8-23-2023 2:30 PM

Smartphone Loss Prevention System Using BLE and GPS Technology

Noshin Tasnim, *The University of Western Ontario*

Supervisor: Haque, Anwar, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science
© Noshin Tasnim 2023

Recommended Citation

Tasnim, Noshin, "Smartphone Loss Prevention System Using BLE and GPS Technology" (2023). *Electronic Thesis and Dissertation Repository*. 9586.
https://ir.lib.uwo.ca/etd/9586

Abstract

Being an all-in-one gadget, smartphones play a vital role in our everyday lives. However, millions of people suffer every year by losing their phones. A lost phone creates a huge security threat and data loss possibility to the users. Some preventive measures are available to protect from unauthorized access. Moreover, there are some post-loss solutions to track down, retrieve data from a lost locked phone, and protect the privacy and security of lost phone data, but those have some drawbacks as well. Considering the situation, our proposed system offers a preventive solution which will protect the smartphones from getting lost. Our system involves a smartwatch which will be connected to smartphone via Bluetooth Low Energy (BLE) and keep track of the distance between the smartphone and the smartwatch worn by user in real-time. The system will be able to identify if the distance goes beyond 20 feet or a customizable distance given by the user and immediately raise an alert in the smartwatch, creating vibration and sound in public places. The system allows users to mark their safe location (e.g., house, office) and radius where their smartphone will be safe, and they don't need alerts. We have developed 3 different models to implement this system with different approaches using Ranging, Haversine formula and Geofencing. For our work, we aim to perceive how accurate our models are in terms of calculating distance and safe location tracking as well as alert response time and models impact on battery life of both smartphone and smartwatch. We have developed an Android application and a smartwatch application that run on both smartwatch and smartphone for each model and compared their performances based on our evaluation parameters. We conducted experiments under various real-world conditions and the system incorporated with Model 1 can generate alert with 96% accuracy when user is away from the smartphone beyond the threshold distance in an unsafe location. This affordable solution will ensure prevention from smartphones getting lost in public places in an effective way securing confidentiality and data protection to users.

Keywords

Smartwatch, Bluetooth Low Energy, RSSI, GPS, Geofencing, Smartphone, Android application.

Summary for Lay Audience

With the advancement of technology, nowadays, people are highly dependent on their smartphone. From personal information to confidential information, people store most of the important and essential information on their smartphones. However, people tend to lose their smartphones being inattentive and leaving their smartphones behind in a public place. Losing their smartphone creates a big financial loss as well as data loss. If there existed a device that could provide alert to the user in case they unintentionally left their phone in a public location, it would prevent the user from losing their smartphones. In this study, we began by analyzing various existing solutions developed by other researchers for smartphone loss prevention systems which provides alert to the users in case they leave their phone. By examining existing solutions, we identify potential research gaps in existing smartphone loss prevention systems and addressing those research gaps implemented a wearable solution for smartphone loss prevention system. We present a system that provides alert to the user when the user goes beyond a pre-set distance threshold (set by the user) from the user's smartphone. In addition, the system doesn't provide the alert when the user is in a safe location saved by the user, rather provide alert in unsafe locations. We implemented the system on smartwatch and an Android User Interface using three different models applying different technologies and techniques to provide solution to the users. We have tested all three of the solutions under various real-world conditions to determine which solution provide more accurate and efficient results. In our testing, we demonstrated that the system could successfully provide alert to the users in case they leave their phone behind in an unsafe location.

Acknowledgments

I would like to thank my supervisor, Dr. Anwar Haque, for his guidance and inspiration for new ideas. I would also like to thank Dr. Haque his continual involvement in the progress of our work as well as for his suggestions and new insights as the study progressed.

Also, special mention to Moinul Islam Syed, who provided me with insights in traditional GPS and RTK GPS.

Finally, I would like to thank my family and friends who have supported me throughout the entire journey of my graduate studies.

Table of Contents

List of Tables

List of Figures

Chapter 1

1 Introduction

If we take a close look at our daily lives, we’ll see how much we are dependent on our smartphones. Smartphones have become an essential part of daily life, changing the way people communicate, work, and entertain themselves. From connecting people from all over the world to accessing a wealth of knowledge, smartphones have revolutionized our society. With the emerging development and advancement of technology, new features are always being added to smartphones which are making our lives easier and more dependent on smartphones. With the help of the internet, smartphones can access a huge area of information ranging from news, weather, traffic reports, and much more. People can also use their smartphones to access online courses, tutorials, and other educational resources. Additionally, smartphones have a range of apps that contain information on diverse topics such as healthcare, finance, travel, and much more. Furthermore, smartphones also have the capability of storing information such as personal contacts, notes, and photos, which people can access at any time.

A smartphone not only has our personal information but also contains confidential information, documents, and personal history. Confidential information can include sensitive data such as bank account details, credit card information, login credentials, personal data, social security numbers, dates of birth, and other sensitive information. If unauthorized parties access this information, it can lead to various consequences. One of the potential impacts is financial loss. Hackers who gain access to confidential financial information can steal money or commit fraudulent transactions, leading to significant financial losses for the victim. Additionally, if personal information such as social security numbers or dates of birth are compromised, it can lead to identity theft, which can result in unauthorized access to financial accounts, loans, and other financial activities, leading to significant financial and legal issues. Another potential impact of confidential information being compromised on a smartphone is reputation damage. If personal emails, text messages, or photos are leaked or accessed, it can damage an individual's reputation and personal relationships. In some cases, unauthorized access to confidential information can lead to business disruption. If confidential information is compromised in a business setting, it can lead to significant disruption to operations, damage to the company's reputation, and potential legal consequences.

As a result, when a smartphone is lost, it creates a severe security risk as well as financial and data loss. Although smartphones have their own security system to be protected from unauthorized access, confidential information on smartphones can still be retrieved in several ways bypassing the security. Some of the known ways are “Jailbreaking”, “Establishing”, “Setting up”. SIM swapping is also a technique used by hackers to gain control of a lost smartphone by tricking the mobile service provider into transferring the SIM card to a different device [1]. According to a study released from Kensington, 70 million smartphones are lost each year, with only 7 percent recovered [2]. This raises an alarming fact that in most cases, the device can’t be recovered which creates a big security issue and a chance of data loss.

In general, people tend to lose their cellphones in public places like supermarkets, restaurants, public transport, mall etc. In 2020, Prey Inc. announced the result of the second edition of the Mobile Theft & Loss Report where they stated 67% of mobile losses occur in interior locations and 33% when users are in transit or movement [3]. This allows strangers to steal the device and creates privacy and security threats.

There are several solutions to recover data from lost phones. Syncing with Google drive, iCloud, CM Cloud etc. helps the users to store their phone data in real time in cloud. However, the device must be connected with internet to upload their data in cloud. This allows the users to recover their data in case their phone gets lost. Also, some solutions can remotely lock and wipe out data from a lost phone. This solution helps users to protect their data privacy. Moreover, “Find my device” option can also help to remotely track the current location of the lost device and recover the device. However, these solutions have some drawbacks too. These solutions require the lost device to be connected with internet or a network or the device has to be switched on. In most cases, lost phones are often found by strangers, and they switched off the phone. By the time the user realizes that he/she has lost the phone, users don’t have the option to secure their data or recover their device.

However, all these solutions are for post-loss situations. Once a phone is lost, the chance of recovery decreases with time. What if we can have a solution which will prevent the device from getting lost in the first place?

In a study [3], it has been published that 69.12% of phones are lost because of misplacement and other 30.88% are because of all kinds of theft. This gives us a view that most of the time, we lose our phone being inattentive and leaving the phone behind in public places. If there was a way to give users an alert in case they leave their phone behind in a public place, it would prevent them from losing their phone.

In our work, we have implemented a system which provides alert when user is away from the smartphone beyond a preferred distance. We have implemented 3 different models to incorporate this system. For our work, our core research goals are as follows:

- How accurate are our models in determining the distance between the smartphone and the smartwatch and providing alert accordingly.
- The impact of our models on smartwatch and smartphone's battery life.
- How fast is our models to response and provide alert to the user.
- The performance of the same algorithmic method in different modules (smartphone and smartwatch) in terms of accuracy, reliability, and efficiency.

1.1 Thesis Contribution

To incorporate with these issues, this thesis aims to contribute to the following aspects-

- Our system is incorporated in wearable device which is physically connected to our body. Now a days, fitness bands and smart watches are very trendy and useful among users. According to the Consumer Watch Report recently released by the NPD Group and CivicScience [4], 7 percent of consumers now own a smartwatch, and 17 percent own fitness trackers. Fitness trackers and smartwatches have gained popularity because they not only allow users to check texts, emails and social media but also help us with fitness, health workout tracking, music control and many more features. One of the major reasons for choosing fitness band/

smartwatch for our solution is that these wearable devices always stay in touch with our skin and use vibration for notifications and alerts as well as sound. Also, as this system will be incorporating in fitness wearable device, users don't have to carry an additional device. Moreover, fitness bands can be purchased at an exceptionally low cost (starting from $15). Therefore, for our solution, a fitness band or smartwatch will be able to provide prompt caution to the user that the user will not miss.

- Our solution can detect the distance between the smartphone and the smart watch which will be worn by the user in real time. Whenever the distance goes over 20 feet or a customizable preferred distance given by the user, the smartwatch generates an alert creating vibration and sound. For the connectivity between the smartphone and fitness band, we have used Bluetooth module as fitness bands and smartphones have built in Bluetooth module.
- Although our system aims to provide real-time alert in case our smartphone gets absent from our 20 feet radius distance or user's customized distance, nonetheless, this will only be suitable for public places. If it starts to give alert when we move away from our smartphone in our safe places like home, office etc. it will be rather annoying to users. Moreover, as our main goal of this system is to prevent our phone from getting lost, there is no risk of losing our smartphones in our safe places. And so, our system provides an option to users to set up their safe places and their radius. Our system will stop providing alerts on those selected radius areas for the users if they move away from their smartphones. For implementing location tracking, we have used the GPS module. When user will set their safe location and provide the radius, GPS module will spot the location and mark the area within the given radius amount to mark it as user's safe location.
- Our system has an Android mobile application through which the user will be able to set their preferred parameters such as-
    1. Distance: The maximum distance between the phone and smartwatch after which the system will generate an alert.
    2. Location of Safe places: User will be able to set their safe places location and radius of the safe places.

- For the real-time distance measurement, we have implemented three separate models -
    (1) Our Model 1 uses GPS of the smartphone to keep track of user's presence in safe locations and the ranging method formula to calculate the distance between two smartphone and smartwatch. This formula uses the Received Signal Strength Indicator (RSSI) of the receiving device to measure the distance [5].
    (2) To keep track of user's presence in safe locations and to calculate the distance between the smartphone and the smartwatch, our Model 2 uses the Haversine method. The Haversine method has a formula which calculates the distance between two geographical coordinates. For our system, we have used current GPS coordinates of the smartwatch as one coordinate point and current GPS coordinates of the smartphone as the other coordinate point and applied the Haversine formula to calculate the distance between the smartwatch and the smartphone in real time.
    (3) Model 3 uses Haversine formula to keep track of user's presence in safe locations and the ranging method formula to calculate the distance between smartphone and smartwatch.
- We conducted experiments to demonstrate the ability of our models to operate under different conditions (including device orientations and positions), and covered various locations, and diverse layouts (closed and open spaces) to compare their accuracy, efficiency, and impact on battery life of smartwatch and smartphone.
- We have achieved an accuracy of 96% accuracy in generating alert when the distance exceeds a predefined threshold using the Model 1 within 0.47 seconds response time. And it also can successfully detect user's presence in safe locations with 90.66 % accuracy.

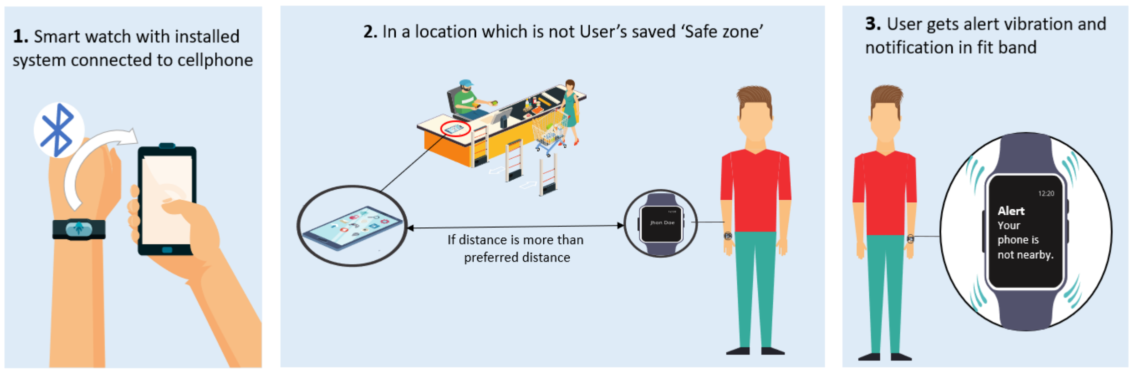

1. Smart watch with installed system connected to cellphone
2. In a location which is not User's saved 'Safe zone'
If distance is more than preferred distance
Jhon Doe
3. User gets alert vibration and notification in fit band
12:20
Alert
Your phone is not nearby.

Figure 1: Proposed System

Our proposed system, "Smartphone Loss Prevention System" is a smart solution to prevent the users from losing their smartphone. This solution aims to protect users from unauthorized access to their private data in case of phone loss, data loss, financial loss and mental disturbance. Furthermore, it offers a very cost-effective, affordable solution which can be accessible to all people. In this digitalized era, technology is improving everyday which is making us more dependent on electronic gadgets, especially on our smartphones. People carry their private documents, personal information as well as corporate information. Therefore, it has become very important to protect this important piece of gadget from being lost in our day-to-day life. Our system offers to provide this protection in a very effective way.

1.2 Thesis Outline

The rest of this thesis is organized as follows. Chapter 2 provides background information regarding smartwatch, smartwatch's features and implemented methods that we have used, and the applications of android app. Chapter 3 provides a literature review of the approaches for smartphone loss prevention systems and the application of such approaches. In the fourth chapter, we proposed a smartphone loss prevention system which provides an alert system when user is away from the smartphone beyond a predefined threshold in unsafe location. We also talked about our 3 developed Models which will be used for the paper. In Chapter 5, we use various real-world evaluation tests to demonstrate the functionality of our design. In Chapter 6, we conclude our work and make recommendations for future work to improve and expand the system.

Chapter 2

2 Background

This chapter will provide background information about the smartwatch that we have used in our work as well as the features of the smartwatch (Bluetooth low energy, RSSI, GPS) that has been used along with the implemented methods (Ranging method, Geofencing). Then, the applications of android app is discussed.

2.1 Smartwatch

With the improving development technology, it has given us a wide range of smart gadgets and wearables to make our life easier and smoother. Smartwatch is one of them. Nowadays, these smartwatches can perform a lot more functions than just providing time and date. Therefore, the demand for smartwatches is increasing day by day. About 18% of men wear smartwatches and 25% of women wear smartwatches which is increasing the global revenue generated by the smartwatch industry [6]. Over the years, usage of smartwatches has increased significantly [Table 1]. The penetration rate in the year 2026 is expected to be 2.93% [6].

Table 1 : The number of smartwatch users over the years [6]

Year	Number of Smartwatch User
2017	93.36 million
2018	122.12 million
2019	148.74 million
2020	186.89 million
2021	202.58 million
2022	216.43 million

Smartwatches typically connect to a smartphone or other mobile device via Bluetooth or another wireless protocol. They may also include their own cellular connectivity for independent use without a paired device. Smartwatches can run a variety of apps and may be compatible with various operating systems such as Android, iOS, or proprietary software. The most important usage of smartwatch is that it works as a fitness tracker. It helps people to track every health detail of their daily life such as the number of steps taken, as well as number of calories burnt throughout the day. It also provides vital information about the heart rate, blood pressure and oxygen levels. Moreover, it monitors our sleep cycle. These fitness features help people to keep a tab on their health and notice any abnormality promptly and take necessary steps accordingly.

Moreover, we can also receive and manage our calls and messages in smartwatch without carrying phone in our hands. Smartwatches can control music playback on a paired smartphone or other device. An important feature is that smartwatches can display notifications and alerts which we have used in our work.

The smartwatch market has vastly grown in recent years. There are many vendors available in the market providing different qualities of smartwatch. Most popular vendors are Apple, Samsung, Huawei, Imoo etc. These vendors sell ready to go products on the market. However, for our implementation purpose, we required an open-source smartwatch. There are some open-source smartwatches available on market such as PineTime Smartwatch by Pine64, Watchy by Squarofumi, OpenHAK, Bangle.js, Bangle.js 2. Although all of the mentioned open-source smartwatches provide BLE service, only Bangle.js 2 contains its own GPS module which is an important element for our implementation. Therefore, we have used Bangle.js 2 smartwatch in our work.

2.1.1 Bangle.js 2

Bangle.js 2 is an open-source, hackable smartwatch which has been developed by Espruino. The first model build by Espruino was Bangle.js. This new model Bangle.js 2 is an upgraded version of the previous one with improved features. The programming language of this smartwatch is JavaScript. Bangle.js is water resistant, AI enabled, and comes with Bluetooth Low Energy, GPS, a heart rate monitor, accelerometer and more [7].

Some of the key features of Bangle.js 2 are [7] –

- 1.3-inch 176x176 always-on 3-bit color LCD full touchscreen display.
- 8MB of flash storage and 256kB of RAM.

- Nordic 64MHz nRF52840 ARM Cortex-M4 processor with Bluetooth LE.
- 200mAh battery, 4 weeks standby time.
- IP67 Water Resistant
- A range of sensors, including an accelerometer, gyroscope, air pressure sensor, temperature sensor, magnetometer, heart rate monitor, GPS, and vibration motor.
- Bluetooth connectivity, enabling it to pair with a smartphone or other devices.
- A range of apps that can be downloaded and installed, including fitness trackers, weather apps, and more.
- Option for developing own app as well modify the firmware and build custom hardware add-ons according to user's need.

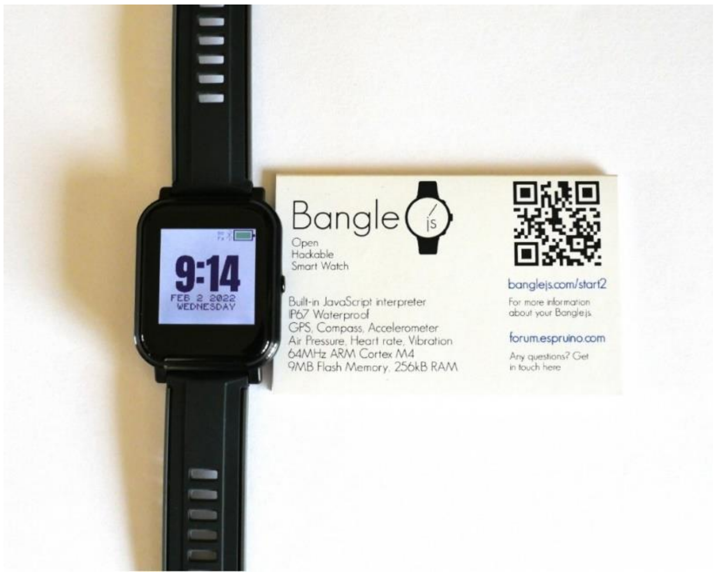

9:14
FEB 2 2022
WEDNESDAY
Bangle js
Open
Hackable
Smart Watch
Built-in JavaScript interpreter
IP67 Waterproof
GPS, Compass, Accelerometer
Air Pressure, Heart rate, Vibration
64MHz ARM Cortex M4
9MB Flash Memory, 256kB RAM
banglejs.com/start2
For more information about your Bangle.js
forum.espruino.com
Any questions? Get in touch here

Figure 2: Bangle.js 2 [8]

Overall, Bangle.js 2 is a powerful and flexible smartwatch that offers a lot of potential for customization and personalization, making it a popular choice among tech enthusiasts and developers.

