



## Individual Project Report

Name: Sehrish Fahim

Email: [sehrish@stud.fra-uas.de](mailto:sehrish@stud.fra-uas.de)

Matriculation No. 1277599

Course: M.Eng (IT)

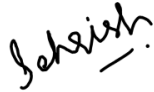
**Supervisor: Prof. Dr.-Ing. Kira Kastell**

Title: Android-based implementation of an emergency system

April | 2021

**Statement**

I confirm that I have written this student project on my own. No other sources were used except those referenced. Content that is taken literally or analogously from published or unpublished sources is identified as such. The drawings or figures of this work have been created by myself or are provided with an appropriate reference. This work has not been submitted in the same or similar form or to any other examination board.

A handwritten signature in black ink, appearing to read 'Seisich' with a horizontal line underneath.

19.04.2021

---

Date, the signature of the student

**Acknowledgment**

Firstly, I would like to thank Prof. Dr.-Ing. Kira Kastell, University of Applied Sciences for allowing me to do my student project in this transportation automation field which will help me to be a solid foundation for my carrier in the future and being my examiner in this individual project.

Moreover, it would be a big mistake here if I did not mention the support and encouragement of my family. I finally would like to thank my friends and seniors for their suggestions and guidance during the project and writing of this project report.

**Table of contents**

<i>1 Introduction</i>	<i>5</i>
<i>2 Theoretical Background</i>	<i>6</i>
<i>2.1 Technologies</i>	<i>6</i>
<i>2.1.1 Android Component</i>	<i>6</i>
<i>2.1.2 Java Component</i>	<i>6</i>
<i>2.1.2 Architecture Diagram</i>	<i>7</i>
<i>2.2 Infrastructure</i>	<i>7</i>
<i>2.3 Mobile Sensors</i>	<i>8</i>
<i>3 Outline Solution</i>	<i>10</i>
<i>4 Detailed Solution</i>	<i>11</i>
<i>3 Testing</i>	<i>17</i>
<i>4 Future Development</i>	<i>24</i>
<i>5 Conclusion</i>	<i>25</i>
<i>9 References</i>	<i>25</i>
<i>10 Appendix</i>	<i>25</i>

## **1 Introduction**

In the modern world, Automation plays a vital role in every field, for instance having a smartphone, smart car. If I talk about the intelligent transportation system it has a huge line of automation for cars, traffic. In this busy life with the amount of vehicles on the road having significantly increased hence the accident ratio has been drastically increased, there is a huge requirement for a system which is fully automated and controls traffic accidents and provides notification as well as good service of actions.

By this thought I came up with the idea to make an intelligent accident detection system, Now the question has raised, what will make it intelligent? does it require a lot of hardware equipment? The answer is no, it only requires a phone which everyone has in this era, so why not use this technology and make something more efficient and intelligent in the transport field.

I have chosen Android for the prototype. By choosing Android it doesn't mean it will only work for this phone, this project can be extended for iOS as well in the future so the idea is just to start with the existing technology and make something good out of it.

The main purpose of this student project is to provide an intelligent solution for accident events, I will explain which steps are required to take to achieve this goal I will use Android, Java, MySQL, Google Api technologies to accomplish the desired output.

## **2      *Theoretical Background***

This chapter will discuss all the theoretical background used in this student project. To understand the implementation of this student work it is essential to discuss the theoretical part from all perspectives which are related to the technology. It is very essential and useful to understand the theories, architecture and software which have been used in this project work. Firstly, an introduction about technologies used to build this software is given. Further-more an overview of the project will be shown, this chapter will discuss:

- Technologies
- Architecture Diagram

### **2.1      Technologies:**

To achieve this project, it is split into two parts of technology, one is a smartphone and one is the backend services, as a smartphone, one with Android technology is used, for backend services the java platform is used with MySQL database.

#### **2.1.1      Android Component**

For this project, the main goal has been achieved by Android technology, there are sensors available built-in in mobile which can work with android OS to detect the accident.

1-accelerometer sensor

2-Gyroscope

The Android platform is also used to create an application, in which the user will log in and enter phone number on which SMS will be send and that's it, the rest of the things will be handled in the back-end by this Android application.

#### **2.1.2      Java Component**

The Java component is one of the main pillars in this project, as the complete back-end is based on the services which are called in the android application.

The user interface to communicate with a hospital is also made by using Java.

