

Daniel Hunegnaw

**FallAlarm – Fall Detection and Alarm App**

**EE P 523 Mobile Applications for Sensing and Control| 08-015-21**

Contents

[**OBJECTIVE** 2](#_Toc80049919)

[**INTRODUCTION** 2](#_Toc80049920)

[**SECTION I** 2](#_Toc80049921)

[Modeling thresholds 2](#_Toc80049922)

[Fall detecton Algorithim 4](#_Toc80049923)

[**SECTION II** 5](#_Toc80049924)

[Design the APP 5](#_Toc80049925)

[**UI**: 5](#_Toc80049926)

[**View Models**: 6](#_Toc80049927)

[**Repositories**: 6](#_Toc80049928)

[**Database**: 6](#_Toc80049929)

[**Fall Detector Background Service**: 6](#_Toc80049930)

[**BroadcastReciever** 6](#_Toc80049931)

[Screenshots of the APP 6](#_Toc80049932)

[**SECTION III** 10](#_Toc80049933)

[ISSUES and mitgations 10](#_Toc80049934)

[POTENTIAL Improvement 11](#_Toc80049935)

[**SECTION IV** 11](#_Toc80049936)

[CONCLUSION 11](#_Toc80049937)

[REFERENCES 11](#_Toc80049938)

# **OBJECTIVE**

The objective of this paper is present the design, and implementation of an Android mobile App called FallAlarm, which is used to detect when a person carrying the phone falls. The project aims to provide knowledge, and hands-on experience in designing, and programming of android app using Kotlin programming language. The project uses the built-in accelerometer sensor present in android phones as source of data for detecting fall.

# **INTRODUCTION**

In this paper, it will be presented the design and implementation of an android app for detecting when a person carrying the phone falls, alerting an emergency contact when a fall is detected via SMS. The app uses the built-in accelerometer , GPS, SMS messaging system in the phone, SQLite database for storing data and is written in Kotlin.

In section I, a brief description of data modeling, and development of threshold-based algorithm will be presented.

In section II, a high-level design of the app and the interaction of various components will be presented using a block diagram. In addition, the implementation of each component will be briefly explained

In Section III the issues faced during the project and future enhancements works will be briefly explained.

Finally, in Section IV, we will provide a conclusion and references used in the project

# **SECTION I**

## Modeling thresholds

To detect a fall, a threshold-based algorithm is proposed. Thus, the first phase of the project was to establish thresholds that will enable us to decide if a fall was detected. In this modeling phase of the project, a simple accelerometer data collection app was developed. The app collects the vector magnitude of the three components of the accelerometer readings using the formula:

Am = √ (A2x+ A2y +A2z)

The app collects the data in an array. Upon a click event of a save button, the data (list of numerical values of the acceleration magnitude in a chronological order) is saved in a file.

Sample data collection was done as follows

1. **Walking** : Run the app, put the phone in a pant-pocket and walk around , and save the collected data
2. **Walk-Sit-Stand** : Run the app, put the phone in pant-pocket and walk, sit, stand, and walk again, and save the data collected
3. **Walk-Fall**: Run the app, put the phone in a pocket, and free fall on a mattress. And save the collected data
4. Data collected from each scenario was exported to excel and a liner graph (x-axis is a sequence of numbers representing relative time), and y-axis representing the acceleration in m/s2 was generated as shown in fig 1 below

Fig 1. Graphical representation of sample data collected.

Interestingly, as one can see from the graph, there is not much difference between walking and walk-sit-stand graph (red and gray lines) patterns. However, for the **walk and fall**, during the fall, the acceleration goes very low close to zero (less than 0.3m/s2, three down bursts of the yellow line) and immediately raises about 10m/s2 upon landing on the ground. Thus , after doing several similar experiments, an average lower and upper threshold patterns was developed. During the fall , the acceleration goes below a **lower** **threshold of 2.5m/s2** , and after impact, the acceleration magnitude goes above an **upper threshold of 10m/s2** within a specific during of **time [0.5, 1] seconds.**

## Fall detecton Algorithim

The following algorithm was developed based on the above modeling and analysis of data



# **SECTION II**

## Design the APP

The overall architecture of the app is presented below in fig 2. Each component of the app is described below.