2.2 Bluetooth Low Energy (BLE)

Wireless communication technology has transformed the way we interact with our devices and has enabled a new generation of smart and connected devices. One of the most widely used wireless communication technologies is Bluetooth, which enables devices to communicate with each other over short distances. However, Bluetooth consumes a good amount of battery power which is a big drawback. Therefore, traditional Bluetooth is not suitable for limited power devices such as wearables, sensors, and other IoT devices. To address this issue, Bluetooth Low Energy (BLE) was developed. BLE provides prominent features of Bluetooth but with low power consumption. BLE operates in the same 2.4 GHz frequency band as traditional Bluetooth [9]. Radio wave bands are used to make communications between devices by both Bluetooth and BLE [10]. However, during transmission, BLE devices remain asleep which helps it with low power consumption as little as 0.01 to 0.5 watt [10][11]. Following this power conservation technique, BLE devices only communicate and transfer data when it is necessary rather than having a long uninterrupted connection.

BLE follows an asymmetry architecture where one device will work as central role, and another will work as peripheral role [12]. The complex and advanced device such as computer, smartphone works in central role and limited power device such as sensors, IoT device works in peripheral role [12]. BLE protocol stack consists of 3 layers - the application layer provides the interface between the Bluetooth stack and the application, the host layer serves the stack of Bluetooth and physical layer handles the data transferring and receiving part. Although BLE devices have limited power and resources, they have some security mechanisms that they follow to have protection against attack. To ensure the confidentiality of the transmitting over network, encryption mechanism is used in BLE devices. Moreover, to have protection against unauthorized access, authentication and authorization mechanism is also used.

BLE has two main types of mode- Connected mode and Advertising-discovery mode. When both sending and receiving data is necessary, it works in connected mode which allows bi-directional data transfer [13]. In advertising-discovery mode, the device is not able to receive any data, it can only transmit data [13].

2.2.1 Nordic nRF52840

There are diverse types of system-on-chip (SoC) available for providing Bluetooth low energy connectivity to devices. Nordic nRF52840 processor has been used in the smartwatch that we are using for our work, Bangle.js 2. The Nordic nRF52840 is a powerful SoC from nRF52 series of chips. This is widely popular for its usage in BLE application in low powered wireless products such as sensor, wearables and IoT devices. It supports the latest Bluetooth 5 specification covering features such as higher data throughput, longer range, increased broadcasting capacity etc. [14].

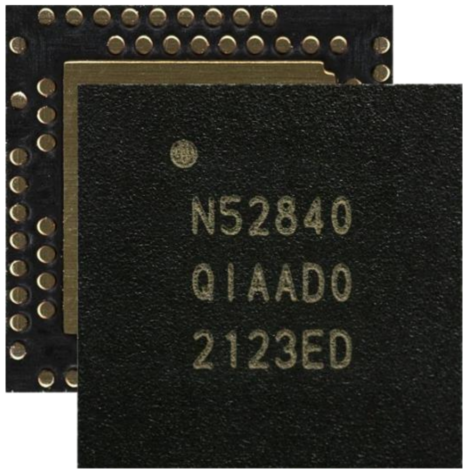


Figure 3: Nordic nRF52840 [15]

One of the good features of this SoC is that it has been designed for low power consumption which makes it perfect for low-powered devices. Through this BLE application of Nordic nRF52840, we have used RSSI from its BLE stack regarding measuring distance between smartwatch and smartphone.

2.2.2 Received Signal Strength Indicator (RSSI)

RSSI is the measurement of the signal strength of wireless connection such as Bluetooth or Wi-Fi. It is measured in decibel-milliwatts (dBm). In the dBm scale, this reference point is represented as 0 dBm [16]. Signals stronger than 1 mW are represented as positive dBm values and signals weaker than 1 mW are represented as negative dBm values. For real-world scenarios, we don't have 0 dBm signal strength. 0 dBm represents a 100% reference point, but it doesn't imply that the signal is always at this level. Instead, signals are usually weaker (negative dBm values) [17]. A signal strength depends on distance and broadcasting power value [18]. The broadcasting power value stays around 2-4 dBm which keeps the RSSI strength around -26 dBm (a few inches) to -100 dBm (40m - 50m) [18]. Stronger signal is represented by higher RSSI value and weaker signal is represented by lower RSSI value. This value can be used in several ways such as in a wireless connection, with the help of the RSSI value, it is possible to determine the distance between the connected devices. Furthermore, it can be used to get the location of a device in some cases. However, due to absorption, diffraction, interference, and other external factors, RSSI value’s accuracy sometimes fluctuates.

There are several methods available to measure the distance between smartphone and smartwatch in a Bluetooth connection using RSSI such as Signal strength fingerprinting, Triangulation and Ranging. However, signal strength fingerprinting requires pre-determined database locations and also, can only provide the estimated distance with an accuracy of 3m [19]. For using Bluetooth triangulation method, it requires 3 Bluetooth devices which makes it not feasible for our work. For this work, we have implemented the Ranging method to get the distance between smartphone and smartwatch using the RSSI value.

2.2.3 Ranging Method

Among all the methods, the ranging method is more efficient in terms of implementation. It needs two Bluetooth devices to calculate the distance. One device works as a locator, and another works as a target that needs to be found. There are two methods which can be followed to measure the distance.

The first method calculates the time it takes to reach the Bluetooth signal from one device to another device [20]. It is called a time of flight. From that time value, it calculates the distance between the devices.

The second method uses RSSI to measure distance where the RSSI value will be of the receiving device. It follows the following formula to compute the distance –

$$Distance = 10^{\left(\frac{Measured\ Power - RSSI}{10 \times N}\right)} [18]$$

Here,

Measured power is a factory-calibrated, read-only constant value which indicates the expected RSSI at a distance of 1 meter [18]. It is also known as 1-meter RSSI [18].

RSSI is obtained from the SoC's Bluetooth stack.

N is constant which stands for the environmental factor [18]. It describes the rate at which the signal strength decreases with distance. It ranges between 2-4 where low value represents low strength and high value represents high strength. For an indoor environment, it is always recommended to keep at a value of 2 as indoor environment has more noise and interference than open space which affects the signal strength.

2.2.4 Haversine Method

There are several formulas to calculate the distance between two points. Those can be applied while calculating distance between two geographical coordinates. However, those formulas work considering earth flat for shorter distances. But these formulas begin to provide wrong result if the distance is more than 20 kilometers as a result of considering the earth flat [21]. This issue can be overcome by considering earth as sphere using spherical geometry. Therefore, the Haversine formula is used to calculate the distance between two points using the latitude and longitude of the two points with the help of spherical geometry. The Haversine formula assumes that the Earth is a perfect sphere, with a radius of approximately 6,371 kilometers or 3,959 miles[22] .

The formula for calculating distance between two point (Latitude 1, Longitude 2) and (Latitude 2, Longitude 2) can be expressed as:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \ .\cos\varphi_2 \ .\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2\,.\,\text{atan2}\left(\sqrt{a}\,, \sqrt{(1-a)}\right)$$

$$d = R\,.\,c$$

Where,

d is the distance between the two points in the same unit as the radius,

R is the radius of the sphere,

φ is latitude in radians where φ_1 is Latitude 1 and φ_2 is Latitude 2,

λ is longitude in radians where λ_1 is Longitude 1 and λ_2 is Longitude 2,

$\Delta\varphi$ is the subtraction of Latitude 2 and Latitude 1,

And $\Delta\lambda$ is the subtraction of Longitude 2 and Longitude 1.

This formula is commonly used in applications regarding navigation and geolocation assuming that the Earth is a perfect sphere. Though in reality, the Earth is an oblate spheroid, which means that it is slightly flattened at the poles and bulges at the equator. Despite this, the Haversine formula still provides a good estimate of the distance between two points on the Earth's surface.

2.3 Global Positioning System (GPS)

Global Positioning System is a satellite-based navigation system which has been developed and launched by the United States Department of Defense in the 1978 [23]. GPS system provides users with services like navigation, timing, and positioning.

This system consists of three segments- the space segment, the control segment, and the user segment [24]. The space segment consists of a constellation of 24 satellites, which orbit around the earth twice a day in a specific pattern transmitting signals to users on geographical position and time of day [24]. The control segment is responsible for tracking, monitoring, managing, and maintaining the satellite constellation with its 3 components -

Earth-based monitor stations, master control stations and ground antenna [25]. The user segment consists of GPS receivers that receive signals from the satellites and use the transmitted information to determine the user's 3-D location and time. GPS positioning works on basic mathematical concept named Trilateration [26]. GPS receiver device receives information from at least three satellites and uses that to calculate the location. As one satellite receives only the distance information from the GPS receiver, it is not able to provide accurate information about the GPS receiver’s location. Therefore, three satellites information is needed to calculate the location of the user. When a satellite sends a signal, it creates a circle with a radius measured from the GPS device to the satellite [27]. Following that, adding a second satellite will create a second circle which will create an intersection area which will narrow down the possible location area of GPS receiver. Finally, the circle created by the third satellite will indicate the possible location of the GPS receiver in the two points produces the by the intersection of three circles [Figure 4]. This will give the 2-dimensional location of the GPS receiver. Fourth satellite is used for a 3-dimensional fix (horizontal and vertical) [28].

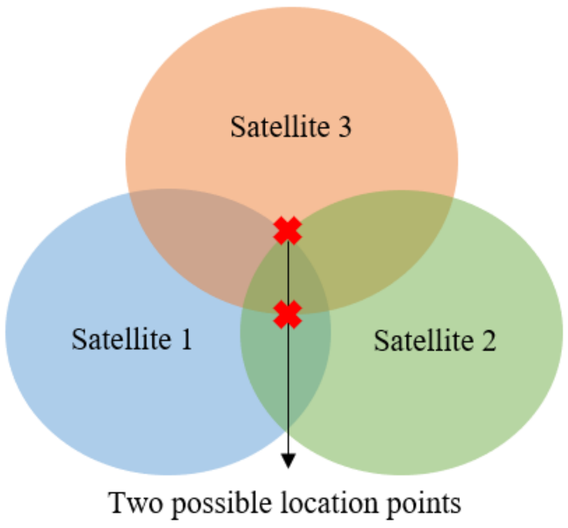


Figure 4: Satellite Ranging

GPS is being used in many gadgets in our everyday life such as smartphones, cars, smartwatches, fitness trackers etc. People around the world are using GPS regularly in their life for different usage of GPS. The main uses of GPS are- determining a position, navigation, timing, tracking, mapping. GPS has revolutionized the way we navigate and is now used in a wide range of applications, including aviation, maritime navigation, land surveying, and emergency services. GPS also has its usage in other industries like autonomous vehicles, drones, sales and services, the military, mobile communications, security, agriculture, fishing etc. Although GPS is a reliable and accurate system, sometimes its accuracy can be hindered by some outside factors like atmospheric conditions, signal obstructions, and receiver errors.

GPS has made a significant impact on our society. It has changed the way people communicate and live and made our environment a safer and easier place to live.

2.4 Geofencing

Geofencing is a location-based technology that enables to create virtual boundaries around a geographic area. This service is usually used in applications or software using GPS, Radio Frequency Identification (RFID), Wi-Fi or cellular data. Depending on how a pre-programmed action is configured, when a mobile device or an RFID tag enters in the virtual boundary, it triggers the action which can be an alert, message, notification, or other action [29].

Geofencing technology is mostly used within the code of mobile application or other software. In this case, GPS module is used to select a specific location according to the user’s decision. Then, around the specific latitude and longitude of that location, a distance-based radius defines a circle around the location on map. The circle works as a virtual boundary. Using the GPS module of the mobile device or other device, geofencing technology allows to trigger action when that device enters or exits the virtual boundary. Geofence can be implemented using APIs provided by third party service providers. Some popular APIs are Google API, Fencer, Geolocation Simulation, Boundaries.io, CA.boundaries.io, IP Geolocation etc. The most popular one is Google Geofencing API which provides support from a cloud server called Google Cloud Platform [30]. When geofencing monitoring is started, a request is sent to the web address of that API to generate a response if any geofencing event occurs. A required action is pre-set by the developer which will be prompted according to the response of API. These APIs are supported in several programming languages and SDKs such as Java, C#, Python, Node.js, PHP, Ruby etc.

With the rising use of smartphones and smart devices, popularity of geofencing is growing especially for plenty of businesses. Companies use geofencing for their marketing strategy to deliver in-store promotions, generating alert as users' step into the range of the store. Also, companies use this to monitor their employees who spend time off-site doing field work. This also helps to keep tabs on check-in and check-out of employees in the office. Geofencing also helps to escalate target ads to a specific audience based on their location. One of the major uses of geofence is seen in social networking application – Snapchat, Facebook, Instagram etc. They provide location specific contents, news, ad, filters, and stickers based on user's location. Usage of geofence can also be seen in the field of smart appliances, security, and audience engagement too.

Geofencing is creating a potential mark to revolutionize how businesses operate and engage with customers fully utilizing the benefits of this technology. In our work, we have implemented geofence in our mobile application to define the safe places for user where no alert will be generated if their phone is not in the pre-selected range.

2.5 Android Application

Android applications are software applications which run on android platforms. These platforms can be mobile, tablet or smartwatch that runs on Android OS. Android is an open-source operating system developed by Google. Therefore, many smartphone manufacturers use Android OS as operating system for their smartphones. Android verified applications are available on the Google play store. However, unverified applications or applications which are on development phase can also be installed using the apk file of the application.

Android applications can be developed using several programming languages like Java, Dart, Kotlin. The preferred integrated development environment software for android app development among developers are Android studio, IntelliJ IDEA, Visual studio etc. Also, there’s a famous open-source software development kit named Flutter which is used to develop both android and iOS applications. Android applications are developed for varied purposes and the available applications cover a wide range of categories like gaming, social networking, productivity, health and fitness, entertainment, and more.

With the availability of a wide range of free development tools and resources, it is easy to develop an application as well as cost-effective. With the help of Android studio, the OS has scaled up on flexibility and adaptability. It can integrate with not only smartphone but also tablet, wearables and android TV too making android apps compatible with IoT devices, AR, VR. Along with that, android platform has enhanced security with new in-build features to have protection against malware and viruses. Furthermore, as an open-source platform, it provides the freedom to customize and develop applications according to requirements with improved scalability and faster deployment.

With over 2.5 billion active users of Android OS spread across the world [31], android applications have become popular and an integral part of our daily lives, providing us with a vast range of services and entertainment options. With continued improvements in technology, Android apps will continue to evolve and offer new and innovative ways to make our life easier and comfortable.

Chapter 3

3 Literature Review

This chapter will review the related literature of previous works in smartphone loss prevention systems approach as well as pinpoint the shortcomings of existing studies. This section discusses different approaches of building the model: hardware based and software based approaches as well as the research gaps.

3.1 Hardware Based Approach

Hussain et al. [32] focused on a RFID based anti-loss and anti-theft solution. They have implemented their system in an RFID reader and equipped that device with a smartphone. For the system, the user is tagged with an RFID token which they attached in name tag, bracelet, watch, key chain etc. The reader emits radio frequency and receives signal back from their user worn tag and by this they keep track of the distance between user and the smartphone. The smart phone creates an audio and visual (Screen blinking) alert if the distance span goes beyond 2-10 feet. Their work has been based on Android operating system and they implemented an android user interface where user can control their profiles, transmission power, and various functions related to passive tag like managing, activating, and disabling the tag IDs. They have contributed to economizing battery overhead, mitigating false alarms, and scanning the tag for multiple times by leveraging the interrogation time and power to make their system more efficient. However, this is a costly solution ($60) as well as design is costly in power budget. Also, it requires equipping an extra device not only with phone but also a tag with the user which makes it less feasible solution.

Another work has been done on smartphone proximity absence detection using RFID technology. Aradhana et al. [33] worked on a IoT based system which will generate audio alert if user moves away more than 10 feet from his/her smartphone. The idea is to equip the cell phone with a low-power RFID Reader and tag the owner with a passive RFID token to keep track of the proximity presence. They presented a software compatible with

Windows XP/7/8/10, Android (up to 5.0.1) operating system. Before using the device, users need to login to their software interface and be authenticated using password. Along with that, the user needs to register his/ her phone number on the system. Whenever the RFID reader detects that the user is not in the appointed range it will generate an audio alert to grab the attention of the user. If the smartphone is not collected in a limited amount of time, the latitude and longitude location of the phone is sent to the specified number via SMS. Their solution offers to provide preventive measures from anti-loss and anti-theft, although the excessive cost of RFID reader and equipping additional device makes the solution less user-friendly.

Bzowski implemented a framework for a smartphone loss prevention system [34]. In this work, a number of Raspberry Pi have been used in user's frequently visited places like home, office, car etc. Those devices monitor the user's smartphone as they move between those frequently visited places and exchange their data with each other using a cloud infrastructure as central datastore. With this communication, the devices check that when the user enters into one entity, they use facial recognition to identify the user and check if the smartphone is with the user or not via searching for Bluetooth signature of the smartphone. If not found the devices will retrieve the last logged event from the shared datastore to determine if the smartphone is currently located in a trusted location where another device is present. If not found the system will attempt to communicate with the smartphone over the Internet and cellular networks to initiate "Find My Phone" services and to gather information on its current location and broadcast an auditory message using text to speech technology to deliver information to user such as where the is currently located and for approximately how long it has not been with the user.

A work has been developed by Yang et al. [35] named Surround-see which is an omni directional camera equipped with the smartphone. This enables peripheral vision around the device to observe the surroundings. Based on that it can trigger reminders based on an environment it recognizes, perceives, and points at specific objects in the mobile device's periphery and most importantly detects when the user walks away from it. As per the detection of user walking away, it can be used to generate an alert too. However, the additional camera itself is a big piece of equipment to be attached with the smartphone which is not feasible to carry with the smartphone. Moreover, it can only cover the front view of the camera, and with a larger area of blind sites, it is not effective for smartphone loss prevention system.

Despite of achieving good result, there are some disadvantages of carrying such additional devices like-

- One potential problem with relying on an additional device to receive alerts if the phone is not nearby is that it requires the user to always carry an extra device.
- An additional device is not practical or feasible for everyone, especially if you are traveling light or do not want to carry multiple devices.
- Another problem is that the additional device may also be lost or stolen, which would render the alert system useless.
- There is also the issue of battery life. If the additional device is constantly monitoring your phone's location and sending alerts, it may drain the battery quickly and require frequent battery replacing which could be inconvenient.
- Moreover, the devices use sound for alert, however, user might not hear the alert sound in a loud noisy public place.

3.2 Software Based Approach

Chaperone, a study by Chen et al. [36], provided a different solution which offered real-time locking and loss prevention for smartphones. This solution is a completely open-source, standalone android application and it does not require any additional device. The solution detects a phone’s unattended status by tracking the owner’s departure by active acoustic sensing via the built-in speaker and microphone of the smartphone. To help reproduce their findings, and improve acoustic sensing, they have collected dataset from both lab and real-world experiments. Once the system detects that the user is not near the phone, the phone triggers an alert method using, e.g., a ringtone, vibration, notification sound, or screen flashing. The system only performs the active acoustic sensing when the device is not in use, not on the user’s body, and in a potentially untrusted or public environment. They have conducted 1345 experiments in distinct locations and different conditions to determine the ability of Chaperone. And the system was able to detect these events in under 0.5 seconds for 95% of the successful detection cases. It also provides an overall precision rate of 93% and an overall recall rate of 96% for smartphone loss events.

Samsung has enabled a notification option for Galaxy watches. When the Bluetooth connection between smartphone and galaxy watch gets disconnected, it provides a notification. However, this is only a notification not an alert as well as user also can't set their preferred distance parameter for when they want to get an alert. Moreover, this feature is always enabled in all locations, even at home. Users raised concern about this as getting notification about Bluetooth disconnection at home is not necessary for them and sometimes can be annoying [37].

Li et al. presents a system called iLock that is designed to prevent data theft on mobile devices [38]. The iLock system is capable of quickly and accurately identifying when a user has become physically separated from their mobile device by analyzing changes in wireless signals. As soon as significant separation is detected, the device is automatically locked to prevent data theft. iLock relies on acoustic signals and can be used with most commercially available mobile devices that have at least one speaker and one microphone. The Samsung Galaxy S5 was used in numerous experiments to validate the effectiveness of iLock. These experiments demonstrated that iLock had a remarkably high success rate with very few false positives and negatives.