### 2.1.3 Architecture Diagram

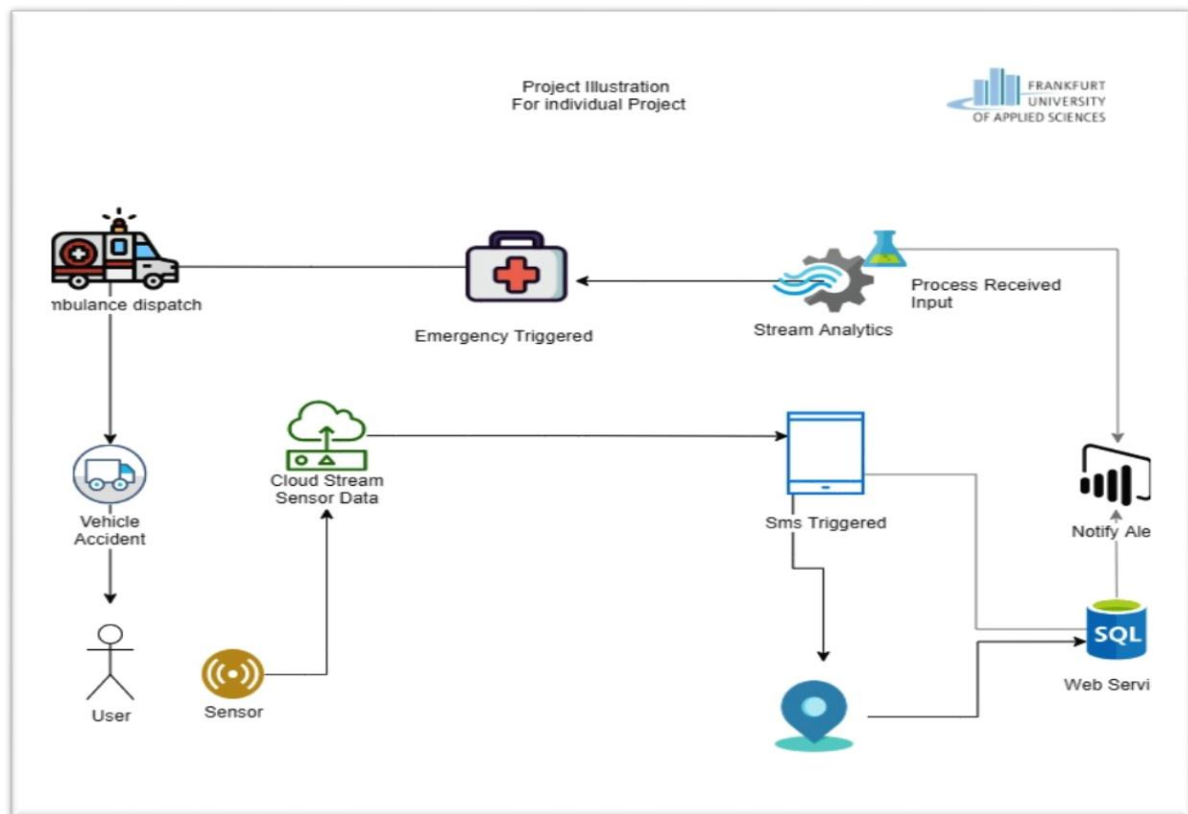
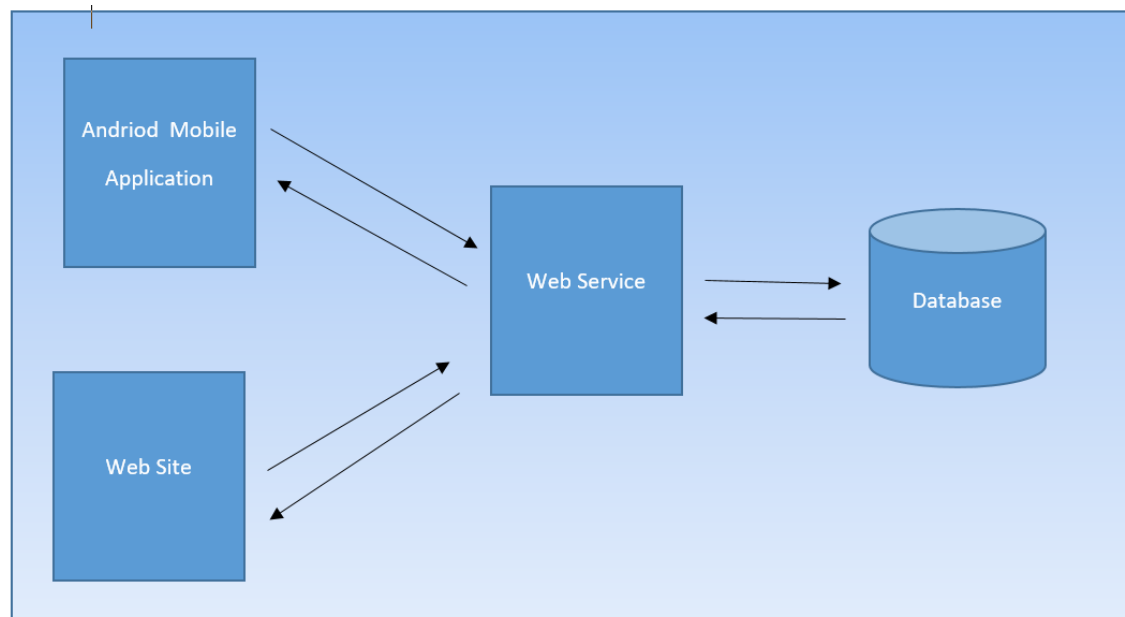


Figure 1 Architecture Diagram

The Architecture Diagram consist of the following components:

1. User: The user is the one who has installed the application and registered him/herself in this application.
2. Sensor: Sensors are inside the mobile and activated with this application so that it can trace the location and accident if happened.
3. SMS Trigger: as soon as it detected an accident, a SMS will be sent to the favourite number,
4. Web Service: this service does multiple tasks, like sending data to the database, creating a user API, accident data, etc.

## 2.2 Infrastructure



**Figure 2 Infrastructure Diagram**

The infrastructure for the application is shown in the above figure. This application is based on the main four components, mobile application, website, web service, and database.

