Accident Detection and Emergency Alert System

Table of Contents

---

1. Introduction

Accidents can be fatal if victims do not receive timely help. This project aims to develop an **Accident Detection and Emergency Alert System** that:

- **Detects accidents** using mobile sensors.
- **Finds the nearest hospital & police station.**
- **Automatically calls and sends messages** to emergency contacts.
- **Shares live location** for quick assistance.

---

2. System Overview

The system consists of:

- **Sensors:** Detect accident impact (Accelerometer & Gyroscope).
- **Location Services:** Fetch real-time GPS coordinates.
- **Google Places API:** Find nearest emergency services.

- **Messaging & Calling:** Notify emergency contacts.
- **User Interface:** Mobile app for configuration.

---

3. Components Required

- **Android Smartphone** with GPS & Internet.
- **Google Maps API Key** for location services.
- **SMS & Call Permissions** enabled.
- **Accelerometer & Gyroscope** to detect sudden impacts.
- **Emergency Contact Database** to store user contacts.

---

4. Accident Detection Mechanism

Using Accelerometer & Gyroscope

Accident detection is based on sudden changes in acceleration and orientation.

- If acceleration exceeds a **critical threshold**, an accident is detected.
- Machine learning can be used to minimize false positives.

Code Snippet:

```
sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)?.let { sensor ->
   sensorManager.registerListener(sensorEventListener, sensor,
SensorManager.SENSOR_DELAY_NORMAL)
}
```

---

5. Fetching User Location

Using GPS (FusedLocationProviderClient)

Fetches the user's real-time location:

```
private fun getUserLocation() {
   fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)
   fusedLocationClient.lastLocation.addOnSuccessListener { location: Location? ->
      if (location != null) {
```

```
            val latitude = location.latitude
            val longitude = location.longitude
            sendEmergencyAlert(latitude, longitude)
        }
    }
}
```

---

6. Finding Nearest Hospital & Police Station

Using Google Places API

We use the nearbysearch API to find hospitals & police stations within **5 km**:

```
val url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=$latitude,$longitude&radius=5000&type=hospital&key=$API_KEY"
```

Once we get the **Place ID**, we retrieve the **phone number**:

```
val detailsUrl = "https://maps.googleapis.com/maps/api/place/details/json?place_id=$placeId&fields=name,formatted_phone_number&key=$API_KEY"
```

---

7. Calling Emergency Services

Using ACTION_CALL Intent

Automatically call the nearest hospital or police station:

```
fun callEmergencyNumber(phoneNumber: String) {
    val callIntent = Intent(Intent.ACTION_CALL)
    callIntent.data = Uri.parse("tel:$phoneNumber")
    startActivity(callIntent)
}
```

---

8. Sharing Location via SMS, WhatsApp & Email

SMS Alert

```
fun sendEmergencyAlert(latitude: Double, longitude: Double) {
    val message = " 🚨 Accident Alert! Location: https://www.google.com/maps?q=$latitude,$longitude"
    SmsManager.getDefault().sendTextMessage(EMERGENCY_CONTACT, null, message, null, null)
}
```

WhatsApp Message

```
fun shareLocationWhatsApp(locationLink: String) {
    val intent = Intent(Intent.ACTION_SEND)
    intent.type = "text/plain"
    intent.putExtra(Intent.EXTRA_TEXT, " 🚨 Accident Alert! My location: $locationLink")
    intent.setPackage("com.whatsapp")
    startActivity(intent)
}
```

Email Alert

```
fun sendEmailAlert(locationLink: String) {
    val intent = Intent(Intent.ACTION_SEND)
    intent.type = "message/rfc822"
    intent.putExtra(Intent.EXTRA_EMAIL, arrayOf("emergency@example.com"))
    intent.putExtra(Intent.EXTRA_SUBJECT, " 🚨 Accident Alert!")
    intent.putExtra(Intent.EXTRA_TEXT, "Accident detected! My location: $locationLink")
    startActivity(Intent.createChooser(intent, "Send Email"))
}
```

---

9. Flow Diagram

```
Accident Detected (Sensor)
        |
Get GPS Location
```

---

10. Challenges & Drawbacks

1. False Positives

- Sudden braking or a phone dropping can trigger false alarms.
- Solution: Implement **machine learning** to detect real accidents.

2. GPS Limitations

- If the phone loses GPS signal (e.g., in tunnels), location tracking fails.
- Solution: Use **network-based location fallback**.

3. Battery Consumption

- Constant sensor monitoring and GPS tracking drain battery life.
- Solution: Optimize by running checks **only during potential accidents**.

4. Dependence on Internet

- API calls require an active internet connection.
- Solution: Implement **offline fallback** (e.g., storing emergency contacts locally).

5. Privacy Concerns

- Users may not want their location shared automatically.
- Solution: Allow users to **confirm before sending alerts** (if conscious).

---

11. Conclusion

- The system ensures quick accident detection & emergency response.
- It uses real-time **GPS, APIs, and automated calling/messaging**.
- This project can **save lives** by enabling faster medical assistance.
- Future improvements include **machine learning**, **offline functionality**, and **better energy optimization**.

🚀 **Next Steps**

- Implement **Machine Learning** for better accident detection.
- Create a **fully working Android App**.
- Deploy on **Google Play Store** for public use.