There are many mobile applications available such as Cerberus, Anti-theft Alarm, McAfee Mobile security, CrookCatcher, Prey, Find my device etc. which can help regarding finding the phones location, remotely locking the phone, securing data of the phone by remotely wiping data, transferring, and backing up data remotely [39]. But these applications are useful after a phone is already lost. They can't provide support to protect the phone from being lost.

3.3 Addressing the Existing Research Void

Although the above-mentioned research efforts have made a significant improvement in this area of research. However, there are some research gaps in the existing works which are mentioned below-

- The works of Hussain et al. [32] and Aradhana et al. [33] are based on RFID technology and both of their systems require equipping an extra device not only with phone but also a tag with the user which is less user friendly. Moreover, the solutions are costly as well as the distance threshold is fixed, and user can't change it. Furthermore, the system also works when the user is in a safe location (home, office etc.), which is unnecessary and annoying for users.

- In [34], we can see that the system requires multiple additional devices in several locations which are costly and less efficient. And this system doesn't provide instant alert which can prevent the smartphone from getting lost.

Table 2 : Existing System's Research Void

Reference	Equipping extra device	Costly Solution	Fixed threshold	Provides alert in both safe & unsafe Location	Doesn't provide Prompt Alert	False Negative Alerts	post-incident solution
[32]	✓	✓	✓	✓			
[33]	✓	✓	✓	✓			
[34]	✓	✓		✓			✓
[35]	✓	✓		✓		✓	
[36]				✓		✓	
[40]			✓	✓			
[38]				✓	✓	✓	✓
[39]				✓	✓		✓

- In [35], it requires an additional camera which is a big piece of equipment as well as expensive to be attached with the smartphone which is not feasible to carry with the smartphone. Moreover, it can only cover the front view of the camera, and with a larger area of blind sites, it is not effective for smartphone loss prevention system. Although the camera provides view of the peripheral space but presents pixel loss which makes the image distorted making the object detection harder. Furthermore, the object seen from the omni-directional lens is smaller than what can be seen with a normal lens. This makes object recognition harder for the system and produce wrong alerts. Additionally, the system also provides alert when the user is in a safe location, which is not necessary for users.

- Although the work of Chen et al. [36] provides a good accuracy, it generates some false negative results in terms of detecting the real owner and wrongly deciding the owner leaving by some movements of user where user is not actually leaving. Moreover, it provides alert even when user is at home or office which makes it bothersome for user.

- The notification of Samsung Galaxy watches regarding Bluetooth disconnection with the smartphone only generates notification when Bluetooth gets disconnected which is not a fixed distance threshold. Moreover, the notification system also works when the user is in a safe location (home, office etc.), which is unnecessary and annoying for users.

- The work of Li et al. [38] and the mobile applications such as Cerberus, Anti-theft Alarm, McAfee Mobile security, CrookCatcher, Prey, Find my device etc. [39] are effective for post-incident scenarios and will not provide instant alert to prevent the phone from getting lost. Moreover, the system also runs its service even when the user is in a safe location.

Unlike the above-mentioned research efforts, we have implemented our system contributing and minimizing the research gaps of the existing system.

- ✓ Our system can be implemented in any smartwatch or fitness band which can be purchased at an exceptionally low cost (starting from $15) which makes it a cheaper solution.
- ✓ Our system doesn’t have a fixed distance threshold, so users can change it according to their need.
- ✓ Our system doesn’t provide alert when the user is in a safe location and only works when user is in unsafe locations.
- ✓ Our system provides prompt alert with an average response time of 0.47 seconds.

- ✓ Although our system involves using an additional device, smartwatch, it is a wearable device which user doesn’t need to carry as well as it is a popular accessory that people mostly use in their everyday life.
- ✓ We have implemented 3 different models to implement different algorithms to evaluate their performance in terms of accuracy, battery optimization and reliability and find the best model which can provide good results in all of the parameters minimizing the false negative result ratio.

Apart from minimizing the research gaps of the existing system in our system, from business perspective, there are several reasons why users will be attracted to buy a smartwatch as a smartphone loss prevention system over other devices:

- For other smartphone loss prevention devices such as RFID devices and omni directional camera, prices start from $50 for RFID readers and $100 for omni directional camera. Whereas fitness bands or smartwatches can start at prices as low as $15 which makes it a cheaper solution compared to other solutions.
- For other loss prevention devices, they use sound and screen flashing as alert. However, in public places due to noises, users can fail to hear the sound alert or notice the screen flashes. Smartwatches always stays in touch with user’s skin. Therefore, when alert is generated, even if users miss the alert sound or screen light, users will be able to notice the vibration alert faster than any other solutions. Also, this is particularly helpful for individuals with hearing impairments as they can feel the vibrations on their wrist.
- Smartwatches are worn on the wrist and are easily accessible. Users are more likely to wear them consistently compared to other standalone loss prevention devices, which can be left at home or forgotten.
- Users don't need to dig through their bags or pockets to find a separate device when they want to use the loss prevention system.

Moreover, promoting smartwatches as loss prevention systems provides sellers and manufacturers of smartwatches with a unique selling proposition that can differentiate their products in a competitive market.

Chapter 4

4 Proposed Smartphone Loss Prevention System

In this chapter, we described our development platform and our proposed architecture that was developed using the platform for development. We have implemented 3 separate models to develop the system-

1. Model 1 (Integrated with RSSI of smartwatch and Android application)
2. Model 2 (Integrated with GPS of smartwatch and Android application)
3. Model 3 (Integrated with GPS and RSSI of smartwatch and Android application)

We have discussed all 3 models in this chapter.

4.1 Development Platform

Our development platform consists of a smartwatch from Espurino (Bangle.js 2), Espurino IDE, Android Studio and an android smartphone from Samsung (Galaxy S22) and details are discussed below:

We have used the latest version of Espurino watch, Bangle.js 2. We have implemented a part of Model 1 and Model 2 in Bangle.js 2 and Model 3 is fully implemented in Bangle.js 2. To maintain the connectivity with the smartphone that we are preventing from getting lost, we have used Bluetooth Low Energy (BLE). Bangle.js 2 uses Nordic nRF52840 chip to implement the BLE feature as well as power management. Nordic nRF52840 chip allows concurrent Bluetooth 5.2 which is the latest version of Bluetooth technology which provides some key benefits [41]-

- Increased Data Transfer Speeds: Bluetooth 5.2 provides a data transfer facility which is faster than the previous version, Bluetooth 5.0.
- Improved Range: Bluetooth 5.2 has increased the range of Bluetooth which broadens the radius of connectivity area.

- Reduced Interference: Bluetooth 5.2 features have reduced interference with other wireless technologies such as Wi-Fi and LTE.
- Improved Security: Bluetooth 5.2 features enhanced security measures which provide improved encryption and protection against eavesdropping and hacking.

This BLE feature has been used to implement all the 3 models of ours. In addition, we have also used the AT6558 navigation chip of the Bangle.js 2 to use the GPS feature. This chip is able to obtain accurate global location information, quick and accurate positioning for any location [42]. It is also designed to be highly power-efficient, which makes it ideal for use in battery-powered devices like wearables and smartphones. GPS of Bangle.js 2 has been used in our Model 2 and Model 3 in different manners.

Espurino which is an open-source firmware has its own IDE to implement programs in Bangle.js 2. We have programmed it from our Chrome browser (Version 111.0.5563.65). This is a wireless process where the connectivity relies on web Bluetooth. The programming on Bangle.js 2 is solely dependent on JavaScript which was the programming language that we used to implement our algorithms for our models.

We have also developed an android application for Model 1, Model 2 and Model 3 using Android studio. Android Studio is the official IDE for Android app development which provides a powerful and user-friendly interface for developing Android applications, including tools for designing user interfaces, coding, debugging, and testing. We have used the programming language Kotlin to develop our algorithm in android studio. The developed application later has been deployed in our smartphone, the Samsung Galaxy S22.

Galaxy S22 is one of the most recent smartphones released by Samsung. It is equipped with the most updated Processor, Bluetooth module and positioning system [Table 3]. We will be using the Bluetooth module to keep connectivity with the smartwatch in all the models. Same as Bangle.js 2, it also allows Bluetooth 5.2 which let them establish a communication link using Bluetooth Low Energy (BLE) and can exchange data wirelessly at high speeds while maintaining a stable and reliable connection. Furthermore, Snapdragon8450 uses the latest technology for positioning features which provides excellent GPS accuracy. We have used this GPS module of Galaxy S22 in our Model 1 and Model 2 to provide safe-zoning feature to our work.

Table 3: Smartphone Specification[43]

Name	Galaxy S22
Processor	Snapdragon8450
OS	Android 12, upgradable to Android 13
Memory	128GB 8GB RAM, 256GB 8GB RAM
Bluetooth	5.2, A2DP, LE
Positioning	GPS, GLONASS, BDS, GALILEO
Battery	Li-Ion 3700 mAh

We utilized a Windows 10 PC Processor Intel(R) Core (TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz and 8.00 GB of RAM for developing both applications for the smartphone and smartwatch.

4.2 Proposed System

Our proposed system offers users an alert system installed in a smartwatch which will notify the users when they are 20 feet or their preferred distance away from their phone. And this alert will generate every place except for users saved safe locations. Therefore, for our proposed system, calculating the distance between smartphone and user and tracking whether user is in a safe zone area are the main key areas. We have implemented 3 separate models to achieve the results of these two areas which are discussed in the following sections. Our research goal is to see how accurately each model performs to generate alerts. Also, we focused on the battery optimization and response time performance of each model as well. Moreover, we have implemented similar algorithms in different modules (smartphone and smartwatch) in each model to compare their performance in different modules and find the most efficient, reliable, and best fitted model. Our core workflow of our proposed system is executed by running a series of steps [Figure 5] repeatedly.

Initially, users can provide two types of information-

- **Preferred Distance:** This value represents the maximum threshold limit of the distance between smartphone and smartwatch. When the distance between smartphone and smartwatch goes beyond the preferred distance, an alert will be generated on the smartphone. If the user doesn't set this, 20 feet will be considered as default value.

- **Safe Location set:** User can set their safe locations such as home, office etc. where alerts will not be triggered if the distance between smartphone and smartwatch goes beyond the preferred distance.

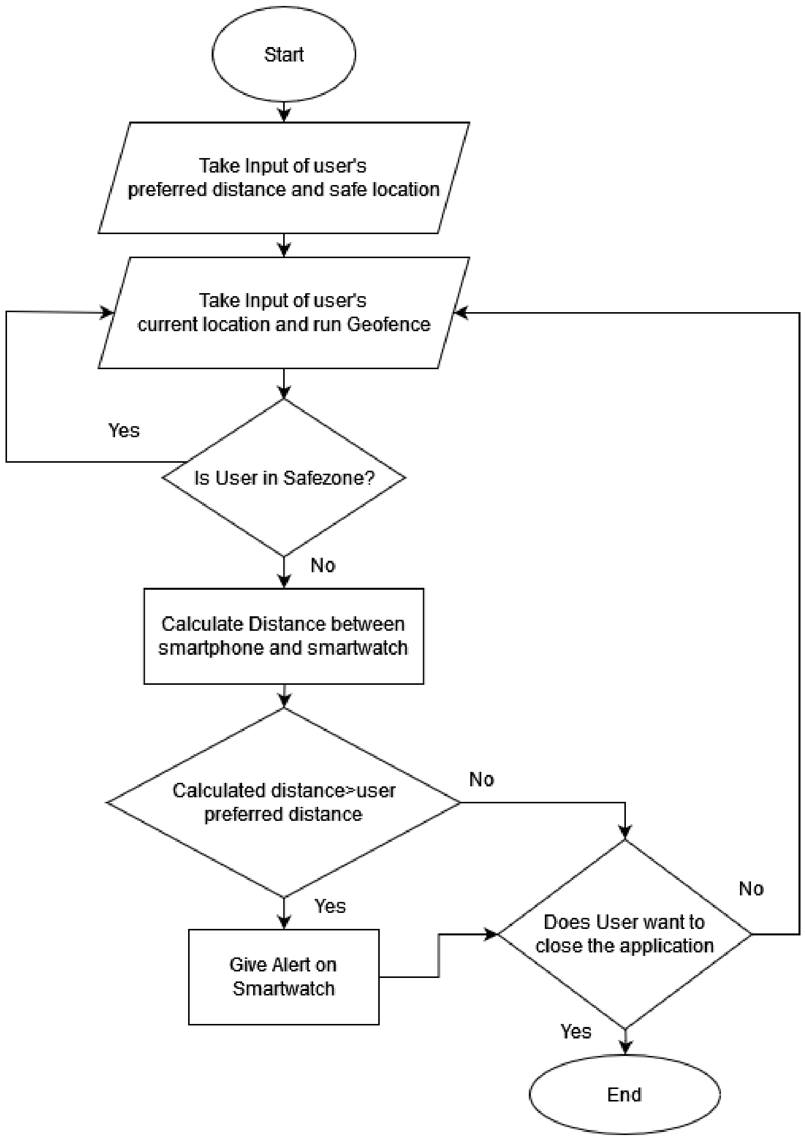

Start
Take Input of user's preferred distance and safe location
Take Input of user's current location and run Geofence
Yes
Is User in Safezone?
No
Calculate Distance between smartphone and smartwatch
Calculated distance>user preferred distance
No
Yes
Give Alert on Smartwatch
No
Does User want to close the application
Yes
End

Figure 5: Workflow of Proposed System

The system starts its repeated process running the following steps in every process. The system first takes users current locations input and run geofencing algorithm. By the geofencing algorithm, the system gets the result whether user is inside a safe location or not. Based on the 2 possible events – a) User is not in safe location and b) User is in safe location, the system executes the following steps-

a) **User is not in safe location:** In this case, system runs the algorithm for calculating the distance between user and smartphone. As our system is installed in the smartwatch worn by the user, ideally, we will calculate the distance between the smartwatch and smartphone. For this distance calculation we have implemented different algorithms in our 3 models to compare their accuracy and efficiency which will be discussed in the following sections. After receiving the distance value, the system will check whether it is greater than the users preferred distance or not.

    i) If it is greater than the preferred distance, the smartwatch will generate an alert to notify the user that their smartphone is not nearby.

    ii) However, if the distance value is not greater than the preferred distance then it will not generate any alert.

    After that the user will check if the user wants to close this application. If yes, then the system will end its process otherwise it will go to the first step and repeat the entire process.

b) **User is in safe location:** In this scenario, the system doesn't execute the distance calculation algorithm as user doesn't need any alert in safe locations. Therefore, to reduce computational cost, we skipped the distance calculation algorithm. When the user is in a safe location, the system only checks whether the user wants to close the application or not. If the user wants to close the application, the system ends the process. Otherwise, it will go to the first step and repeat the entire process.

For implementing the two key areas- distance calculation and safe location tracking, we have implemented different algorithms and techniques to compare their accuracy and efficiency. Following these different approaches, we have built 3 different smartphone loss prevention system models which are described in the following sections.

4.3 Model 1 (Integrated with RSSI of smartwatch and Android application)

Our Model 1 consists of an android app installed in the smartphone that we want to prevent from getting lost and the smartwatch that user wears. In this model, the user directly interacts with the android application, and doesn't need to interact with the smartwatch. Through the android application, the user can set their preferred distance and safe locations. Based upon adding a new safe location or existence of previously safe location/s, the system starts the geofencing algorithm. The broadcast receiver captures the event if user enters or exits from any safe location. If any event occurs or the user changes the preferred distance value, the application sends an intent to bangle.js 2 smartwatch which contains the event information. Even if the user doesn't set any preferred value, the system works perfectly considering default values. The default value of preferred distance is 20 feet and if user doesn't have any previously saved safe location, the system will consider that user is in unsafe location always.

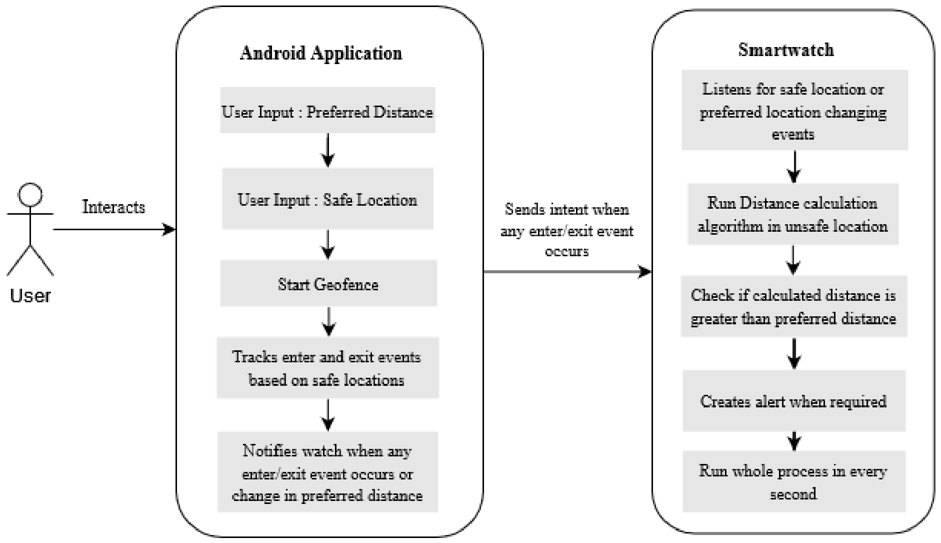

User
Interacts
Android Application
User Input : Preferred Distance
User Input : Safe Location
Start Geofence
Tracks enter and exit events based on safe locations
Notifies watch when any enter/exit event occurs or change in preferred distance
Sends intent when any enter/exit event occurs
Smartwatch
Listens for safe location or preferred location changing events
Run Distance calculation algorithm in unsafe location
Check if calculated distance is greater than preferred distance
Creates alert when required
Run whole process in every second

Figure 6: Model 1 Architecture

In the smartwatch end, the system listens for any event information sent by the android applications and updates accordingly in smartwatch's end. Based on user's current location whether user is inside a safe location or not, it runs the distance calculation algorithm and generates alert when necessary. The whole process runs in every second repeatedly.

4.3.1 Android Application Module for Model 1

As one of the main functions of our android application is to work with geofencing, we need some permissions from the device to execute this. Our application mostly depends on the location feature of the phone via GPS module. Usually, for any new application of smartphone most of the permissions stay denied by default. For requesting geofence

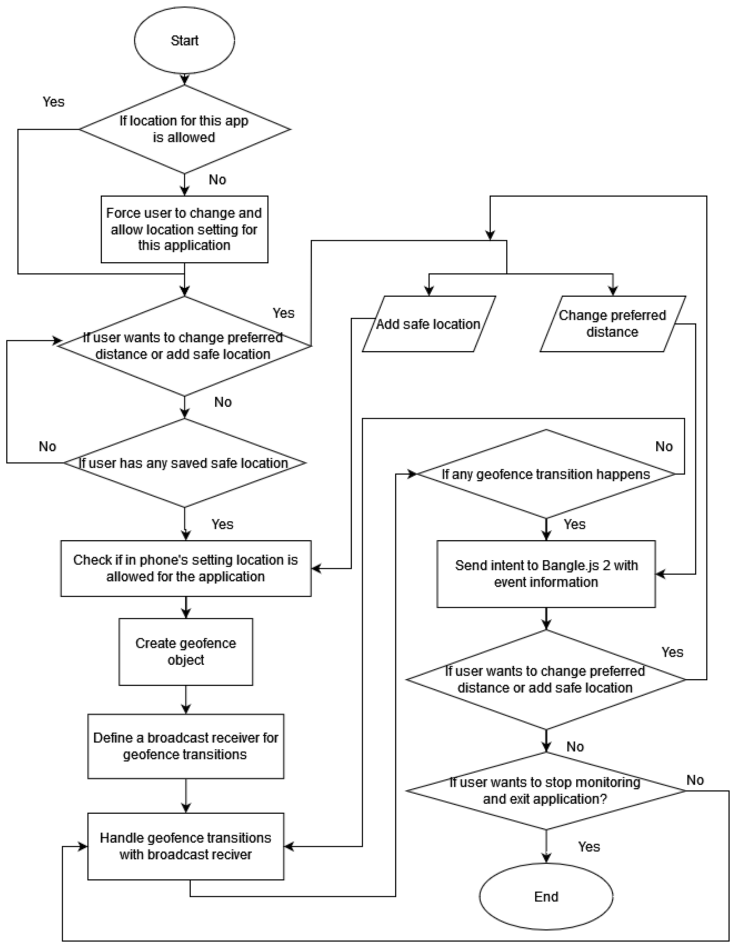

Start
If location for this app is allowed
Yes
No
Force user to change and allow location setting for this application
If user wants to change preferred distance or add safe location
Yes
No
Add safe location
Change preferred distance
If user has any saved safe location
No
Yes
If any geofence transition happens
No
Yes
Check if in phone's setting location is allowed for the application
Send intent to Bangle.js 2 with event information
Create geofence object
If user wants to change preferred distance or add safe location
Yes
No
Define a broadcast receiver for geofence transitions
If user wants to stop monitoring and exit application?
No
Yes
Handle geofence transitions with broadcast reciver
End

Figure 7: Workflow of Android application Module for Model 1

monitoring, it needs to request the necessary permissions. To use geofencing, your app must request the following:

- ACCESS_FINE_LOCATION : To access GPS data such as device's precise location, the system needs this permission to be allowed.
- ACCESS_BACKGROUND_LOCATION: Background location access is needed if the app constantly shares location with other users or uses the Geofencing API [44]. As our application will use the Geofencing API, we need background location permission.