- The mobile application is based on sensors working and a location tracker, For the user, it also provides an interface to add the favorite phone number by registering in the application.
- The website is for tracking the location on a map and for hospital users to see accident details.
- The web service acts like the main engine which does all the backend work creating users, setting data into the database, sending SMS.
- The database has all the data which we need to proceed with.

## 2.3 Mobile Sensors

Most Android phones have sensors for motion detection which gives accurate results.

In this project we are using the following sensors:

- Accelerometer
- Gyroscope



## Accelerometer and Gyroscope:

Accelerometer measures acceleration including force of gravity. By, detecting acceleration of the mobile it is possible to measure the velocity and speed to know the exact force used to detect an accident, while the gyroscope identifies rotation [1]. By using these available sensors, for this project, I analyse and calculate speed and gforce. I have used different values to check the speed to drop mobile and angles it was purely designed by attempt to create a scenario for mobile drop while person is sitting in a car, there is no accurate value available on internet therefore, I have created test by myself to at least reach to the point where mobile trigger the accident event. The following values are finalized for minimum threshold values for speed and gforce.

Speed: 1000 m/s and gforce : 1.7 m/s<sup>2</sup>

Above threshold values has been set to detect an accident, for further development these values are configurable and can be change or modify.

## Parameters for Warning and Waiting time:

In this project there are few parameters created for ease so that later is can be configure following are the parameters and its explanations:

TIME\_TO\_WAIT (in seconds): When accident alert warning shows on screen it will stay till this value.

RADIUS\_FOR\_OTHER\_USER\_IN\_METER: when accident happens, the other users near the configured distance in meter will be notify about the accident and its location.

3 **Outline Solution:** The project is based on three parts:

- Web solution
- Mobile application
- Web services

### 3.1 Web Solution

Web solution is provided for hospital admins, each hospital needs to register on this page, then they can monitor the accidents and take the required action.

### 3.2 Mobile Solution:

A mobile solution is provided for users so that the application will able to track if an accident happens, the user needs to register on the application and add a friend's number. At the time of accident information will be sent to this phone number.

### 3.3 Web Services:

Web service is responsible to build communication between application and users. Therefore, several database entries are needed, e.g. The database will handle by the web services.

An important point is, for location and other services, Google doesn't support free service anymore. I have created an account to get the location and other services by which Google provides an API Key which we can use.

#### 4 Detailed Solution

Until now it was all theoretical and a general idea about how the application works and what it does, now let's have a look at its design and complete the functional behaviors and features and its set up requirement.

##### Set up Requirement:

- Install MySQL
- Import Database
- Install JDK

##### Code Explanation:

```
ConfigConstants.baseUrl = "http://" + ipText + ":8085";
apiInterface = APIClient.getClient().create(APIInterface.class);

Toast.makeText(getBaseContext(), text: "Checking User...", Toast.LENGTH_SHORT).show();
User userData = new User( userId: 0, phoneText, name: "", email: "", passText);
Call<User> call1 = apiInterface.getUserData(userData);

call1.enqueue(new Callback<User>() {
    @Override
    public void onResponse(Call<User> call, Response<User> response) {

        User responseData = response.body();
        if (responseData != null && responseData.getUserId() > 0) {

            SharedPreferences.Editor editor = sharedPreferences.edit();
            editor.putString("userphone", phoneText);
            editor.putString("password", passText);
            editor.putString("userid", responseData.getUserId() + "");
            editor.putString("ipText", ipText + "");
            editor.commit();

            loggedInUserId = responseData.getUserId();
            goToHomeActivity();
        } else {
            Toast.makeText(getBaseContext(), text: "User is invalid:", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Figure – User Login

The above code shows the validation part of the mobile user to sign-in on the mobile application.

```

private void initializeAccidentNotification() {

    PusherOptions options = new PusherOptions();
    options.setCluster("eu");

    Pusher pusher = new Pusher( apiKey: "a9e662200a7506256f55", options);
    pusher.connect(new ConnectionEventListener() {

        @Override
        public void onConnectionStateChange(ConnectionStateChange change) {
            Log.i( tag: "Pusher", msg: "State changed from " + change.getPreviousState() +
                " to " + change.getCurrentState());
        }

        @Override
        public void onError(String message, String code, Exception e) {
            Log.i( tag: "Pusher", msg: "There was a problem connecting! " +
                "\ncode: " + code +
                "\nmessage: " + message +
                "\nException: " + e
            );
        }
    }, ConnectionState.ALL);

    Channel channel = pusher.subscribe( channelName: "my-channel");
    channel.bind( eventName: "my-event-" + loggedInUserId, new SubscriptionEventListener() {

```

Figure - Accident Notification

The above code is setting notifications to the user for accident events.

```

private void createNotificationChannel(String message) {

    CharSequence name = "accident";
    String CHANNEL_ID = "Accidents";
    String[] messagesParts = message.split( regex: "#");
    String title = messagesParts[0];
    String text = title;

    if(messagesParts.length > 2) {
        double lat = Double.parseDouble(messagesParts[1]);

        String strLng = messagesParts[2].replace( target: "\", replacement: "");
        double lng = Double.parseDouble(strLng);

        String address = getAddress(lat, lng);
        if(address != null) {
            text = address;
        }
    }
}

```

Figure - Notification channel

Above code shows the notification channel to notify user about nearby accident.