Diagram

Description automatically generated

Fig 2. FallAlam App architecture

### **UI**:

The UI has of the app has two main screens . the Home Screen and the Setting Screen, which navigable from a side bar drawer menu

* **Home Screen**: used to display historical ( recent 5 detected falls) detected falls. The historical data observes the database via LiveData through the View Model and thus always is up to date with the latest incident data. It has also two buttons, **start button** – to start a background service for fall detection and **stop button** – to stop the background service. In addition, it has flashing (red and green) square to indicate that the background service is running
* **Setting Screen:** used to setup the emergency contact. It has two text boxes for entering the emergency contact’s name and a phone number. It used the View Model to interact with the database

The UI uses View Models to interact with the database. It observes the data changes in the database, especially history, to update the UI as it happens

### **View Models**:

There are two view models , one for emergency contact and the other for the history. The View Models interacts with the database, both write and read, through repositories.

### **Repositories**:

There are two repositories , contact and history repository, to provide an encapsulation over the room-based database

### **Database**:

The app uses SQLite database for storing emergency contact and history of fall detection data. A library called **Room** is used to provide an encapsulation over the SQLite database. The database has two tables , contact and history.

### **Fall Detector Background Service**:

As the name describes, this is the background service. This is the main component of the app. It has the following major functionality

* Gets the emergency contact’s phone number from the database via repository
* Listens to the Accelerometer sensor
* Used the GPS of the phone to track the location of any possible fall incidents
* Based on accelerometer data, it runs a fall-detection algorithm. If a fall is detected, it used SMS service to send a message to the emergency contact about the incident. And used the repository to write a history about the incident
* It periodically sends its status ( while alive) to the main use through a broadcast receiver

### **BroadcastReciever**

Used to receive status ‘alive’ signal from the Fall Detector Background service in the UI

## Screenshots of the APP

Below are presented a few screen shots of the app and as well as the text message received in the emergency contact’s phone

Text

Description automatically generated

Fig 3. Home Screen with latest 5 fall incidents

Graphical user interface, application, Teams

Description automatically generated

Fig 4. Sidebar menu

Graphical user interface, application, Teams

Description automatically generated

Fig 5. Emergency Contact input screen

Text

Description automatically generated

Fig 6. SMS message received from the FallAlarm app

# **SECTION III**

## ISSUES and mitgations

One of the biggest issues in the FallAlarm is false positives. For example, if a phone is dropped to the ground, a false positive alarm is sent to the emergency contact.

One Possible mitigation a possible future work is to add a **Cancel Button**. Adding a cancel button in the home screen will enable a user to cancel sending false alarms to emergency contact.

## POTENTIAL Improvement

Potential improvements that should be done, but not done due to time constraint in this project, includes polishing and branding the UI, adding validations for inputs such as phone numbers, battery optimization since there is a background service.

# **SECTION IV**

## CONCLUSION

In this project tremendous amount of knowledge and experience has been acquired. Hand on experience on how to use Kotlin programming language to write android apps, how to read and use sensor data, how to create background service, how to create and interact with SQLite database, how to use services such as SMS were explored and experienced.

The project was more than successful in providing the knowledge and the experience expected of the course material.

In addition, it has provided a positive psychological influence in creating the confidence to be able to convert any ideas into an app.

## REFERENCES

<https://developer.android.com/guide/topics/sensors/sensors_motion>

<https://developer.android.com/training/data-storage/room>

<https://www.geeksforgeeks.org/sending-a-text-message-over-the-phone-using-smsmanager-in-android/>

<https://medium.com/the-sixt-india-blog/ways-to-communicate-between-activity-and-service-6a8f07275297>

<https://www.geeksforgeeks.org/sending-a-text-message-over-the-phone-using-smsmanager-in-android/>