These permissions can be granted by the user by selecting "Allow all the time" option for the application's location setting. For our application, the system first checks if these permissions are allowed for our app or not. If it's not allowed, then our application forcefully takes the user to smartphone's setting to grant "Allow all the time" access for our app following the Algorithm 1.

Algorithm 1: Setting Location Permission

```
1:  Check if the permission for the location index is denied or the permission for
    the background location index is denied.
2:  IF any of the above conditions are true, THEN
3:          Show a Snackbar message that explains the reason for the permission
            and provides a button to take the user to the app settings screen.
4:          The Snackbar message should be displayed using the make() method.
5:          The setAction() method is used to set the text and behavior of the button.
6:          The startActivity() method is used to open the app settings screen.
7:  END IF
```

After granting the permission, the user can change the preferred distance as well as add safe locations. For this model, the user can save a maximum of 5 safe locations. The system has listeners to check whether they perform these changes by implementing button listeners. If the user changes the preferred distance, a new intent is created, and the system sends the intent to bangle.js 2 by implementing Algorithm 2. The systems add the preferred distance value given by user as “event” in JSON object which is sent to the Bangle.js 2.

Algorithm 2: Sending Intent to Bangle.js 2

```
1:  Create a new JSONObject and assign it to jsonData
2:  Set the value of event in jsonData naming it as ‘t”
3:  Create a new Intent with the action "com.banglejs.uart.tx" and assign it to
    sendIntent.
4:  Set the "line" extra of sendIntent to "\u0010GB($jsonData)\n".
5:  Broadcast sendIntent using sendBroadcast method
```

On the other side, for adding a safe location, the user needs to be present in that location. User needs to provide 2 information-

- **Name:** Name of the safe location.
- **Radius:** Geofencing algorithm marks every safe location as a circle. User needs to provide the radius value of the circle based on users need about how much area user wants to include in the safe location zone. The unit of the radius in our work is feet.

Upon getting the information from the user, the system calls Algorithm 3 and fetches the user’s current location. The system integrates with Google API to get the current location of the user. From that, the system extracts latitude and longitude information and saves the data to be persistent across application restarts. For storing the data, we have used Shared Preferences which is a persistent way to store and retrieve data in android applications.

Algorithm 3: Get Current Location

```
1:  Build a GoogleApiClient object using the GoogleApiClient.Builder() method and
    assign it to the variable mGoogleApiClient.
2:  Add the LocationServices.API to the GoogleApiClient using the addApi()
3:  Connect to the GoogleApiClient using the connect() method on the
```

```
     mGoogleApiClient object.
4:   Create a LocationRequest object using the create() method and assign it to the
     variable mLocationRequest.
5:       Set the priority of the location request to PRIORITY_HIGH_ACCURACY
         using the setPriority() method.
6:       Set the interval between location updates to UPDATE_INTERVAL using
         the setInterval() method.
7:       Set the fastest interval for location updates to FASTEST_INTERVAL using
         the setFastestInterval() method.
8:   Check if the app has the required location permissions calling Algorithm 1.
9:   **IF** the app doesn't have the required location permissions, **THEN**
         return from the method.
10:  **ELSE**
11:        Request location updates using the LocationServices.FusedLocationApi.
           requestLocationUpdates() method, passing in the mGoogleApiClient,
           mLocationRequest, and 'this' as arguments.
12:  **END IF**
13:  Retrieve the latitude value from mLocation and store it in the "latitude" variable.
14:  Retrieve the longitude value from mLocation and store it in the "longitude"
     variable.
```

After receiving the latitude and longitude value, the system stores the value in a shared preference. Shared preferences don’t provide correct value if it is modified after saving one. Therefore, the system has enabled to create maximum 5 shared preferences which will store the value of 5 safe locations.

Before starting geofencing, the system checks how many shared preferences has been created. If no shared preferences has been created, that means the user didn’t add any safe location and the system will not start geofencing process. If one or more than one Shared Preferences is found, then the system starts the geofencing process. To implement the geofencing process, our system uses Google Geofencing API. To use the Geofence API, the Google Maps API key has been integrated in the manifest through Google cloud Platform. First, the system creates a geofencing object with all the saved safe location's latitude, longitude, name, and radius value running Algorithm 4. While building every geofences, the Id will be the name of the safe location. And for the circular region, the latitude and longitude value of the safe location is considered as center of the circle and radius of the safe location is the radius of the circle. As transition type, system only monitors 2 transitions for each geofence-

- GEOFENCE_TRANSITION_ENTER: When user enters in a safe location, this event occurs.
- GEOFENCE_TRANSITION_EXIT: when user exits a safe location, this event occurs.

Defining these in transition type allows the broadcast receiver to take necessary steps in case these events occur. For every saved safe location, a geofence is created and added to a list named "geofencelist". Then, we create new geofencing requesting adding the "geofencelist" which needs to be monitored and build it in step 14 of Algorithm 4.

Algorithm 4: Create Geofencing Object

```
1:  FOR each created safe location, DO
2:          Create a new geofence using the Geofence.Builder() method
3:          Set the request ID for the geofence to the name of the safe location
4:          Set the circular region for the geofence to the latitude, longitude, and
            radius of the safe location.
5:          Set the expiration duration for the geofence.
6:          Set the transition types "GEOFENCE_TRANSITION_ENTER" and
            "GEOFENCE_TRANSITION_EXIT" of interest for the geofence.
7:          Build the geofence object and add it to the "geofencelist".
8:  END FOR
9:  Build a new geofencing request using GeofencingRequest.Builder(). method.
10:         Set the initial trigger "GEOFENCE_TRANSITION_ENTER" and
```

```
            "GEOFENCE_TRANSITION_EXIT"  for the geofencing request.
11:         Add the "geofencelist" to be monitored to the geofencing request.
12:         Build the geofencing request.
13:     Remove any existing geofences that use the pending intent for geofence updates.
14:     Add the new geofence request with using geofencingClient.addGeofences()
        method with new geofencing Request and PendingIntent as parameters.
```

Geofencing function should be running in the background after closing the application and handle the geofences transitions. To implement that, we have used Broadcast Receiver. A BroadcastReceiver gets updates when an event occurs, such as a transition into or out of a geofence and can start long-running background work [45]. We have defined a PendingIntent to start and define the Broadcast Receiver in Algorithm 5. Every time a new geofencing request is built, geofence is added and it uses PendingIntent to start broadcast receiver anew. When user adds a new safe location, it needs to be added to the geofence object. Therefore, system calls the Algorithm 4 from step 1 to add the new safe location to the geofence object. However, previously created geofence still uses the Pending Intent. Thus, it is important to remove any existing geofence object that uses pending intent. And so, before adding the new geofence in step 14, system removes any existing geofence that uses system’s Pending Intent.

Algorithm 5: Define Broadcast Receiver

```
1:  Initialize a lazy value geofencePendingIntent with the following code block:
2:          Create a new Intent object with context this and
            GeofenceBroadcastReceiver::class.java as parameters, and assign it to
            variable intent.
3:          Set the action of intent to ACTION_GEOFENCE_EVENT.
4:          Create a new PendingIntent object using PendingIntent.getBroadcast
5:          with the following parameters:
6:                  this: current context
7:                  Request code for the PendingIntent
8:                  intent: the Intent object created in step 2
```

```
9:                PendingIntent.FLAG_UPDATE_CURRENT: flag to update the
                  existing PendingIntent if it already exists.
10:         RETURN the PendingIntent object created in step 4.
```

The location services of the system listen for the events when user enters or exits any safe location. When this event occurs, it sends the Intent contained in the PendingIntent which was included in the geofence request to add geofences. The system's broadcast receiver can notice when any Intent invokes and extracts the geofencing event from the intent. Later, it sees the type of transition. Based upon type of transition, it sends intent to Bangle.js 2 following Algorithm 2. If the transition is "GEOFENCE_TRANSITION_ENTER", it sends "enter" in the "event" attribute of Algorithm 2. Otherwise, if the transition is "GEOFENCE_TRANSITION_EXIT", it sends "exit" in the "event" attribute of Algorithm 2.

This geofencing monitoring runs in the background even after closing the application and notifies the smartwatch promptly when any triggering event occurs.

4.3.2 Smartwatch Module for Model 1

The key functionality of this module is to calculate the distance between the smartphone that we want to prevent from getting lost and the smartwatch that will be worn by the user. For calculating distance, we will be using the RSSI value of the Bluetooth connection between smartphone and smartwatch. To incorporate the RSSI value in distance calculation, our system uses Ranging method's formula which uses measured signal strength for RSSI value-

$$Distance = 10^{\left(\frac{Measured\ Power - RSSI}{10 \times N}\right)} \text{ Meter[18]}$$

For Bangle.js 2 the measured power is considered as -56 and we have set the value of N=2 considering our system may have to work in an indoor environment which may affects the signal strength. Moreover, the unit of ranging formula is Meter. To convert it to feet, we have multiplied the distance by 3.28084. Therefore, our final formula will be-

$$Distance = 10^{\left(\frac{-56-RSSI}{10\times 2}\right)} \times 3.28084\ feet$$

$$= 10^{\left(\frac{-56-RSSI}{20}\right)} \times 3.28084\, feet$$

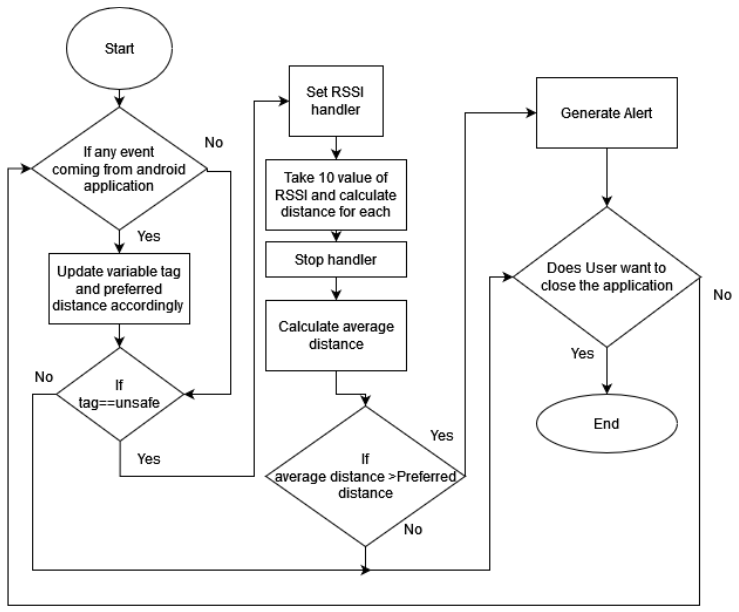

Start
If any event coming from android application
No
Yes
Update variable tag and preferred distance accordingly
If tag==unsafe
No
Yes
Set RSSI handler
Take 10 value of RSSI and calculate distance for each
Stop handler
Calculate average distance
If average distance >Preferred distance
Yes
No
Generate Alert
Does User want to close the application
No
Yes
End

Figure 8: Workflow of Smartwatch Module for model 1

Initially, the preferred distance is set as 20 feet. The variable that keeps track of whether user is in safe location or unsafe location is called “tag” for our system. The default value of “tag” is “unsafe” so that it can start the safety measure steps for preventing the smartphone from getting lost immediately. Later, the system can act upon receiving the event sent from the android application. System uses Event Listener, Algorithm 6, to update the preferred distance and safe/unsafe location information sent by the android application. The system updates the value in its own environment variable “preferredDistance” and “tag”.

Algorithm 6: Event Listener

```
1:  Define a global function called "GB" that takes an event as input.
2:  Check the type of the event by evaluating the "t" property of the event object.
3:  IF the type of the event is "enter", THEN
4:          Set the value of the "tag" variable to "safe".
```

```
5:  END IF
6:  IF the type of the event is "exit", THEN
7:      Set the value of the "tag" variable to "safe".
8:  END IF
9:  IF the type of the event is not "enter" or "exit", THEN
10:     Parse the "t" property of the event as an integer
11:     Store it in the "preferredDistance" variable.
12: END IF
```

When the user is in a safe location, running the distance calculator, Algorithm 7, is not necessary as well as it will save computational cost and energy. Therefore, we only run the distance calculation algorithm when the user is in an unsafe location. To get the RSSI value, we use NRF.setRSSIHandler(data) function. This is a method of the Nordic Semiconductor's SoftDevice API that sets a callback function to handle changes in the Received Signal Strength Indication (RSSI) value of the Bluetooth Low Energy (BLE) connection. The “data” parameter in the callback function is an object that contains information about the current RSSI value. When we set the handler, it starts providing the RSSI values. Sometimes, for noise and interference, some RSSI values can be affected. However, if we consider taking 10 values of RSSI and use the average value, that one affected RSSI value will not be able to make any tremendous changes as it will be scaled after calculating average. Therefore, for more accurate value, the system takes 10 values of RSSI and calculates distance for those 10 RSSI values separately. After getting 10 values, the system stops the handler so that it can stop fetching RSSI data for a while. As the handler keeps fetching RSSI value in a few milliseconds’ duration, stopping the handler saves some computational cost and prevents from having a long thread queue and making RAM memory full. If the system finds that the average distance is greater than the preferred distance that means that user went beyond the preferred distance value leaving his/her phone behind. In this scenario, our system generates an alert on the smartwatch. It will prompt a message on the smartwatches screen “Your Phone is not nearby.” Which will turn on the screen light. Moreover, the system will call the Bangle.beep function 5 times for generating vibration and call Bangle.buzz functions 5 times to generate sound to alert the user. Thus, our system generates the alert targeting users 3 senses – audio, visual, touch. User can dismiss the message clicking the okay button on the smartwatch's screen.

Algorithm 7: Distance calculation using RSSI

```
1:  Declare and initialize “rssicount” and set variable to 0.
2:  Declare and initialize “sum” and set variable to 0.
3:  IF tag equals "unsafe",  THEN
4:      Set RSSI handler to function(data)
5:          Calculate distance = 10 raised to the power of
            ((-56 - (data)) / (20)) multiplied by 3.28084.
6:          Add distance to sum.
7:          IF increment of rssicount is greater than 9 THEN
8:              Remove RSSI handler.
9:              Calculate average distance by dividing sum by 10.
10.             IF average distance is greater than preferredDistance THEN
11.                 Display prompt "Your Phone is not nearby." with title
                    "Alert!".
12.                 Call the beep function of the Bangle object 5 times.
13.                 Call the buzz function of the Bangle object 5 times.
14:             END IF
15:         END IF
16:     END FUNCTION
17: END IF
18. WAIT for 1000 milliseconds to rerun this algorithm.
```

This algorithm keeps running every 1000 milliseconds to provide a constant monitoring to prevent the smartphone from getting lost.

4.4 Model 2 (Integrated with GPS of smartwatch and Android Application)

Our model 2 uses an android application interface for user interaction similar as the Model 1. Users can provide their preferred distance and safe location information via application interface. In this model, the user can save a maximum of 5 safe locations like Model 1 using Shared Preferences. However, for this model, we have implemented GPS based distance calculator algorithm using haversine formula. Haversine formula calculates the distance between two points using the latitude and longitude of both points. In order to calculate the distance between smartphone and smartwatch, we need the current GPS

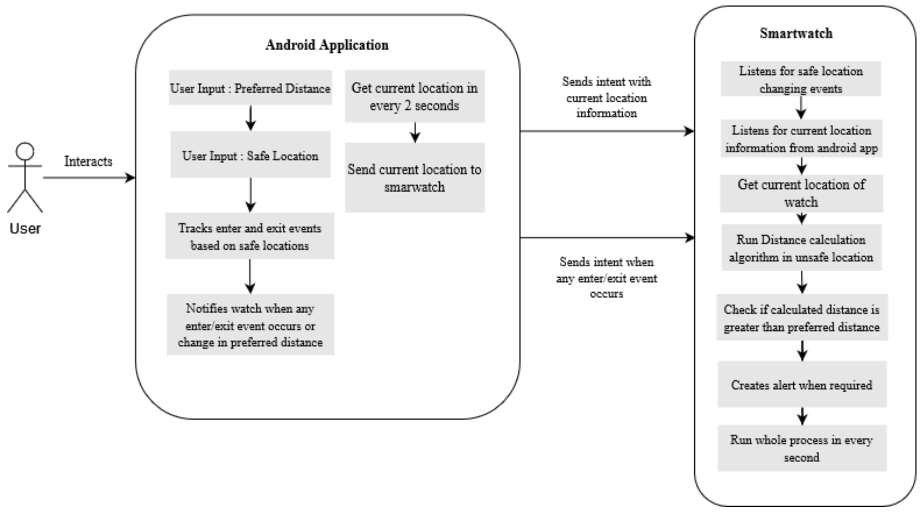

Android Application
User Input : Preferred Distance
User Input : Safe Location
Tracks enter and exit events based on safe locations
Notifies watch when any enter/exit event occurs or change in preferred distance
Get current location in every 2 seconds
Send current location to smarwatch
User
Interacts
Sends intent with current location information
Sends intent when any enter/exit event occurs
Smartwatch
Listens for safe location changing events
Listens for current location information from android app
Get current location of watch
Run Distance calculation algorithm in unsafe location
Check if calculated distance is greater than preferred distance
Creates alert when required
Run whole process in every second

Figure 9: Model 2 Architecture

information of both smartphone and smartwatch. And the location of both smartphone and smartwatch may change from time to time. So, the system calls the current location fetching algorithm for both smartwatch and smartphone in every 2 seconds and updates the variables accordingly. However, when user is in safe location, system doesn’t call the current location fetching algorithm for both modules to save energy and reduce computational cost. When a user is in an unsafe location both modules run the location fetching algorithm and smartwatch module runs the distance calculation algorithm and generates alert when it is needed.

4.4.1 Android Application Module for Model 2

Similar as Model 1, for Model 2, it is mandatory that the location setting for this application must be “Allowed all the time”. We have used Algorithm 1 to check the location setting for the application and take the user to settings of phone if the location permission is not in allow mode for this application. For saving the safe locations, the system uses Shared Preferences and Algorithm 3 to get the current location same as model 1.