```

private String getAddress(double lat, double lng) {
    Geocoder geocoder = new Geocoder( context: this, Locale.getDefault());
    try {
        List<Address> addresses = geocoder.getFromLocation(lat, lng, maxResults: 1);
        Address obj = addresses.get(0);

        String street = obj.getThoroughfare();
        String houseNumber = obj.getSubThoroughfare();
        String postalCode = obj.getPostalCode();
        String address = street;

        if(houseNumber != null) address = address + " " + houseNumber;
        if(postalCode != null) address = address + " " + postalCode;

        Log.v( tag: "IGA", msg: "Address" + address);
        return address;
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        Toast.makeText( context: this, e.getMessage(), Toast.LENGTH_SHORT).show();
        return null;
    }
}

```

Figure - Coordinates to address

The above code shows how I converted the coordinates into addresses by using a geocoder.

```

private boolean checkAndRequestPermissions() {

    int fineLocation = ContextCompat.checkSelfPermission( context: this, android.Manifest.permission.ACCESS_FINE_LOCATION);
    int sendSms = ContextCompat.checkSelfPermission( context: this, android.Manifest.permission.SEND_SMS);
    int internet = ContextCompat.checkSelfPermission( context: this, android.Manifest.permission.INTERNET);
    int vibration = ContextCompat.checkSelfPermission( context: this, android.Manifest.permission.VIBRATE);

    List<String> listPermissionsNeeded = new ArrayList<>();

    if (fineLocation != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(android.Manifest.permission.ACCESS_FINE_LOCATION);
    }
    if (sendSms != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(android.Manifest.permission.SEND_SMS);
    }
    if (internet != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(android.Manifest.permission.INTERNET);
    }
    if (vibration != PackageManager.PERMISSION_GRANTED) {
        listPermissionsNeeded.add(android.Manifest.permission.VIBRATE);
    }

    if (!listPermissionsNeeded.isEmpty()) {
        ActivityCompat.requestPermissions( activity: this,
            listPermissionsNeeded.toArray(new String[listPermissionsNeeded.size()]),REQUEST_ID_MULTI);
        return false;
    }
}

```

Figure- Permission check for mobile user

The above code shows the permission check for the user after sign-in first time.

```
private void detectShake() {
    mShakeService.setOnShakeListener(new ShakeService.OnShakeListener() {
        @Override
        public void onShake(int count) {
            Long lastCallDateTimeInMillis = sharedPreferences.getLong( key: "lastAccidentCallDateTime", defValue: 0);
            Long currentDateTimeInMillis = System.currentTimeMillis();
            Long diffInMilliseconds = currentDateTimeInMillis - lastCallDateTimeInMillis;

            if(lastCallDateTimeInMillis == 0 || (diffInMilliseconds > ConfigConstants.TIME_DELAY_FOR_ACCIDENT_CALL_IN_MILLIS)) {
                Intent dialogIntent = new Intent(getBaseContext(), AccidentCallPendingActivity.class);
                dialogIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                startActivity(dialogIntent);
            } else {
                Log.d( tag: "shake_detected", msg: "Not sending accident call for delay");
            }
        }
    });
}
```

Figure- Detect accident

The above code shows the event trigger when the sensor detects the shake, above code will launch the warning screen which will stay on screen for 30 seconds.

```
public void saveFriendDetail(View view) {
    String userId = sharedPreferences.getString( key: "userid", defValue: "");
    EditText editPhone = (EditText) findViewById(R.id.friend_number_input);

    String phoneText = editPhone.getText().toString();
    UserFriend userFriend = new UserFriend(userId, phoneText, id: 0);
    Call<UserFriend> callSaveFriend = apiInterface.saveUserFriend(userFriend);

    callSaveFriend.enqueue(new Callback<UserFriend>() {
        @Override
        public void onResponse(Call<UserFriend> call, Response<UserFriend> response) {
            UserFriend respondLocationData = response.body();
            fillFriendsList();
        }

        @Override
        public void onFailure(Call<UserFriend> call, Throwable t) {
            Toast.makeText( context: HomeActivity.this,
                text: "Check Internet connection or Server", Toast.LENGTH_LONG).show();
            call.cancel();
        }
    });
}
```

Figure- Save the phone number

The above screen shows the saving method when the user enters mobile numbers into the list.

```

    ringtone = RingtoneManager.getRingtone(getApplicationContext(), notification);
    vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);

    try {
        if(detectVibrate(getApplicationContext())) {
            vibrator.vibrate(ConfigConstants.TIME_TO_WAIT);
        }
        ringtone.play();
    } catch (Exception e) {
        e.printStackTrace();
    }

    waitAndCallAccidentHandler = new Handler(Looper.getMainLooper());
    waitAndCallAccidentHandler.postDelayed(new Runnable() {
        @Override
        public void run() { isNeedToCallForAccident(); }, ConfigConstants.TIME_TO_WAIT);

    waitAndAutoCloseProcessesHandler = new Handler(Looper.getMainLooper());
    waitAndAutoCloseProcessesHandler.postDelayed(new Runnable() {
        @Override
        public void run() { stopAllProcesses(); }, ConfigConstants.TIME_TO_CLOSE_ALERT);
    }
}

```

Figure - Accident warning screen

The above code shows when a screen pops up for an accident warning it holds for 30 seconds this can be seen in the parameter in the above code name "TIME\_TO\_WAIT" after that it will call the method name "isNeedToCallForAccident" method.

