. "Indoor Positioning and Indoor Navigation" by João Barros and Nuno Borges Carvalho (2017) - This book covers the fundamentals of indoor positioning and navigation, including signal propagation models, localization algorithms, and system design considerations.
2. "Indoor Positioning Technologies" by Jari Nurmi and Elena Simona Lohan (2019) - This book provides an overview of various indoor positioning technologies, including Bluetooth, Wi-Fi, UWB, and RFID.
3. "Indoor Positioning Using Bluetooth Low Energy Beacons" by Luis M. Bergasa and Arturo de la Escalera (2018) - This paper provides an overview of Bluetooth Low Energy (BLE) beacons and their use in indoor localization systems.
4. "Indoor Localization Using Wi-Fi Fingerprinting: A Survey" by Mohammad Mehedi Hassan and Muhammad Mahbubur Rahman (2019) - This survey paper provides an overview of Wi-Fi fingerprinting-based indoor positioning systems, including their advantages and limitations.
5. "Ultra-Wideband Indoor Positioning Systems: A Survey" by Y. Gu, J. Zhang, and Z. Zhu (2018) - This survey paper provides an overview of ultra-wideband (UWB) indoor positioning systems, including their advantages and challenges.
6. "RFID-Based Indoor Positioning Systems: A Survey" by Mohammad Mehedi Hassan and Muhammad Mahbubur Rahman (2018) - This survey paper provides an overview of RFID-based indoor positioning systems, including their advantages and limitations.
7. "Visual-Based Indoor Positioning Systems: A Survey" by Zhenyu Chen, Wei Wang, and Xiaohua Jia (2018) - This survey paper provides an overview of visual-based indoor positioning systems, including their advantages and limitations.
8. IndoorAtlas website (https://www.indooratlas.com/) - IndoorAtlas is a company that specializes in indoor positioning technology, offering a software development kit (SDK) for integrating indoor positioning into mobile applications.
9. Estimote website (https://estimote.com/) - Estimote is a company that offers Bluetooth beacons and an SDK for building indoor positioning and proximity applications.
10. Navisens website (https://navisens.com/) - Navisens is a company that offers a motionDNA platform for indoor positioning and navigation using sensors in mobile devices.