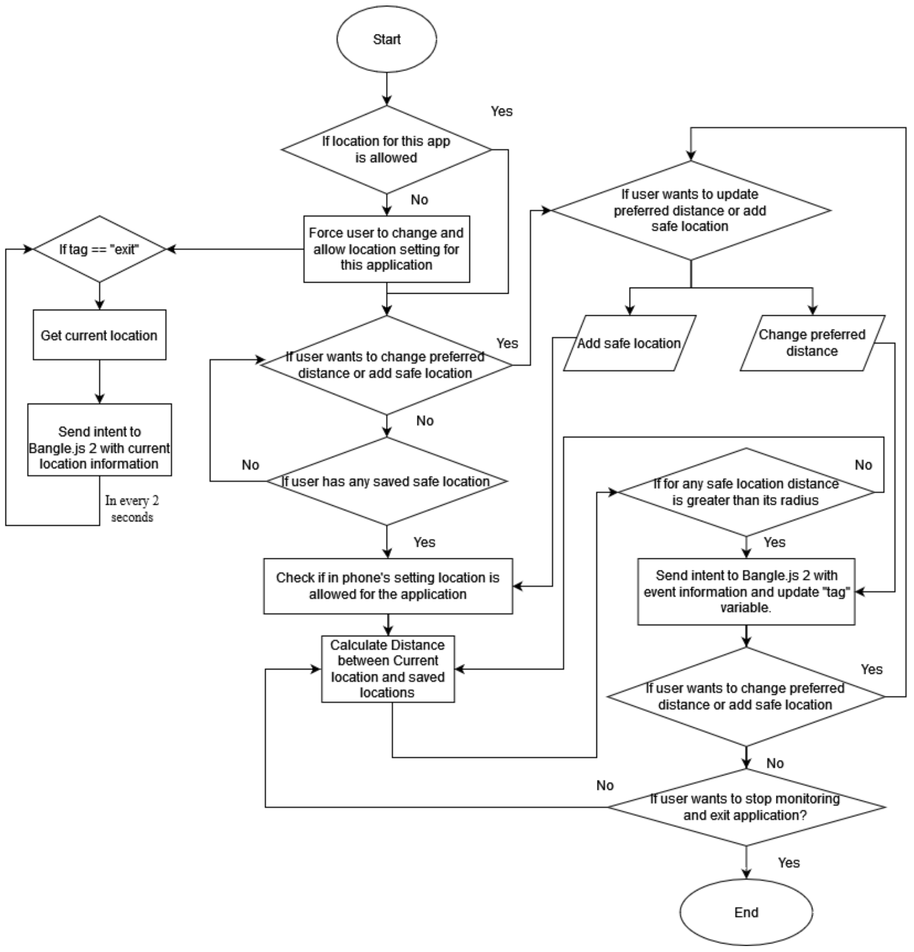

Start
If location for this app is allowed
Yes
No
Force user to change and allow location setting for this application
If tag == "exit"
Get current location
Send intent to Bangle.js 2 with current location information
In every 2 seconds
If user wants to change preferred distance or add safe location
Yes
No
If user has any saved safe location
No
Yes
Check if in phone's setting location is allowed for the application
Calculate Distance between Current location and saved locations
If user wants to update preferred distance or add safe location
Add safe location
Change preferred distance
If for any safe location distance is greater than its radius
No
Yes
Send intent to Bangle.js 2 with event information and update "tag" variable.
If user wants to change preferred distance or add safe location
Yes
No
If user wants to stop monitoring and exit application?
No
Yes
End

Figure 10: Workflow of Android Application Module for Model 2

For model 2, we didn’t implement the geofencing algorithms using Google API in android application. We have used a distance calculation technique using haversine formula to keep track whether user is in safe location or not. For the distance calculation using haversine formula for both android application module and smartwatch module, system needs its current location which needs to be updated in every 2 seconds. Moreover, as we are not using any broadcast receiver, we need to keep the app running if the background after closing the application to send smartwatch current location of the smartphone constantly and to track whether the smartphone has entered or exited any safe location. To execute that, we have created a location service that continues to work even after the Android application is closed.

The service uses the Fused Location Provider API to get the user's current location. Using onCreate method of the service, a new instance of the Fused Location Provider API is created. In the onStartCommand method of the service, the system fetches smartphone’s current location using the Fused Location Provider API following Algorithm 8. As our system starts the location fetching service onCreate method, the application continues to run even after the application is closed. This algorithm extracts the user’s current location and updates the latitude and longitude variable. For this model, we need to get the current location in every 2 seconds as well as send the updated latitude and longitude information to smartwatch in every 2 seconds. Therefore, we run the Algorithm 8 with time intervals of 2 seconds. Along with getting the current information, it also sends the latitude and longitude information to smartwatch calling the Algorithm 2 in every 2 seconds. While sending intent to bangle.js 2, the system puts latitude and longitude value in the “event” variable of the Algorithm 2.

Algorithm 8: Location Service

```
1:  Set the locationClient to DefaultLocationClient with the applicationContext and
    location provider client.
2:  Call getLocationUpdates on locationClient with a time interval of 2000L
3:  On each update:
4:          Extract the latitude from the location object and store it in latitude variable.
5:          Extract the longitude from the location object and store it in longitude
            variable.
6:          Call Algorithm 2 passing Latitude and Longitude information in “event”
```

```
        variable.
7:  Launch the coroutine on the serviceScope.
```

Another key step is to check if the user is in a safe location or not. The system uses the following haversine formula to implement that-

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 \,.\cos\varphi_2 \,.\sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2\,.\,\mathrm{atan2}\left(\sqrt{a}\,, \sqrt{}(1-a)\right)$$

$$d = R\,.\,c\ \ km$$

For the value of R, the system uses 6372.8 as a value of radius of earth. Also, as the unit of the Haversine formula is kilometer, the system converts it to feet by multiplying with 3280.84. So, the final formula is –

$$d = 6372.8\,.\,c.\,3280.84\ \ km$$

To calculate the distance between smartphone's current location and each saved location, the system calls Algorithm 9 which uses haversine formula.

Algorithm 9: Haversine

```
1:  Define a function named haversine that takes four Double parameters - lat1, lon1,
    lat2, and lon2.
2:  Set the value of R to 6372.8.
3:  Convert latitudes and longitudes to radians by using the toRadians method
    and save them in variable a1 and a2.
4:  Calculate the differences between latitudes and longitudes using (lat2 - lat1) and
    (lon2 - lon1) formulas saved as b1 and b2 and convert them to radians.
5:  Use the haversine formula:
6:      Calculate the value of sin^2(b1/2) + sin^2(b2/2) * cos(a1) * cos(a2) and
        store it in variable c.
7:      Calculate the square root value of c and (1-c) and save them in variable c1
```

```
          and c2.
8:        Calculate the value of 2 * R * atan(c1, c2).
9:    Convert the calculated distance from kilometers to feet by multiplying the result
      by 3280.84.
10:   **RETURN** the result.
```

Before running the distance calculation algorithm, the system checks if the user has any saved safe location or not. If the user has one of more than 1 saved safe locations, then it starts the process for distance calculation. For each saved safe location, the system calculates the distance between current location and safe location using Algorithm 10. For this, the system uses the latitude and longitude value of both locations. When user is in a safe location, the distance value will always be less than the radius of the safe location. And, when user is not in a safe location, the distance value will always be more than the radius of the safe location. Implementing this concept, the system keeps track whether the user is in a safe location or not. Therefore, after calculating the distance, it checks if the distance value is more than the radius of the safe location. For any of the safe locations, if this event happens the system updates the value of the variable “tag” to “unsafe”. Otherwise, it sets the value of “tag” to “safe”.

Algorithm 10: GPS distance calculator

```
1:    IF user has 1 or more than 1 safe location THEN
2:        Create an ArrayList named List1 with all the saved safe locations info.
          Initialize the variable i to 0.
3:        FOR each item in List1 DO:
4:            Call Algorithm 9 to calculate distance passing parameters- current
              location’s latitude, longitude, current item’s latitude, longitude.
5:            IF calculated distance <radius of the current item, THEN
6:                set the value of i to 1.
7:            END IF
8:        END FOR
9:        IF the value of i is 1 THEN
```

```
10:             Set “tag” to "safe" and call Algorithm 2 to send tag information to
                smartwatch.
11:         ELSE
12:             Set “tag” to "unsafe" and call Algorithm 2 to send tag information to
                smartwatch.
13:         END IF
14:     END IF
```

The system needs to run Algorithm 10 in every 2 seconds to get updated about if the user has entered or exited any safe location. To incorporate that, we have user handler class which calls the two algorithms in every 2 seconds following the Algorithm 11.

Algorithm 11: Schedule Service

```
1:  Initialize handler variable as a new instance of Handler class.
2:  Initialize runnable variable as null.
3:  Initialize delay variable with the value 2000
4:  Define the onResume() function
5:      Call the postDelayed() method of the handler object with a new instance
        of the Runnable interface as the first argument.
6:      Inside the Runnable instance:
7:          Call postDelayed() method of handler object with runnable and
            delay converted to Long as arguments.
8:          Call Algorithm 10.
9:          Assign the Runnable instance to runnable variable.
10:     Call super.onResume().
```

As our application can run in the background for the LocationService implementation, scheduler service also runs in the background after closing the application and keeps the distance calculation running in order to notify smartwatch if user’s presence in safe location.

4.4.2 Smartphone Module for Model 2

In model 2, the system keeps track of the distance between smartphone and the smartwatch by implementing haversine formula which has also been used in android application module. The system uses Algorithm 12 to listen for any event coming the android application. In case of, the information regarding if user is in safe location or not, it updates the variable “tag” accordingly. For events related to smartphones GPS information that is send by the android application in every 2 seconds, system updates “phnLat” variable for

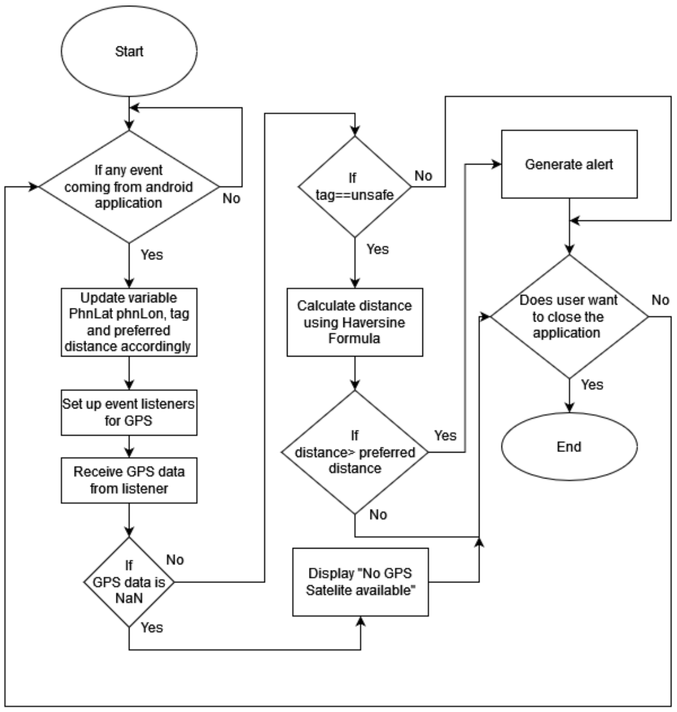

Start
If any event coming from android application
No
Yes
Update variable PhnLat phnLon, tag and preferred distance accordingly
Set up event listeners for GPS
Receive GPS data from listener
If GPS data is NaN
No
Yes
Display "No GPS Satelite available"
If tag==unsafe
No
Yes
Calculate distance using Haversine Formula
If distance> preferred distance
Yes
No
Generate alert
Does user want to close the application
No
Yes
End

Figure 11 : Workflow of Smartwatch Module of Model 2

storing smartphone’s latitude value and “phnLon” variable for storing smartphone’s longitude value accordingly. And the event related to preferred distance is updated for variable “preferredDistance”. For the default value of preferred distance, it is set as 20 feet. And the default value of the “tag” variable is set as “unsafe”.

Algorithm 12: Event Listener for model 2

```
1:   Define a global function called "GB" that takes an event as input.
2:   Check the type of the event by evaluating the "t" property of the event object.
```

```
3:  **IF** the type of the event is "enter", **THEN**
4:          Set the value of the "tag" variable to "safe".
6:  **IF** the type of the event is "exit", **THEN**
7:          Set the value of the "tag" variable to "safe".
8:  **IF** the type of the event is starts with "latitude", **THEN**
9:          Extract the numeric value from the event starting from the 10th character.
10:         Convert it to a floating-point number.
11:         Set the value of the "phnLat" variable to this value.
12: **IF** the type of the event is starts with "longitude", **THEN**
13:         Extract the numeric value from the event starting from the 10th character.
14:         Convert it to a floating-point number.
15:         Set the value of the "phnLon" variable to this value.
16: **IF** the type of the event is starts with "radius", **THEN**
17:         Extract the numeric value from the event starting from the 7th character.
18:         Set the value of the "rad" variable to this value.
19: **ELSE**
20:         Parse the "t" property of the event as an integer
21:         Store it in the "preferredDistance" variable.
22: **END IF**
```

The system doesn’t start working on the GPS data fetching algorithm or distance calculation algorithm unless the system receives the latitude and longitude value of the smartphone. The reason behind this is that without the latitude and longitude value of the smartphone, the system can’t calculate the distance using haversine formula. If the system starts working without these values, it will generate wrong result. Moreover, the energy and computational cost of fetching the GPS data for the smartwatch will also be saved for this error scenario.

Once the system receives the latitude and longitude value of the smartphone, it starts fetching smartphone’s GPS information. For Bangle.js 2, there is a build-in event listener which fetches current GPS data. The system calls the event listener by Bangle.on('GPS', onGPS) command. By the “GPS” parameter, it calls for getting the GPS data. When GPS data is available it calls the callback function “onGPS” sending the GPS value to the function’s parameter. System also calls Bangle.setGPSPower(1, 'app') function to set the power level of the GPS receiver. The first argument, 1, sets the power level to the highest available value. This is used when the GPS receiver needs to obtain a strong GPS signal, for example when the device is being used outdoors or in a location with weak GPS signal. The second argument, "app", specifies that the power level setting is only applied to the current application running on the device. This is useful when the GPS receiver is only needed for a specific task within the application and can be powered down when the task is complete, to conserve battery life.

We have implemented the distance calculation algorithm using haversine formula in the call back function “onGPS”. When GPS data is available, the data is sent to “onGPS” function as parameter. The system runs Algorithm 13 to calculate distance using Algorithm 9. In this algorithm, the system checks if the fix object contains valid GPS information or not. If the latitude value is not a number, it will mean that Bangle.js 2 couldn’t connect to any satellite to get its GPS information. In this scenario, the system doesn’t proceed to calculate the distance. The listener keeps fetching the GPS information. When Bangle.js 2 connects with satellite and sends valid GPS data, the fix object has a fix property that is truthy (i.e., not null, undefined, false, 0, NaN, or an empty string). When the GPS data is valid and the user is in an unsafe location, the system extracts latitude and longitude information from the fix and saves them in watchLat and watchLon variables. After that, the system calls Algorithm 9 to apply haversine formula in order to calculate distance and generates alert if condition is fulfilled.

Algorithm 13: onGPS

```
1:  Define a function called onGPS that takes a parameter called fix.
2:  IF fix.lat is not a number, THEN
3:          Display "No GPS Satellite available" on screen.
4:  IF fix.fix is true and tag is “unsafe”, THEN
5:          Set a variable called “watchLat” to the value of fix.lat rounded to 6 using
6:          the toFixed method.
```

```
7:          Set a variable called "watchLon" to the value of fix.lon rounded to 6 using
            the toFixed method.
8:          Call Algorithm 9 passing watchLat, watchLon, phnLat and phnLon to
            calculate distance.
9:          IF distance is greater than preferredDistance THEN
10:             Display prompt "Your Phone is not nearby." with title "Alert!".
11:             Call the beep function of the Bangle object 5 times.
12:             Call the buzz function of the Bangle object 5 times.
13:         END IF
14:     END IF
```

This algorithm doesn’t require any additional service to call it repeatedly to get constant monitoring. The event listener does this work on its own and keeps generating GPS data and calls the callback function (Algorithm 13) repeatedly. Therefore, this system provides a constant monitoring service to provide protection to the smartphone device from getting lost.

4.5 Model 3 (Integrated with GPS and RSSI of smartwatch and Android application)

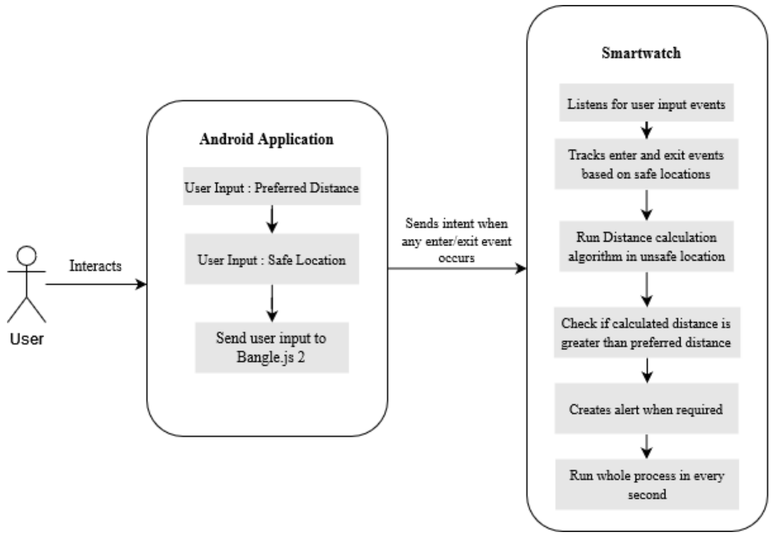

User
Interacts
Android Application
User Input : Preferred Distance
User Input : Safe Location
Send user input to Bangle.js 2
Sends intent when any enter/exit event occurs
Smartwatch
Listens for user input events
Tracks enter and exit events based on safe locations
Run Distance calculation algorithm in unsafe location
Check if calculated distance is greater than preferred distance
Creates alert when required
Run whole process in every second

Figure 12 : Model 3 Architecture

The two key functionalities of our proposed system is to calculate distance between the smartphone and the smartwatch and tracking if user is in a safe location or not. In the previous two models, we have used the android application module to keep track whether user is in a safe location and the smartwatch module to calculate distance between the smartphone and the smartwatch. In our model 3, we have implemented both key functionalities in the smartwatch module. The tracking of user's location to determine whether user is in a safe location is implemented by using the GPS module of smartwatch and have used Bluetooth module of the smartwatch to calculate between the smartphone and the smartwatch. For this model, we have used the android application only to take user input such as the preferred distance value and the safe location information. For this model, the system can save 1 safe location using shared preferences.

4.5.1 Android Application Module for Model 3

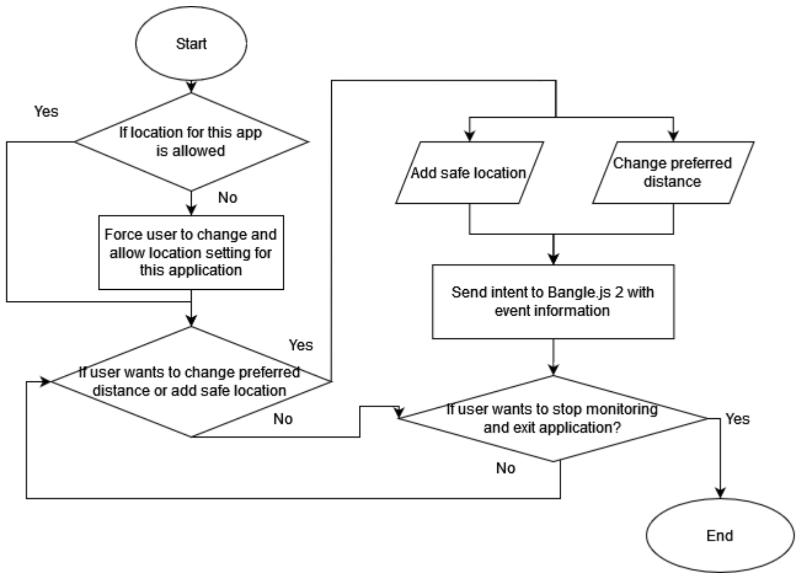

Start
If location for this app is allowed
Yes
No
Force user to change and allow location setting for this application
If user wants to change preferred distance or add safe location
Yes
No
Add safe location
Change preferred distance
Send intent to Bangle.js 2 with event information
If user wants to stop monitoring and exit application?
Yes
No
End

Figure 13 : Workflow of Android Application Module for Model 3

For our preferred distance value and safe location's name and radius value, user needs to provide user's preferred input. In bangle.js 2 there is no keyboard module through which the user can provide their input in the smartwatch's screen. Therefore, we have used the android application interface to take user input regarding preferred distance and safe location.