```

private void isNeedToCallForAccident() {

    Long lastCallDateTimeInMillis = sharedPreferences.getLong( key: "lastAccidentCallDateTime", defValue: 0);
    Long currentDateTimeInMillis = System.currentTimeMillis();
    SharedPreferences.Editor editor = sharedPreferences.edit();
    Long diffInMillis = currentDateTimeInMillis - lastCallDateTimeInMillis;

    if (lastCallDateTimeInMillis == 0 || (diffInMillis > ConfigConstants.TIME_DELAY_FOR_ACCIDENT_CALL_IN)) {
        editor.putLong("lastAccidentCallDateTime", currentDateTimeInMillis);
        editor.commit();
        Log.d( tag: "shake_detected", msg: "sending accident call");
        sendAccidentCall();
    }
}

public void sendAccidentCall() {
    LocationTrack locationTrack = new LocationTrack(getApplicationContext());

    if (locationTrack.canGetLocation() == true) {
        double longitude = locationTrack.getLongitude();
        double latitude = locationTrack.getLatitude();
        locationTrack.sendAccidentLocation( longitude: longitude + "", latitude: latitude + "", loggedInUser);
        sendSMSToFriends(latitude, longitude);
        Log.d( tag: "shake_detected", msg: "sending sms lat log" + latitude + " " + longitude);
    } else {
        Log.d( tag: "shake_detected", msg: "can not get location");
    }
}

```

Figure - Accident method event

The above method checks the accident from the same mobile that did not happen 20 seconds ago (to eliminate false alarm) then it will send accident coordinates to the database and send SMS to mobile numbers.

```
@PostMapping("/save-accident")
public Accident save(@RequestBody AccidentDto accidentDtoo) throws JsonProcessingException {

    Accident accident = new Accident();
    accident.setLat(accidentDtoo.getLat());
    accident.setLon(accidentDtoo.getLon());
    accident.setCreatedDate(new Date());
    accident.setProcessed(false);

    User user = userRepository.findById(accidentDtoo.getUserId());

    Date endDateTimeRange = new Date();
    Date startDateTimeRange = new Date(System.currentTimeMillis() - (4 * 60 * 60 * 1000));

    accidentRepository.findAccidentByUserAndDateRange(startDateTimeRange, endDateTimeRange);
    Hospital hospital = findNearestHospital(accidentDtoo.getLat(), accidentDtoo.getLon(), accidentDtoo.getUserId());

    accident.setUser(user);
    accident.setHospital(hospital);
    accidentRepository.saveAndFlush(accident);

    try {
        notificationService.findAllUserNearByToSendNotification(accidentDtoo.getLat(), accidentDtoo.getLon(), accidentDtoo.getUserId());
    } catch (ParseException e) {
        e.printStackTrace();
    }
}
```

Figure- Service for saving accident

The above code shows the events after accident detection it saves the record in the database. By accident coordinates, it calls a function to find the nearest hospital and save it to the database. It then finds all users near the radius of the accident to send the notification.

```
private Hospital findNearestHospital(String lat, String lon, long accidentid) throws JsonProcessingException {

    HttpHeaders headers = new HttpHeaders();
    headers.set("Accept", MediaType.APPLICATION_JSON_VALUE);

    String url = "https://maps.googleapis.com/maps/api/place/findplacefromtext/json";
    RestTemplate restTemplate = new RestTemplate();

    UriComponentsBuilder builder = UriComponentsBuilder.fromHttpUrl(url)
        .queryParams("input", "hospital")
        .queryParams("inputtype", "textquery")
        .queryParams("fields", "formatted_address,name,rating,opening_hours,geometry")
        .queryParams("locationbias", "circle:1000@" + lat + "," + lon)
        .queryParams("key", "AIzaSyD9mLX49vr0Vkw4j5t8g");

    HttpEntity<> entity = new HttpEntity<>(headers);

    HttpEntity<NearestHospitalsResponse> response = restTemplate.exchange(
        builder.toUriString(),
        HttpMethod.GET,
        entity,
        NearestHospitalsResponse.class);
}
```

Figure - Find the nearest hospital

The above code shows the method to find nearest hospital by sending the coordinates to google API and it returns the nearest hospital.

```
public void findAllUserNearByToSendNotification(String lat, String lon, User user) throws ParseException {
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
    Date today = formatter.parse(formatter.format(new Date()));
    List<Integer> alreadySentUserId = new ArrayList<>();
    int RADIUS_FOR_OTHER_USER_IN_METER = 1000;

    List<Location> locations = locationRepository.findAllUserLatestLocation(today);

    for(Location location: locations) {
        if(location.getUserId() != user.getId() && alreadySentUserId.contains(location.getUserId()) == false) {
            double distance = distanceInMeter(lat, lon, location.getLat(), location.getLon());
            if(distance <= RADIUS_FOR_OTHER_USER_IN_METER) {
                sendNotification(location.getUserId(), message: "There is an accident nearby#" + lat + "#" + lon);
            }

            System.out.println("----- Distance = " + distance);
        }
        alreadySentUserId.add(Integer.parseInt(location.getUserId() + "#"));
    }
}
```

Figure - Filter users to send notification

The above code shows the logic for filter the users to send a notification, it checks all user's latest coordinates and filter which users are near to the location of the accident as the parameter "RADIUS\_FOR\_OTHER\_USER\_IN\_METER" if a user is inside of this range, Notification will be sent to them. This parameter value is set to 100 meters. This value is practical and this application has all real-time data for nearby users that is why in this range users will notify easily and this radius parameter can be increase or decrease as well.

```
@Autowired
AccidentRepository accidentRepository;

@GetMapping("/web/accidentlocation")
public String Test(@Param("uid") String uid, Model model) throws JsonProcessingException {
    long userId = Long.parseLong(uid);
    Accident accident = accidentRepository.findFirstByUserIdOrderByIdDesc(userId);
    if (accident == null) {
        AccidentDto accidentDto = new AccidentDto();
    } else {
        AccidentDto accidentDto = AccidentDto.builder()
            .userId(userId)
            .lat(accident.getLat())
            .lon(accident.getLon())
            .createDate(accident.getCreateDate())
            .build();
        model.addAttribute("accidentDto", accidentDto);
    }

    return "accidentlocation";
}
```

Fig- User accident location

The above method explains the UI code for the user. it fetches the latest record from the database for a given user and passes it to the screen.

#### 4.1 Mobile Application (Intelligent Accident Detection IAD)

The user has to download the application. Once it is downloaded, the user has to create a signup, and three fields are required

-User name, password, and web server IP.



The web server IP is needed as an input because this solution is not deployed on any server and it is locally running. To make the complete solution run, a user has to give the IP address of the web service which is running. Ideally, the web service and this application must connect to the same internet (IP address) so that they can access each other,

This field will be deleted in the future if the web service is deployed on any server.

Once a user is logged in it needs some permissions to be set up, for example, to send SMS and to access the location data, the user has to allow one time to settings. Then this application will be running in the background.

Now users can add friends and family numbers on the screen They can be deleted or more numbers can be added.

After that step, this application is ready to use.