While adding safe location, the system needs to get the current location's latitude and longitude to set as safe location's latitude and longitude. Therefore, the system first uses Algorithm 1 to ensure the location access of the application to be allowed. After that, it uses Algorithm 3 to get the current location's information. After receiving the location information, it sends the latitude, longitude, and radius value to bangle.js 2 as intent using the Algorithm 2. For this model, the system can save 1 safe location using shared preferences. If the user try to input more than one safe location, a Snackbar message will be displayed in the Android application mentioning that maximum capacity exceeded.

4.5.2 Smartwatch Module for Model 3

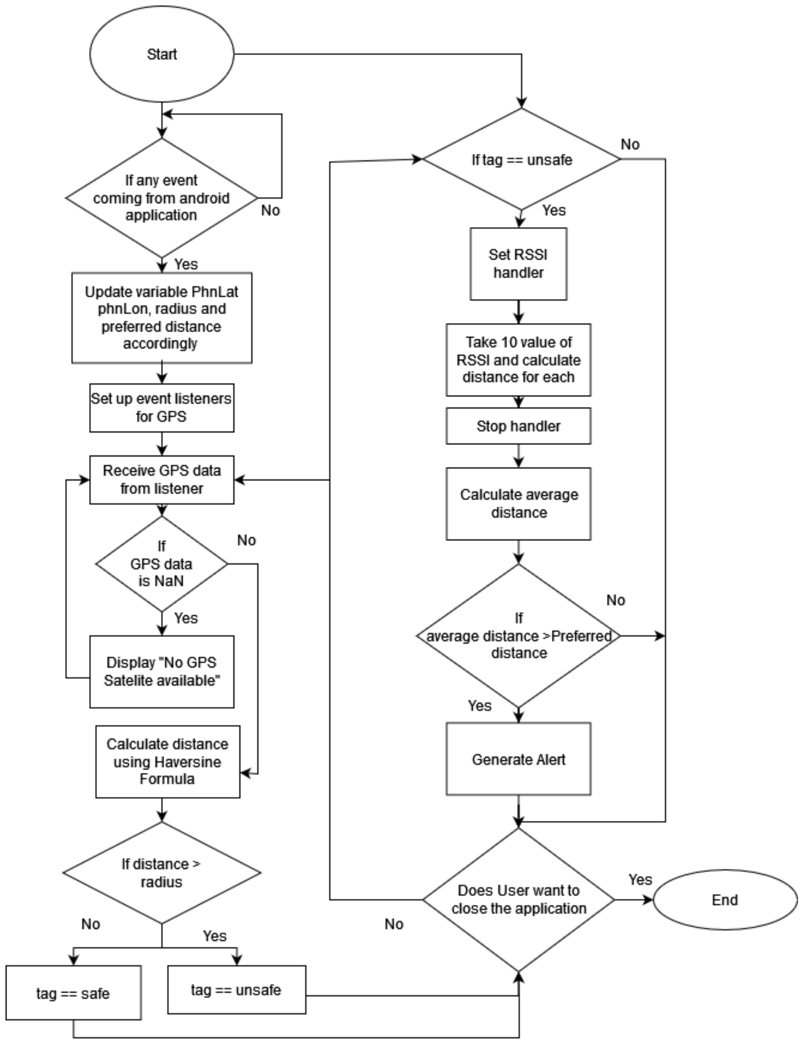

Start
If any event coming from android application
No
Yes
Update variable PhnLat phnLon, radius and preferred distance accordingly
Set up event listeners for GPS
Receive GPS data from listener
If GPS data is NaN
No
Yes
Display "No GPS Satelite available"
Calculate distance using Haversine Formula
If distance > radius
No
Yes
tag == safe
tag == unsafe
If tag == unsafe
No
Yes
Set RSSI handler
Take 10 value of RSSI and calculate distance for each
Stop handler
Calculate average distance
If average distance >Preferred distance
No
Yes
Generate Alert
Does User want to close the application
Yes
No
End

Figure 14 : Workflow of Smartwatch module for Model 3

In this module, the system uses haversine formula to keep track if user is in safe location and RSSI formula to calculate distance between the smartphone and the smartwatch. At the beginning, the system starts the distance calculation algorithm before receiving the information about the user's presence in an unsafe location. By default, the system keeps the variable "tag" to "unsafe" and starts the distance monitoring process immediately. Simultaneously, it also starts the process of checking if the user is in a safe location.

The system performs the distance calculation using RSSI which is performed by running Algorithm 7. When user is in an unsafe location, Algorithm 7 calculates the distance between smartphone and smartwatch using RSSI formula and generates alert if distance is found greater than the preferred distance. By default, the preferred distance value is set 20 feet. However, when user changes the value through android application, the value will be updated through the event listener algorithm.

Along with the distance calculation algorithm, for the checking process about user's presence in safe location, system runs a listener using Algorithm 12 to receive intent from the android application regarding user input. Unless the user adds a safe location and event listener receives information about latitude, longitude, and radius of the safe location, it doesn't start the tracking process. After receiving the information, the system starts its work to track if the user is in a safe location or in an unsafe location using the haversine formula. Similar to Model 2, the system needs to get current GPS information to implement the haversine formula. Therefore, the system uses the event listener Bangle.on('GPS', onGPS) to gather current GPS information. When GPS data is available, the callback function "onGPS" is called using the GPS data as parameter. The system runs Algorithm 14 to calculate distance using Algorithm 9. In this algorithm, system check that if the parameter GPS object carries valid GPS information. Upon finding of valid GPS information, the system extracts latitude and longitude information from the fix parameter and saves them in watchLat and watchLon variables. Then, the system calls Algorithm 9 to apply haversine formula to calculate distance and update the "tag" variable accordingly.

Algorithm 14: onGPS for Model 3

```
1:    Define a function called onGPS that takes a parameter called fix.
```

```
2:   IF fix.lat is not a number, THEN
3:          Display "No GPS Satellite available" on screen.
4:   IF fix.fix is true, THEN
5:          Set a variable called “watchLat” to the value of fix.lat rounded to 6 using
6:          the toFixed method.
7:          Set a variable called “watchLon” to the value of fix.lon rounded to 6 using
            the toFixed method.
8:          Call Algorithm 9 passing watchLat, watchLon, safeLocLat and
            safeLocLon to calculate distance.
9:          IF distance is greater than radius THEN
10:                Set value of tag to “unsafe”.
            ELSE
                   Set value of tag to “unsafe”.
13:         END IF
14:  END IF
```

This callback function gets called repeatedly and updates the global variable tag accordingly. Based upon the current value of tag, the system runs the Algorithm 7 side by side to provide a constant monitoring process to prevent the phone from getting lost by giving alert to user when necessary.

Chapter 5

5 Evaluation, Experimental Results and Discussion

The proposed system from chapter 4 is tested on the development platform from section 4.1. This chapter will provide our findings and analysis from the tests carried out to develop the smartphone loss prevention system. Section 5.1 will give an overview of how our evaluation will work. Section 5.2 will present the user interface testing of implemented system on development platform. Section 5.3 will provide the findings and tests performed on 3 different proposed models of our work and Section 5.4 will present a comparative result analysis of all the approaches we used in our research.

5.1 Evaluation Setup

To evaluate the detection performance of our models, we conducted experiments that simulated different smartphone loss scenarios in 5.3. We have conducted these experiments in various locations and different infrastructures to check our model's performance under different circumstances. We have also considered using different preferred distances. In these experiments, we performed series of departure events where a departure event indicates that the user is leaving the smartphone and user has exceeded the preferred distance from the smartphone.

5.1.1 Evaluation Platform

We have used Bangle.js 2 smartwatch and Samsung Galaxy S22 smartphone as our evaluation platform. We have installed the appropriate android application from each model before conducting a test. And the settings for the smartphone have been set as Table 4.

Table 4 : Settings of Smartphone

Bluetooth	On (paired and connected with Bangle.js 2)
Location of our Android Application	Allowed (While using the app)

We have installed the appropriate watch face incorporated with our system for smartwatch from each model in Bangle.js 2 before conducting a test. After that, we set that watch face as default. And the settings for the smartphone have been set as Table 5.

Table 5 : Setting of Bangle.js 2

BLE	On
Bluetooth Make connectable	Yes

5.1.2 Experiment Scenario

For each manual evaluation test, our aim is to see if our model generates alert when we go beyond the preferred distance threshold value and for this, the following steps are followed-

1. The user sets the preferred distance and set a safe location before initiating the test.

2. In an appropriate test location based on the test, the user places the smartphone in a specific place and starts to move away from the smartphone.

3. The user keeps walking further away from the smartphone with an aim to go beyond the preferred distance value using a measuring tape to keep track of the distance between smartphone and smartwatch.

4. While moving further away from the smartphone, observes the smartphone's screen to see if any alert generates.

5. When an alert is generated, we collect the distance value where the model generated the alert and where it should have generated the alert in actual scenario.

5.1.3 Evaluation Metrics

We have collected and stored data for each test. We used the Jupiter Notebook platform to process and generate performance evaluation metrics our models in Python 3.7 language. We used the NumPy, Matplotlib, and Seaborn libraries are used to create graphs for the experimental findings. To evaluate our model's performance, we have used the following metric-

5.1.3.1 Accuracy

The model evaluation criterion called classification accuracy is widely used and important. Accuracy is an essential evaluation metric, particularly in binary classification models, where the goal is to classify instances into one of two classes. The model provides passed/ failed result for each case in a test dataset, and the results are then compared to the known labels for the test set examples. Accuracy measures the proportion of correctly classified instances by a model, and it is calculated by dividing the total number of correctly classified instances by the total number of instances.

$$Accuracy = \frac{Number\ of\ passed\ tests}{Number\ of\ total\ tests} \times 100 \qquad [46]$$

5.2 User Interface Test

Our proposed models have been implemented in the smartwatch as a watch face and in the smartphone as an android application.

5.2.1 Smartwatch Interface Test

The name of the watch face is seen to be set as "Anti-loss Smartphone". When it is set and opened, it opens successfully within an average time delay of 1 second. It shows the battery percentage on top. Along with that, it also shows real- time date and time on the screen. It also shows if the user is in an unsafe or safe location. For unsafe locations, it displays "unsafe" and for safe locations, it displays "safe" [Figure 15]. A test was performed to see if the watch receives and updated value correctly. While being in an unsafe location, if a safe zone is added for that current location, it is updating the value in watch app's display to "safe" within time delay of average 6.23 seconds.

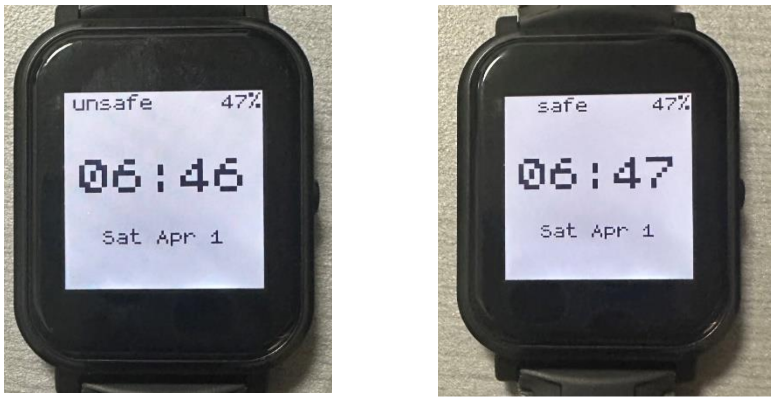


Figure 15 : Interface of Model 1 with safe and unsafe location mode

.

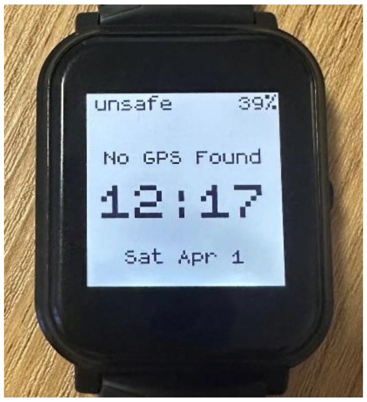
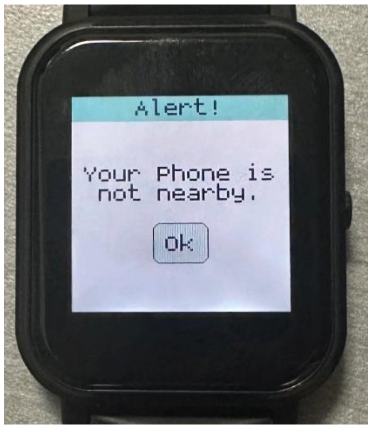


Figure 16: Interface of Model 2 and Model 3 when GPS satellite is not connected.

When the watch is taken out beyond the set preferred distance value, the watch performs following things-

1. Turn on the screen light.
2. Show an alert with a message "Your phone is not nearby".
3. Create a buzzing sound.
4. Create vibration.

After clicking the "Ok" button, it goes to the default screen with battery percentage, date and time information showing screen.



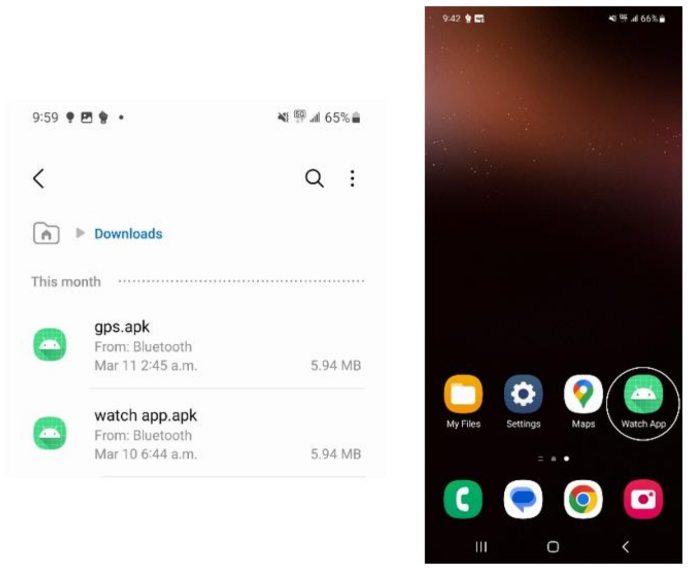
Alert!

Your Phone is not nearby.

Figure 17: Alert notification on Smartwatch

5.2.2 Android Application Interface Test

The apk file for android application is installed in the Samsung Galaxy S22 successfully. Upon successful installation, the application is seen in the home screen as “Watch App”.



gps.apk
From: Bluetooth
Mar 11 2:45 a.m. 5.94 MB

watch app.apk
From: Bluetooth
Mar 10 6:44 a.m. 5.94 MB

Figure 18 : Android Application on Home Screen

When the application is opened, it asks for user’s permission regarding allowing the application to use the device’s location. After allowing the location, the location access is changed to “Deny” going to settings of the “Watch App” to test if the application again asks for the location access again. The application does again ask for allowing the location for the application.

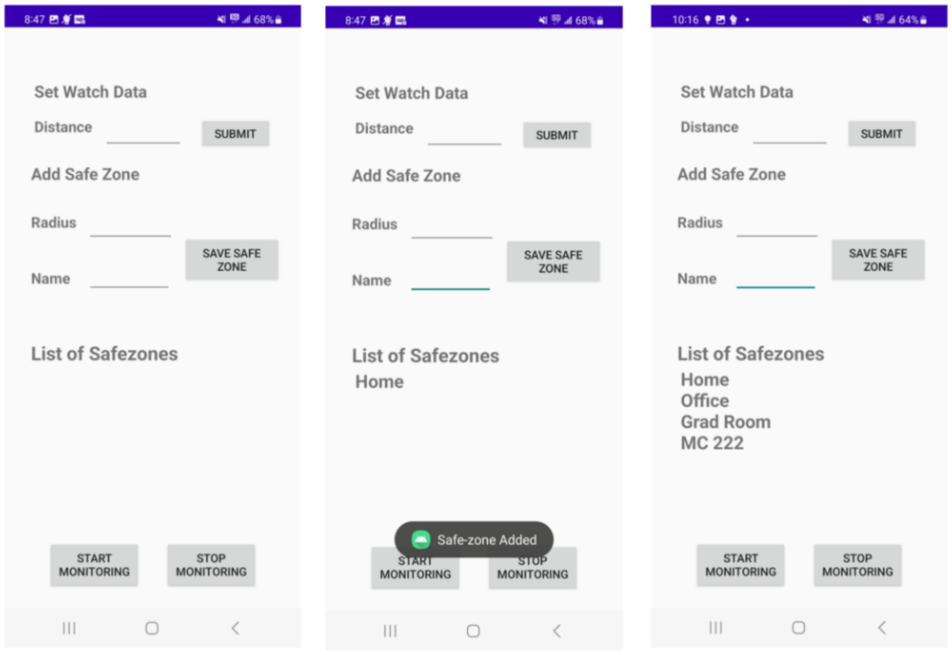


Figure 19 : Watch App Interface

The application has several sections where it takes inputs and display saved inputs-

1. **Set Watch Data:** In this field, the user is able to input preferred distance value successfully.
2. **Add Safe Zone:** In this field, the user can add a new safe zone. The application takes safe location's name and radius successfully. And shows a message when the location is added successfully.
3. **List of Safe zones:** It displays the list of all the saved safe zones. To test the data persistency, the application was closed after saving a safe zone and again re-opened the application and found that all the previously saved safe zones are seen in the list of safe zones.
4. **Start/Stop Monitoring:** User can start the tracking of user's position in safe location by clicking the "Start Monitoring" button and stop the tracking by clicking

the “Stop Monitoring” button successfully. These buttons are not visible for the application for model 3 as the monitoring doesn’t take place in android application.

5.3 Smartphone Loss Prevention System Test

We have evaluated 3 of our models in different environments and observed their performances.

5.3.1 Distance Calculation and Generating Alert Test

Our aim is to see if our model generates alert when we go beyond the preferred distance threshold value. When we go beyond the preferred distance threshold value, it is expected that the system will generate an alert. In this test, our objective is to confirm whether the system indeed triggers the alert as intended (a true positive) when the user go beyond the preferred distance threshold. Conversely, if the system generates an alert when the user has not exceeded the threshold, it would be considered a false positive. Therefore, our primary objective in this test is to observe the accuracy of our models and monitor the occurrence of true positives and false positives.

5.3.1.1 Open Spaces

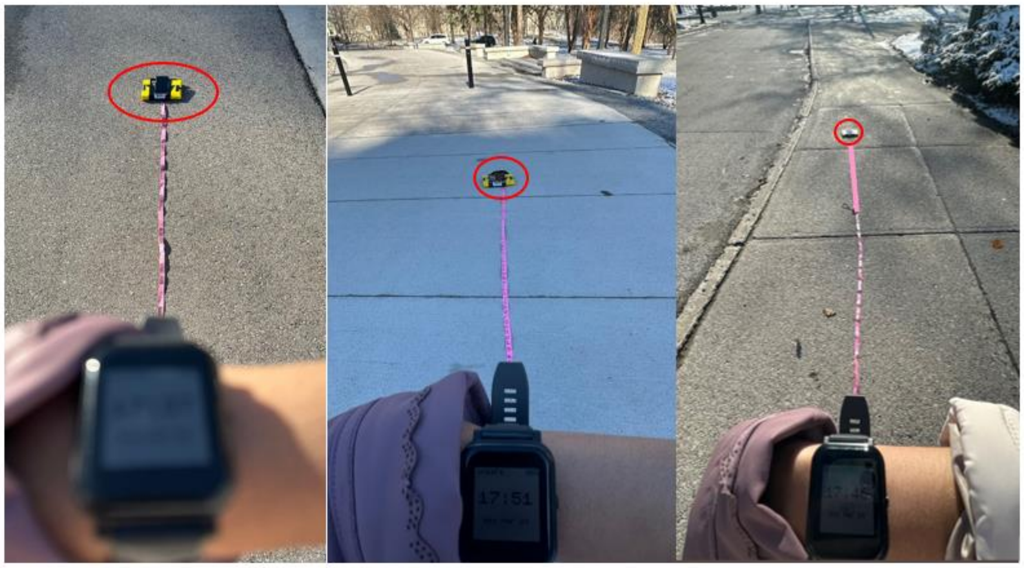


Figure 20: Open spaces evaluation setup

We have tested our models in 4 open space locations and for each location we have performed tests to check how accurately our models calculate the distance between smartphone and smartwatch and generate alert. We have tested this performance of our system with different preferred distances. For each preferred distances we have performed 20 tests in each location and evaluated their performance by their accuracy [Table 7]. This test has been performed for unsafe locations as this function doesn't perform in safe locations. In total, we have collected 560 test data in open space locations.