Once a user will fall or any unusual shake will be detected, it will generate the alert message on the phone and give the user a chance to react if it is a false alarm or no accident has happened. The time span, in which it has to be answered, is configurable. Right now the timing is set to wait for 10 seconds for the user to take action or press ok if it is not an accident. If a user will not react after the configured set of time, the application will send an SMS to a family member about the accident along with the accident location address as a text and its coordinates with a google map link. and the tracking MAP link. At this time, information will be sent to the hospital too. Also, this application will notify nearby users about the accident location, this radius is also configurable, right now it is 1000 meters. So it notifies all the users nearby by 1000 meters to that accident location.

#### **4.2 Web Application (Intelligent Accident Detection IAD):**

The web application is only for the hospital admin so that they can track where an accident happened and they can mark it as done, it can only be marked by the hospital that received the case. The only requirement to use this system is that the user (hospital admin) has to register to the page by giving the hospital information so that they can see the accident information if it happens nearby. Accident information will come automatically in the hospital monitoring screen.

### **5 Testing**

Following are the steps are taken for the test case:

- 1- Run java application for web services.
- 2- Run the android application.
- 3- Open two emulators E1 and E2.

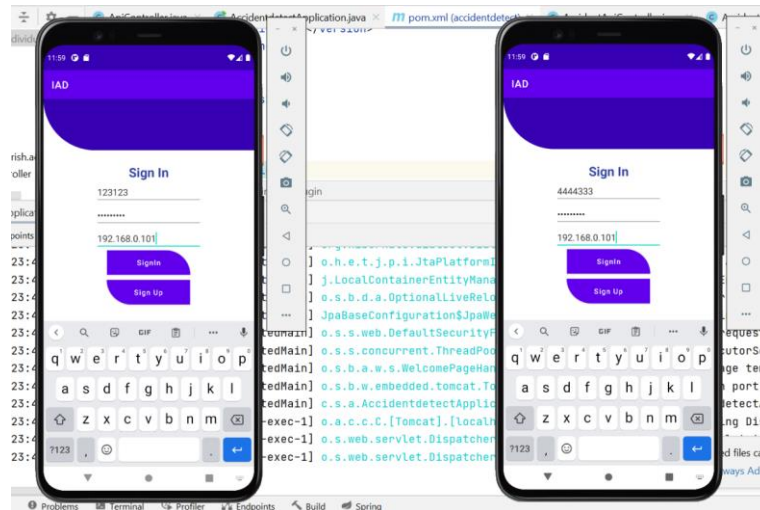


Figure 3- E1 and E2 Login

#### 4- Connect your android phone

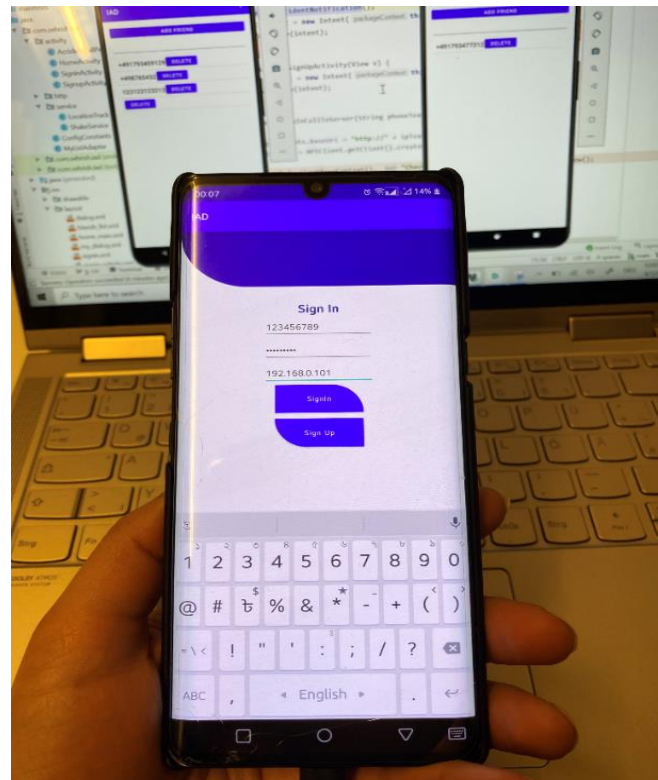


Figure 4- Andriod-Mobile Login

- 5- Now login into E1 with user1
- 6- Login to E2 with user2

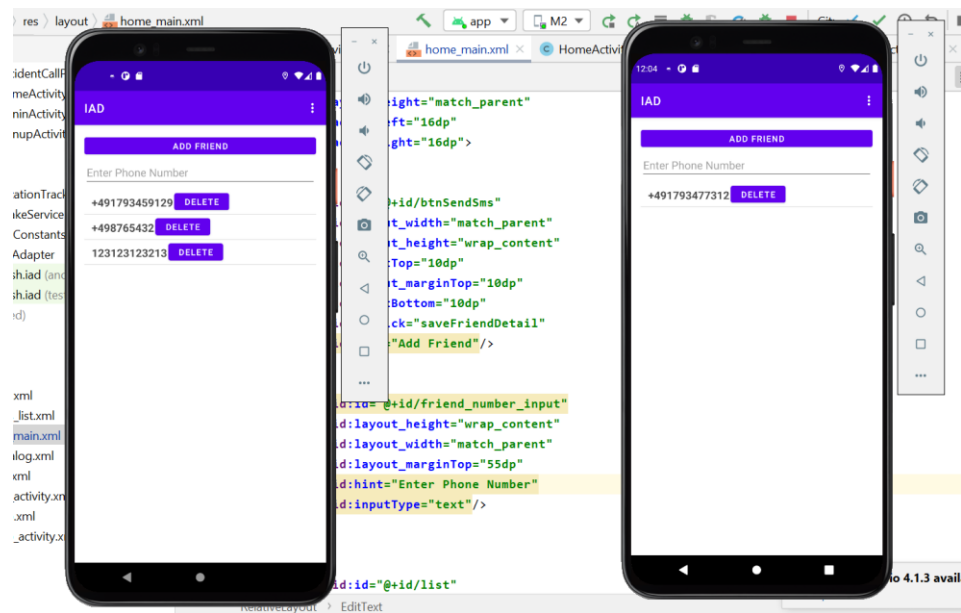


Figure 5 E1 and E2 Home Page