Table 6 : Actual Distance vs Calculated Distance of Model 2 in Open space

Actual Distance (feet)	Calculated Distance using Smartwatch's GPS(feet)	Calculated Distance using iPhone 14 pro's GPS value (feet)
5	119.6	29.6
10	136.4	57.18
15	157.8	82.56
20	194.7	104.23

The reason for excluding Model 2 is that its distance calculation has not performed well enough to provide satisfactory accuracy due to the weak performance of the GPS module of smartwatch. Therefore, We can see in Table 6 that the actual distance value and calculated distance value has big difference which leads to make all the results true negative. As a resource-constrained device, our smartwatch's GPS module is weak and so we have used another device's GPS module which is the latest and more powerful than our smartwatch's GPS module to evaluate the performance of our model 2. We have used

Table 7 : Accuracy of Model 1, and Model 3 for different Preferred Distances for open spaces

Preferred Distance (feet)	Accuracy of Model 1	Accuracy of Model 3
5	100%	100%
8	98%	95%
10	95%	97%
12	95%	96%
15	91%	93%
18	94%	90%
20	90%	91%

iPhone 14 pro's GPS module's value instead of our smartwatch's GPS module's value in our model 2 and conducted 40 tests to evaluate the distance calculation performance of model 2. The evaluation result of using a powerful GPS result has performed better than the smartphone's GPS. Nonetheless, the result is not promising enough to provide a reliable result for our system. The model 2 which uses haversine formula to calculate the distance between the smartphone and smartwatch has not performed reliably enough to provide convincing accuracy for our system.

Table 8: Number of True Positives and False Positives for Model 1, Model 2, and Model 3 in Open Spaces

Model Name	True Positive	False Positive
Model 1	530	30
Model 2	0	560
Model 3	529	31

From Table 9, we can see that best performing model in terms of distance calculation and generating alert among all the models is Model 1 which has been integrated with RSSI of smartwatch and Android application.

Table 9 : Average Accuracy of Model 1 and Model 3 for open spaces

Model	Average Accuracy
Model 1	94.64%
Model 3	94.46%

5.3.1.2 Close Spaces without Environmental Interference

We have evaluated our models in 4 close space locations and for each location, we have performed tests to check if the watch generates alert when the watch is beyond preferred distance value away from the smartphone. For these tests, no environmental Interferences (wall, big furniture's, tree) has been placed between the smartphone and smartwatch. We have evaluated the performance of our system with different preferred distances. For each preferred distance we have performed 20 tests in each location and evaluated their performance by their accuracy. We have collected 560 test data for close space locations with no environmental interferences.

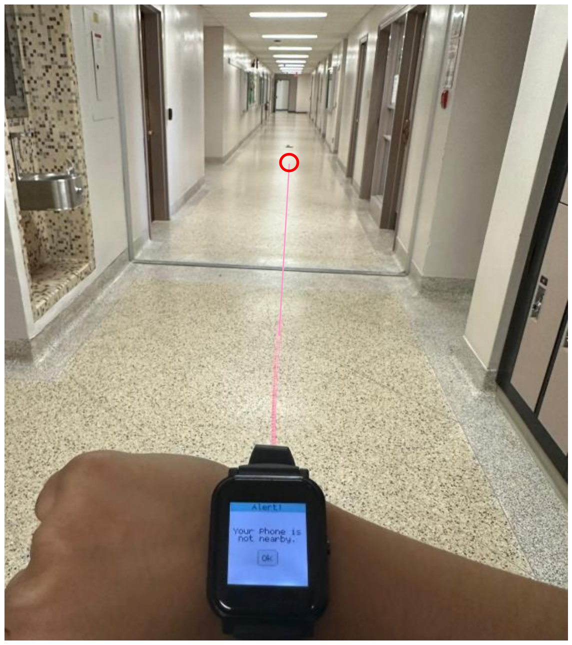


Figure 21: Closed spaces without environmental interference evaluation setup

For model 2, it is essential for the smartwatch to have GPS satellite connection to achieve GPS information. However, in closed spaces the smartwatch fails to connect to GPS satellite. The reason behind this is that IoT resource-limited devices like smartwatches rely on signals received from GPS satellites to determine their location and they require a clear line of sight to the sky to be received properly. These signals can be weakened or completely blocked by physical barriers such as buildings, walls, and ceilings. Therefore, areas with limited access to the sky will have a harder time receiving GPS signals and

Table 10 :Actual Distance vs Calculated Distance of Model 2 in Closed space without environmental interference

Actual Distance (feet)	Calculated Distance using Smartwatch's GPS(feet)	Calculated Distance using iPhone 14 pro's GPS value (feet)
5	148.4	34.64
10	175.9	59.4
15	197.2	91.47
20	220.1	109.43

determining their location accurately. So, for model 2 we tested it by keeping the smartwatch near windows or balconies where the watch could connect to GPS satellite and get GPS value. In the Table 11, the performance of the models can be seen for different preferred distances.

Table 11 : Accuracy of Model 1, and Model 3 for different Preferred Distances for Closed Spaces without Environmental Interferences

Preferred Distance (feet)	Accuracy of Model 1	Accuracy of Model 3
5	100%	100%
8	100%	99%
10	99%	100%
12	100%	97%
15	95%	91%
18	92%	93%
20	94%	92%

Similar as Open space evaluation, we have used iPhone 14 pro's GPS module's value instead of our smartwatch's GPS module's value in our model 2 and conducted 40 tests to evaluate the distance calculation performance of model 2 for closed space without environmental interference [Table 10]. The model 2 also has not performed reliable enough to provide convincing accuracy for our system in closed space without environmental interference.

Table 12: Number of True Positives and False Positives for Model 1, Model 2, and Model 3 in Closed Spaces without Environmental Interferences

Model Name	True Positive	False Positive
Model 1	544	16
Model 2	0	560
Model 3	538	22

From Table 13, we can see that best performing model in terms of distance calculation and generating alerts among all the models is Model 1.

Table 13 : Average accuracy of Model 1 and Model 3 for Closed Spaces without Environmental Interferences

Model	Average Accuracy
Model 1	97.14%
Model 3	96.07%

5.3.1.3 Close Spaces with Environmental Interference

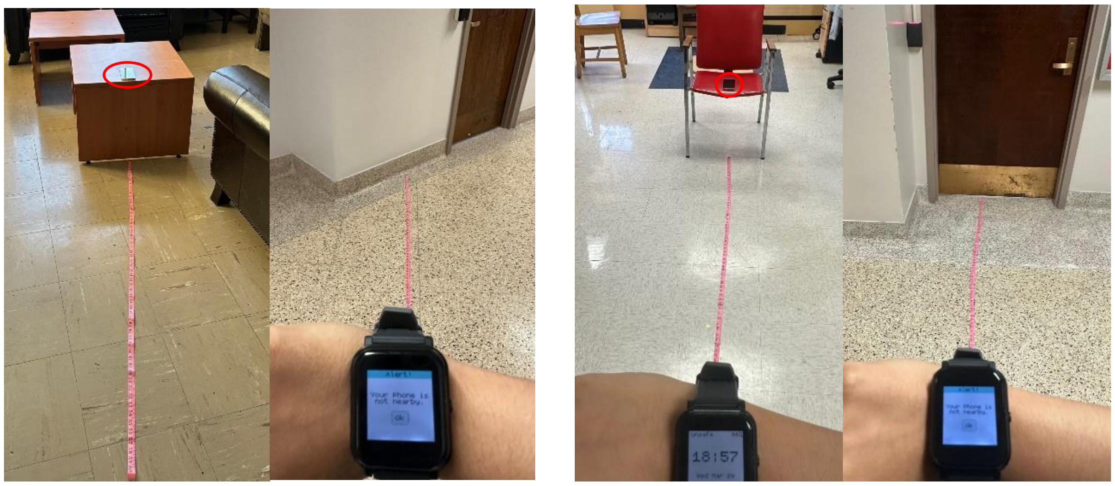


Figure 22: Closed spaces with wall/door as environmental interference evaluation setup in 2 locations

Our models have been tested in 4 close space locations and for each location, we have performed tests to check if the watch generates alert when the watch is taken beyond preferred distance value away from the smartphone. For these tests, we have placed the smartphone and smartwatch in such positions where there will be one or multiple walls

Table 14: Actual Distance vs Calculated Distance of Model 2 in Closed space with environmental interference

Actual Distance (feet)	Calculated Distance using Smartwatch's GPS(feet)	Calculated Distance using iPhone 14 pro's GPS value (feet)
5	139.8	37.53
10	180.45	64.24
15	201.33	85.19
20	217.09	103.65

between them. We have tested the performance of our system with different preferred distances. For each preferred distance we have performed 20 tests in each location and evaluated their performance by their accuracy. We have collected 560 test data for close space locations with walls as environmental interferences. For model 2 we tested it by keeping the smartwatch near windows or balconies where the watch could connect to GPS satellite and get GPS value.

Table 15 : Accuracy of Model 1, Model 2, and Model 3 for different Preferred Distances for Closed Spaces with Environmental Interferences

Preferred Distance (feet)	Accuracy of Model 1	Accuracy of Model 3
5	100%	100%
8	98%	97%
10	99%	100%
12	96%	98%
15	97%	94%
18	93%	90%
20	90%	95%

Same as previous evaluation, we have used iPhone 14 pro's GPS module's value instead of our smartwatch's GPS module's value in our model 2 and conducted 40 tests to evaluate the distance calculation performance of model 2 for closed space with environmental interference [Table 14]. The model 2 also has not been reliable enough to provide convincing accuracy for our system in closed space with environmental interference.

Table 16: Number of True Positives and False Positives for Model 1, Model 2, and Model 3 in Closed Spaces with Environmental Interferences

Model Name	True Positive	False Positive
Model 1	538	22
Model 2	0	560
Model 3	539	21

From Table 17, we can see that best performing model in terms of distance calculation and generating alerts among all the models is Model 3.

Table 17 : Average accuracy of Model 1 and Model 3 for Closed Spaces with Environmental Interferences

Model	Average Accuracy
Model 1	96.07%
Model 3	96.25%

5.3.2 Safe Location Test

As safe locations are mostly closed spaces such as home, office etc., we have tested our models in 4 closed space locations and for each location we have performed tests to check if the watch tracks whether the user is situated in the safe locations or not. To generate the test, we set the safe locations standing in the center of the safe location and providing radius value. After that, we started to move away from the center point wearing the smartwatch. Then we go beyond the radius value measuring the distance by measuring tape. When the distance between the center point and current position is more than radius value, we monitored the tracking of safe locations variable in the smartphone screen to change from

Table 18 : Accuracy of Model 1 and Model 2 for different Radius for Closed Spaces

Preferred Distance (feet)	Accuracy of Model 1	Accuracy of Model 2
5	86%	84%
10	90%	89%
15	87%	81%
20	94%	90%
25	92%	92%
30	95%	89%

"safe" to "unsafe". We have tested the performance of tracking safe locations with different radius values. For each radius values, we have performed 10 tests in each location and evaluated their performance by their accuracy [Table 18]. In total, we have collected 280 test data in closed space locations. Due to the poor performance of the GPS of smartwatch, our model 3 has not performed well enough to provide closer to accurate results regarding safe location tracking. Even in safe location, model 3 detected that user is in 'Unsafe' location for all the cases. The reason behind this discrepancy is that the calculated radius distance using haversine formula is not close to the actual radius at all which leads our model 3 to make the wrong decision regarding safe distance tracking. Therefore, the model 3 has not been reliable enough to provide convincing accuracy for our system regarding safe location tracking.

Table 19: Number of True Positives and False Positives for Model 1 and Model 2 for Safe Location Tracking

Model Name	True Positive	False Positive
Model 1	254	26
Model 2	245	35

From Table 20, we can see that best performing model is Model 1 in terms of tracking safe locations.

Table 20 : Average accuracy of Model 1 and Model 2 for tracking Safe Locations

Model	Average Accuracy
Model 1	90.71 %
Model 2	87.50 %

5.3.3 Battery Life Test

5.3.3.1 Smartphone’s Battery Consumption Test

The battery life of the smartphone has been evaluated for Model 1 and Model 2 where the Watch App runs continuously in the background with Bluetooth connection enabled. As for model 3, the application doesn’t run in the background as it is only used for providing user input. Therefore, Model 3 hasn’t been evaluated for smartphone’s battery consumption test.

In order to assess the battery consumption of our application, we conducted two types of comprehensive tests. The first test involved monitoring the smartphone's battery life with no applications running for a continuous 18-hour period, simulating idle usage. The second test focused on the battery consumption of the smartphone while running only our Android application for a continuous 18-hour period.

Baseline Battery Discharge:

We have conducted 2 similar tests to measure the smartphone's battery discharge rate with no application running. At the start of each test, the smartphone has been fully charged to

100%. Each test was conducted over an 18-hour period. During this time frame, we monitored and recorded the remaining battery percentage at the end of each hour. This served as our baseline reference for understanding the natural rate of battery consumption under normal operating conditions.

Application-Specific Battery Consumption:

The second test focused on assessing the battery life impact of our application running exclusively on the smartphone. At the start of each test, the smartphone has been fully charged to 100%. After that, we opened the application and started the monitoring of safe location tracking. After starting the monitoring process, we closed the application and the application kept running in the background. All other running applications on the smartphone has been closed to minimize interference. During the whole testing time, no other application has been used and the smartphone hasn't been used for any other activity.

To assess the battery performance of our Model 1 and Model 2 over an 18-hour period by monitoring the battery discharge percentage at hourly intervals, we have conducted 3 battery consumption tests for Model 1 and 3 battery consumption tests for Model 2. At the end of each hour, the remaining battery percentage on the smartphone has been recorded. We conducted each test for 18 hours.

Figure 23 illustrates Time vs Smartphone's Battery Discharge Rate graph for a variety of tests for smartphone. These two types of tests allowed us to compare the battery performance of the smartphone in its idle state (no application running) vs its performance while only our application was in use. The result of Model 1 reflects an efficient performance, as it demonstrates minimal battery consumption compared to the baseline, indicating that our application has a minimal battery overhead on the smartphone's battery life. For Model 2, which involves continuous location fetching and periodic intent transmissions to the smartwatch, we noted a slightly higher but still commendable battery consumption rate compared to the baseline. During a few tests, the smartphone was occasionally in motion for certain periods, simulating real-world usage scenarios. However, it did not have any significant impact on the observed battery consumption.

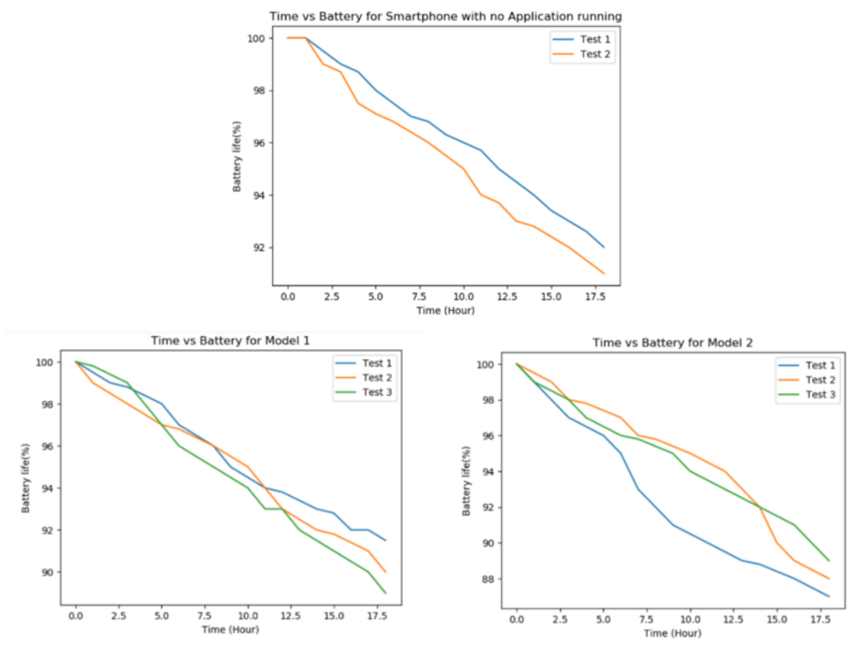


Figure 23: Baseline Battery Discharge vs Application-Specific Battery Consumption for Smartphone

5.3.3.2 Smartwatch's Battery Consumption Test.

In order to assess the battery consumption of our application on smartwatch's end, we conducted two types of comprehensive tests similarly as 5.3.3.1. The first test monitors the smartwatch's battery life with no applications running for a continuous 12-hour period, simulating idle usage. The second test focused on the battery consumption of the smartwatch while running only our application for a continuous 12-hour period.

Baseline Battery Discharge:

We have conducted 2 similar tests to measure the smartwatch's battery discharge rate with no application running. We aim to establish a baseline reference for battery consumption by monitoring a smartwatch devoid of any active applications. At the beginning of each test, the smartwatch has been fully charged to 100%. Over a 12-hour period, we diligently tracked and recorded the remaining battery percentage at the end of each hour.

Application-Specific Battery Consumption:

The second test focused on assessing the battery life impact of our application running on the smartwatch. The battery life of the smartphone has been evaluated for Model 1, Model 2, and Model 3 where the watch face with our proposed system installed and Bluetooth connection enabled. We conducted two types of power consumption tests for each model-

1. Monitoring battery consumption when user is in safe location.
2. Monitoring battery consumption when user is not present in safe location.

In total, we have conducted 18 tests for all 3 models.

1. Model 1
   - I. When user is in safe location – 3 tests.
   - II. When the user is not in a safe location – 3 tests.
2. Model 2
   - I. When user is in safe location – 3 tests.
   - II. When the user is not in a safe location – 3 tests.
3. Model 3
   - I. When user is in safe location – 3 tests.
   - II. When the user is not in a safe location – 3 tests.

At the beginning of each test, the smartwatch has been fully charged to 100%. We conducted each test for 12 hours and monitored the battery percentage every hour. During the whole testing time, no other application has been used and the smartwatch hasn’t been used for any other activity.

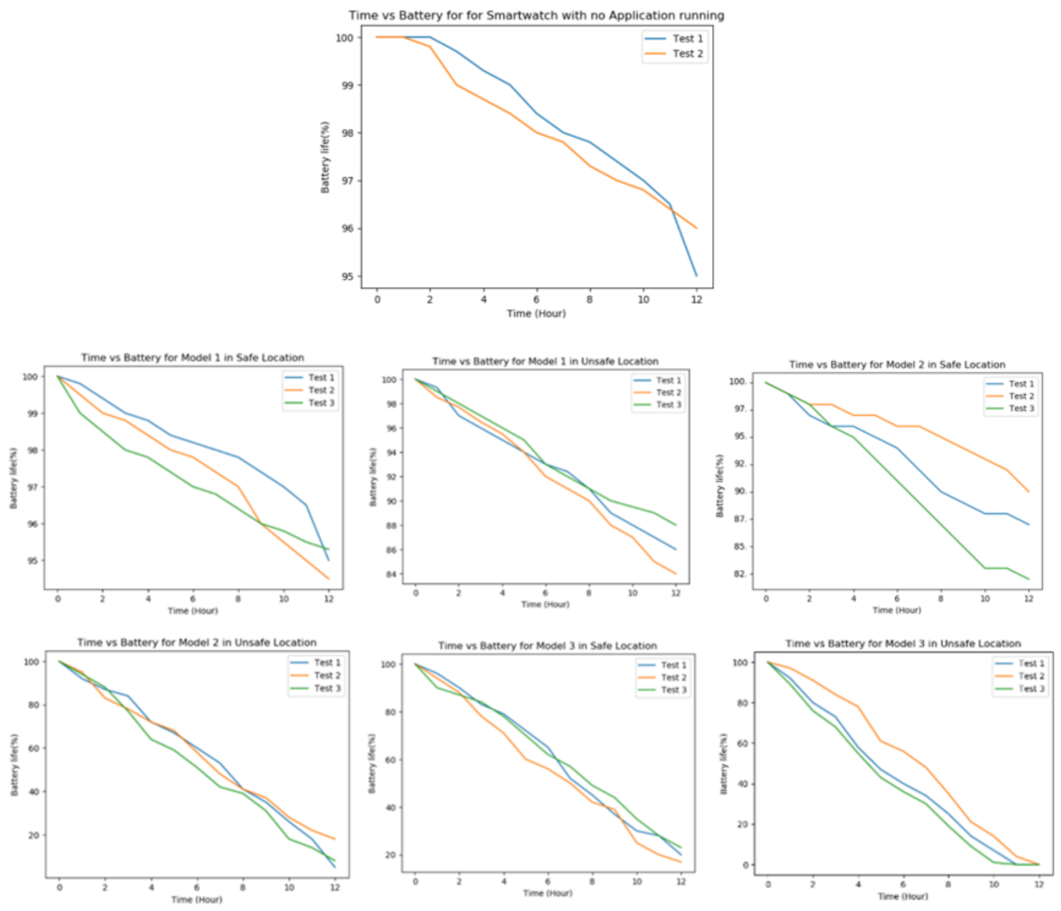


Figure 24: Baseline Battery Discharge vs Application-Specific Battery Consumption for Smartwatch

Figure 24 illustrates Time vs Smartwatch's Battery Discharge Rate graph for a variety of tests for smartwatch. For Model 1 in safe location, we can see that the power consumption rate is low compared to the baseline which indicates minimal battery consumption. For Model 1 in unsafe location, we can observe slightly higher consumption rate as the system continuously uses the Bluetooth module which consumes 0.75mA power [7]. The battery consumption of Model 1 in unsafe location, while slightly higher, remains within satisfactory limit.

For Model 2 in safe location, we have observed a slightly elevated consumption rate compared to the baseline as the system receives and process intent from the smartphone in every 2 seconds. However, for Model 2 in unsafe location and Model 3, the power consumption rate is much higher than the consumption rate of baseline, Model 1, and Model 2 in safe location. The reason behind this is that Model 2 (unsafe location) and Model 3 use GPS modules which consume a high amount of power. Model 1 only uses the Bluetooth module which consumes 0.75mA, whereas the GPS module that Model 2 in unsafe location and Model 3 use consumes 26mA power [7]. It is visible that the GPS module's power consumption rate is about 34.67 times more than the BLE module. Furthermore, the Model 3 in unsafe location has the highest consumption rate among all as it uses both BLE and GPS module. From Figure 24, we can observe that Model 1 is the most efficient model in terms of battery consumption for smartwatches.

5.3.4 Alert and Safe Location Update Response Time

We have conducted 20 tests for each model to check the response time of the smartwatch generating an alert when the distance between the smartphone and smartwatch is more than the preferred distance. Along with that, we have also conducted 20 tests to see how much time the smartwatch takes to update the safe location tracking information on the smartwatch's screen when user goes outside a safe location and enters in a safe location. Table 21 illustrates the average response time of the smartwatch generating alert and updating the safe location tracking information.

In terms of response time for alert generation, we can observe that Model 1 and Model 3 demonstrated exceptional performance. For both models, we set the wait time for each invocation to 1 second. However, if some callback function takes some time to execute, it might still be running when the next invocation is scheduled. This can lead to overlapping executions, and this cause the appearance of a shorter interval between executions.

Model 1 and Model 2 both use Android application of smartphone's module to perform safe location tracking. The slightly higher response time of updating safe location information may happen due to the processing time required for a series of consecutive tasks: monitoring safe location tracking on the smartphone, sending an intent to the smartwatch, processing the received intent in smartwatch's end, and displaying it correctly.

Moreover, we can observe that the response time for alert generation in Model 2 and the response time for updating safe location information in Model 3 took the highest amount of time among all. This may happen due to delay of establishing a connection with GPS satellites and collecting accurate GPS data.

Table 21 : Average Response time of Generating Alert and Safe Location Tracking update.

	Model 1	**Model 2**	**Model 3**
Average Response time for Alert Generation	0.47 Seconds	7.34 Seconds	0.59 Seconds
Average Response time for updating safe location information	6.23 Seconds	7.06 Seconds	7.23 Seconds

5.4 Overall Results and Findings

As per our research goals, we have evaluated our models to observe the performance of our models in terms of providing accurate response. Moreover, we have also conducted experiments to see the power consumption of our models and conducted the battery discharge percentage vs time test to evaluate our models. Also, evaluation regarding the response time of providing alert and updating safe location information has also been performed. Furthermore, we have observed how similar components provided different results in different modules. As shown in Table 22, the best performing model among these three methods is Model 1. Considering all the research goals, we can observe from the previous sections that –

- In terms of successfully calculating distance and generating alert accordingly Model 1 has provided highest accuracy, 96% in most of the test cases. We can observe that our Model 1 and Model 3 performed slightly better in closed spaces rather than open spaces. Outdoor environments are often more susceptible to interference compared to closed spaces due to interference from other wireless devices, natural obstacles, and atmospheric conditions. This interference can introduce noise into RSSI measurements, reducing accuracy for open space locations.

- For tracking safe location, our Model 1 has also achieved the highest accuracy, 90.66%.
- Along with that, Model 1 has been identified as the most efficient in terms of power consumption as its performance in power consumption has been found best among three models.
- Moreover, in terms of response time of the smartwatch generating alert and updating the safe location tracking information, Model 1 produced the best result.
- We have also observed at the performance of similar algorithms in different module such as the haversine algorithm was both implemented in smartwatch (Model 2 for distance calculation and Model 3 for safe location tracking) and smartphone (Model 2 safe location tracking) in separate models. Due to poor GPS accuracy of the Smartwatch, Model 2 couldn't perform well in calculating distance and generating alert and Model 3 couldn't perform well in tracking of user's presence in safe locations using haversine algorithm. Whereas this similar algorithm performed well in terms of safe location tracking in Model 2.

Overall, considering all of the evaluation parameters, Model 1 has outperformed Model 2 and Model 3 in all the test scenarios.

Table 22 : Summary of the performance of all Models

		Accuracy of Model 1	**Accuracy of Model 2**	**Accuracy of Model 3**
Distance calculation and Alert generating test	Open Spaces	94.64%		94.46%
	Closed spaces with no environmental interference	97.14%		96.07%
	Closed spaces with environmental interference	96.07%		96.25%
Safe Location Test		90.71%	87.50 %	

Chapter 6

6 Conclusion

In this thesis, we presented a smartphone loss prevention system using 3 different approaches with different methods. Our core system is designed to help prevent the loss of smartphones and provide a reliable and accurate way to make user aware in smartphone loss scenarios. We began by discussing the importance of smartphones, smartphone's data, and the potential consequences of losing them. We then reviewed related literature and identified the shortcomings of existing systems. Our proposed solution addresses these shortcomings by using the RSSI of Bluetooth and the Haversine formula in separate models to calculate the distance between a smartphone and a smartwatch.

We developed an Android application that runs on the smartphone and a smartwatch application acting as a watch face that runs on the smartwatch. The two applications communicate using Bluetooth, and the applications combinedly works to keep track of users' presence in safe location and calculates the distance between the smartphone and the smartwatch in unsafe location and generates alert accordingly. To evaluate the performance of the system, we conducted experiments using different predefined thresholds and in various locations and different infrastructures. The results showed that the system can detect when the distance between the two devices exceeds a predefined threshold in an unsafe location and generate alert accordingly, and it has a low false positive rate. Of all 3 models of ours, Model 1 which has been implemented with RSSI method and Geofencing has acquired high accuracy rates in most of the evaluations and outperformed Model 2 and Model 3. Due to the poor quality of GPS performance of smartwatch, Model 2, and model 3 couldn't perform well combinedly in all of the features of our system.

In conclusion, our proposed smartphone loss prevention system is a reliable and accurate solution for preventing the loss of smartphones. It can help users avoid the potential consequences of losing their smartphones and to reduce the risk of losing a phone and protect personal data. Moreover, the system saves users from the stress and inconvenience of losing their phone. Overall, we believe that our research can contribute to the development of more advanced smartphone loss prevention systems in the future.

6.1 Limitations and Future Work

As the performance of our models depends on the GPS accuracy of both smartphone and smartwatch, performance of the GPS module is considered as a limitation of our models. Especially, the performance of the GPS module of smartwatch has been found weak according to our system requirement. Our model 2 and model 3 would have performed well in terms of calculating distance and safe location tracking using the haversine method if the GPS module of the smartwatch provided much more accurate results. Standalone GPSs are typically accurate to within a 4.9 m (16 ft.) radius under open sky which worsens near buildings, bridges, and trees [47]. Moreover, the GPS module of resource constrained smartwatches performs a lot less than conventional GPS. This amount of error is not ideal for our system. Therefore, to address this issue, implementation of Real-Time Kinematic (RTK) GPS will be a perfect solution for our system. This technique is mostly used for applications that require higher accuracies, such as centimeter-level positioning [48]. Using RTK can improve positional accuracy to within 3 centimeters [49]. This will efficiently be able to support our system by user presence in the safe places given by the user as well as in the distance calculation by haversine method.

Moreover, in cases where the model does not generate an alert, and this aligns with the actual scenario where an alert should not be generated, we consider this outcome as a true negative. However, due to time constraints, we were unable to conduct a comprehensive true negative test during this phase of testing. Evaluating true negatives is an important aspect of model validation, as it helps confirm that the system correctly identifies situations where alerts are not warranted. Therefore, investigating true negatives will be designated as part of future work to ensure the model's overall accuracy and effectiveness in different operational contexts.

Our primary objective is to determine whether our model generates an alert when we surpass the preferred distance threshold value. Currently, we have conducted tests in both indoor and outdoor settings to evaluate this functionality. However, it's important to observe how the models work in highly crowded public places. due to time constraints, we were unable to execute tests in crowded settings during this phase. As part of future work, we are committed to conducting tests in highly congested areas. This approach will help us comprehensively validate the model's alert generation capability under more demanding and realistic conditions.

Furthermore, we have used the android application mostly to implement the safe location tracking feature of our system. However, the computation cost and power consumption of the android application can be substantially lowered if we offload the work to server. For time constraint, we could not implement the geofencing (Model 1) and haversine algorithm (Model 2) for safe location tracking in server. Nonetheless, this can be very efficient for the user in terms of significantly diminished energy consumption.

Smartwatches typically have smaller batteries than smartphones, which could limit their usefulness in long-term location tracking for the distance calculation by haversine method. Future work could focus on optimizing power consumption and developing new technologies to extend battery life.

Bibliography

[1] “SIM swap fraud explained and how to help protect yourself | Norton.” https://us.norton.com/blog/mobile/sim-swap-fraud#

[2] “Mobile Device Security: Startling Statistics on Data Loss and Data Breaches | The ChannelPro Network.” https://www.channelpronetwork.com/article/mobile-device-security-startling-statistics-data-loss-and-data-breaches

[3] “Prey’s Mobile Theft & Loss Report 2020 Finds 67% of Mobile.” https://www.globenewswire.com/en/news-release/2020/03/13/2000245/0/en/Prey-s-Mobile-Theft-Loss-Report-2020-Finds-67-of-Mobile-Losses-Occur-in-Interior-Locations-33-When-Users-are-in-Transit-Movement.html

[4] “Most Consumers Don’t Wear a Watch Daily, Survey Finds – JCK.” https://www.jckonline.com/editorial-article/most-consumers-dont-wear-a-watch-daily-survey-finds/

[5] H. Wang, J. Wan, and R. Liu, “A novel ranging method based on RSSI,” *Energy Procedia*, vol. 12, pp. 230–235, 2011, doi: 10.1016/J.EGYPRO.2011.10.032.

[6] “Smartwatch Statistics 2023: How Many People Use Smartwatches?” https://www.demandsage.com/smartwatch-statistics/

[7] “Bangle.js 2 - Espruino.” https://www.espruino.com/Bangle.js2

[8] “Bangle.js 2 Smart Watch.” https://shop.espruino.com/banglejs2

[9] Z. Hajiakhondi-Meybodi, M. Salimibeni, K. N. Plataniotis, and A. Mohammadi, “Bluetooth low energy-based angle of arrival estimation via switch antenna array for indoor localization,” *Proceedings of 2020 23rd International Conference on Information Fusion, FUSION 2020*, Jul. 2020, doi: 10.23919/FUSION45008.2020.9190573.

[10] J. J. Treurniet, C. Sarkar, R. V. Prasad, and W. De Boer, “Energy consumption and latency in BLE devices under mutual interference: An experimental study,” *Proceedings - 2015 International Conference on Future Internet of Things and Cloud, FiCloud 2015 and 2015 International Conference on Open and Big Data, OBD 2015*, pp. 333–340, Oct. 2015, doi: 10.1109/FICLOUD.2015.108.

[11] “Bluetooth LE Working, Architecture, Uses.” https://www.spiceworks.com/tech/iot/articles/what-is-bluetooth-le/

[12] “BLE | Bluetooth Low Energy – Why Everyone Is Using It?” https://www.avsystem.com/blog/bluetooth-low-energy-ble/

[13] “How Bluetooth Low Energy Works: Advertisements (Part 1) | Novel Bits.” https://novelbits.io/bluetooth-low-energy-advertisements-part-1/

[14] “Bluetooth 5 - nordicsemi.com.” https://www.nordicsemi.com/Products/Bluetooth-Low-Energy/Bluetooth-5

[15] “nRF52840 - Bluetooth 5.3 SoC - nordicsemi.com.” https://www.nordicsemi.com/products/nrf52840

[16] “Why is almost everything negative in Wireless? - Cisco Community.” https://community.cisco.com/t5/small-business-support-knowledge-base/why-is-almost-everything-negative-in-wireless/ta-p/3159743 (accessed Aug. 23, 2023).

[17] “Wi-Fi Signal Strength: That Cool -70 dBm | Dong Knows Tech.” https://dongknows.com/wi-fi-signal-strength-dbm-explained/ (accessed Aug. 23, 2023).

[18] “Find Distance Between Two Bluetooth Devices Formula | BeingCoders.” https://medium.com/beingcoders/convert-rssi-value-of-the-ble-bluetooth-low-energy-beacons-to-meters-63259f307283

[19] “Can Bluetooth Be Used To Measure Distance? (Explained!).” https://propairing.com/can-bluetooth-be-used-to-measure-distance/

[20] J. Larsson, “Distance Estimation and Positioning Based on Bluetooth Low Energy Technology,” 2015, [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-174857

[21] “Computing Distances.” https://cs.nyu.edu/visual/home/proj/tiger/gisfaq.html (accessed Mar. 10, 2023).

[22] “Distance on a sphere: The Haversine Formula - Esri Community.” https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128 (accessed Mar. 11, 2023).

[23] “Global Positioning System History | NASA.” https://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

[24] “GPS.gov: GPS Overview.” https://www.gps.gov/systems/gps/

[25] “Can Bluetooth Be Used To Measure Distance? (Explained!).” https://propairing.com/can-bluetooth-be-used-to-measure-distance/

[26] “GPS.gov: Trilateration Exercise.” https://www.gps.gov/multimedia/tutorials/trilateration/

[27] “What Is GPS? The Complete Guide to the Uses of GPS | Geotab.” https://www.geotab.com/blog/what-is-gps/

[28] “17. Satellite Ranging | The Nature of Geographic Information.” https://www.e-education.psu.edu/natureofgeoinfo/c5_p18.html

[29] “What Is Geofencing? Everything You Need to Know About Location-Based Marketing.” https://www.smartbugmedia.com/blog/what-is-geofencing

[30] “Geofencing API | Google Developers.” https://developers.google.com/location-context/geofencing (accessed Apr. 26, 2023).

[31] "Android Statistics (2023) - Business of Apps." https://www.businessofapps.com/data/android-statistics/

[32] M. J. Hussain, L. Lu, and S. Gao, "An RFID Based Smartphone Proximity Absence Alert System," *IEEE Trans Mob Comput*, vol. 16, no. 5, pp. 1246–1257, May 2017, doi: 10.1109/TMC.2016.2591524.

[33] D. Aradhana, V. Deepa, H. T. Tasleem S, S. Yasmeen, and S. L. Shazeen, "RFID BASED SMART PHONE PROXIMITY ABSENCE ALERT SYSTEM USING IOT," 2017. [Online]. Available: www.ijtre.com

[34] N. Bzowski, "A framework for a smartphone loss prevention system Final Progress Report," UG Thesis, Dept of Computer Science, Western University, 2021.

[35] X. D. Yang, K. Hasan, N. Bruce, and P. Irani, "Surround-See: Enabling peripheral vision on smartphones during active use," *UIST 2013 - Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, pp. 291–300, 2013, doi: 10.1145/2501988.2502049.

[36] J. Chen, U. Hengartner, H. Khan, and M. Mannan, "Chaperone: Real-time Locking and Loss Prevention for Smartphones," in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020, pp. 325–342. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/chen-jiayi

[37] "Solved: Annoying notification when not connected to phone - Samsung Community - 458178." https://us.community.samsung.com/t5/Galaxy-Watch/Annoying-notification-when-not-connected-to-phone/td-p/458178

[38] T. Li, Y. Chen, J. Sun, X. Jin, and Y. Zhang, "iLock: Immediate and Automatic Locking of Mobile Devices against Data Theft," *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, doi: 10.1145/2976749.

[39] “The 7 Best Android Anti-Theft Apps to Protect Your Phone.” https://www.makeuseof.com/tag/4-android-anti-theft-solutions-compared-which-is-the-best/

[40] “Solved: The ‘disconnected from phone’ buzz - Samsung Community - 2201239.” https://us.community.samsung.com/t5/Galaxy-Watch/The-quot-disconnected-from-phone-quot-buzz/td-p/2201239 (accessed Apr. 13, 2023).

[41] “The differences between Bluetooth 5.0 and 5.2 - Newbit Information.” https://www.newbitsiot.com/the-differences-between-bluetooth-5-0-and-5-2/ (accessed Mar. 18, 2023).

[42] “Mini GPS/BDS Unit (AT6558) | m5stack-store.” https://shop.m5stack.com/products/mini-gps-bds-unit (accessed Mar. 18, 2023).

[43] “Specs | Samsung Galaxy S22 & S22+ | Samsung Canada.” https://www.samsung.com/ca/smartphones/galaxy-s22/specs/ (accessed Mar. 16, 2023).

[44] “Request location permissions | Android Developers.” https://developer.android.com/training/location/permissions (accessed Mar. 22, 2023).

[45] “Create and monitor geofences | Android Developers.” https://developer.android.com/training/location/geofencing (accessed Mar. 22, 2023).

[46] “How to Check the Accuracy of Your Machine Learning Model | Deepchecks.” https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/ (accessed Mar. 30, 2023).

[47] “GPS.gov: GPS Accuracy.” https://www.gps.gov/systems/gps/performance/accuracy/ (accessed Mar. 31, 2023).

[48] "Real time operation with cm level accuracy." https://support.sbg-systems.com/sc/kb/latest/inertial-sensors-operation/real-time-operation-with-cm-level-accuracy (accessed Mar. 31, 2023).

[49] "Real-time Kinetics (RTK). Centimeter-level accuracy for the… | by Mauricio Andrada | Medium." https://mauricioandrada.medium.com/real-time-kinetics-rtk-dd791a8e55d5 (accessed Mar. 31, 2023).

Curriculum Vitae

Name:	Noshin Tasnim
Post-secondary Education and Degrees:	M.Sc. Candidate, Computer Science The University of Western Ontario 2021-2023
	B.Sc., Computer Science & Engineering BRAC University, Dhaka, Bangladesh 2015-2019
Honours and Awards:	Western Graduate Research Scholarship (WGRS) The University of Western Ontario 2021-2022
Related Work Experience	Graduate Teaching Assistant The University of Western Ontario 2021-2023