- 7- Login with your android phone with user3
- 8- Set the permissions on your phone as it will prompt all the steps respectively.

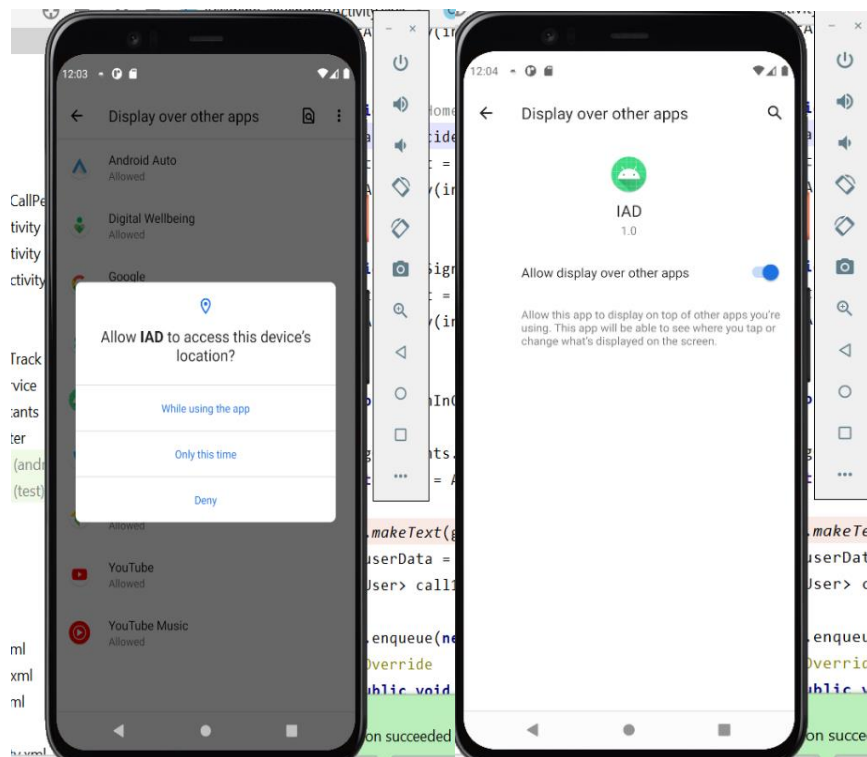


Figure 6(a) - Rights

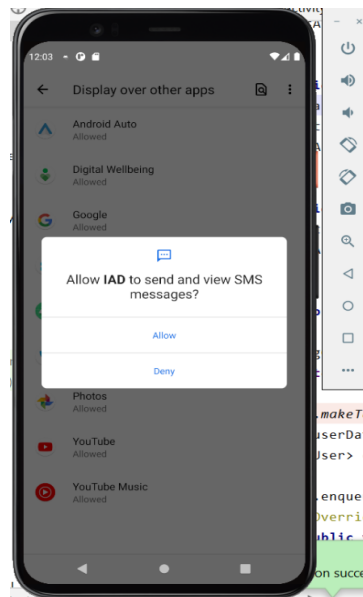


Figure 6(b) - Rights

- 9- As the emulator doesn't have a real sim or location, we have inserted dummy data for E1 and E2 for the current location.
- 10- The mobile device will fetch the current location.
- 11- Imagine your mobile is in the location as described in figure 7:

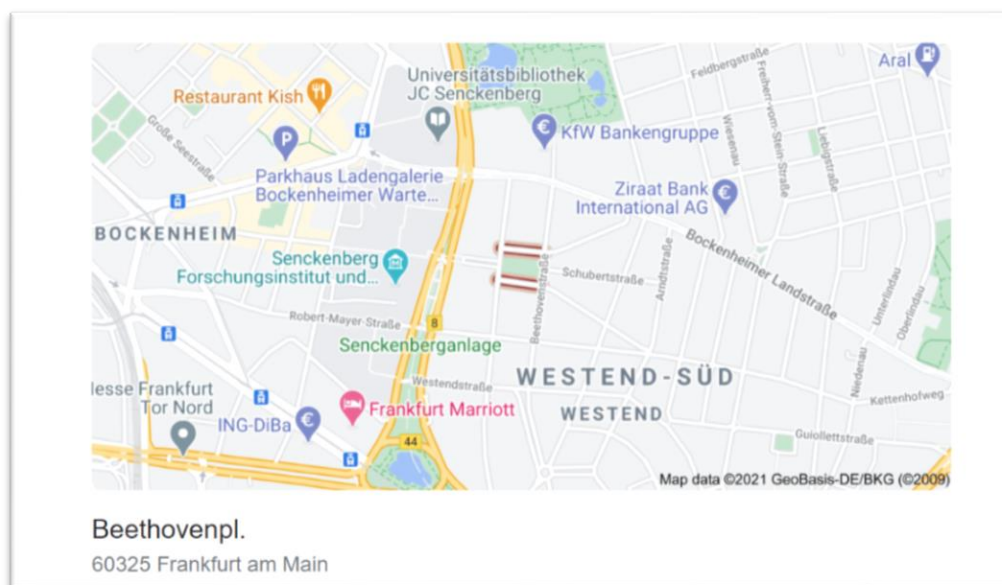



Figure 7 – Current Location [2]

- 12- For E1 and E2 we can set E1 nearby this area
- 13- E2 should be in other areas to test the complete scenario

location  Enter a SQL expression to filter results (use Ctrl+Space)

	123 id	created_date	lat	lon	123 user_id
134	7,050	2021-04-03 00:08:44	50.11848146	8.65579772	34
135	7,049	2021-04-03 00:08:29	50.11849084	8.65582157	34
136	7,048	2021-04-03 00:08:15	50.11844261	8.65567783	34
137	7,047	2021-04-03 00:03:53	50.1185063	8.6536323	1
138	7,046	2021-04-03 00:03:49	50.1162536	8.6545679	2

Figure 8 –Location Database

14- Now our devices are ready.

15- On the web page, we can open the hospital UI and login with the hospital which is nearby this area, see figure-9.

## User Registration - Sign In

E-mail:

Password:

Figure 9 – Hospital Login

16- Our complete test setup is ready.

17- Now imagine you are sitting on a chair (assume it is a car) and quick shake or throw your phone

18- Now you can see it will give a pop-up, see figure-10

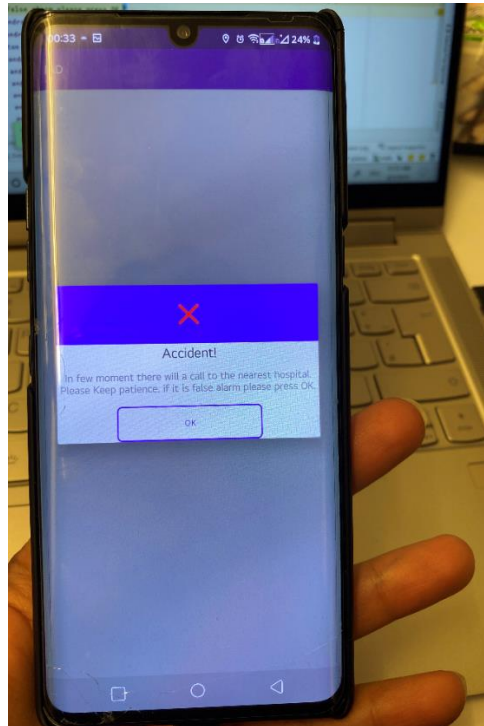


Figure 10 – Accident-Popup

- 19- If you can close the pop-up with 30 seconds (configurable) it will not do anything.
- 20- If you will not close the pop-up after the configured time (see point 19) it will send an SMS to the person you added to the list with the location of the accident (for testing that mobile user should be on the same internet).it will send SMS to all numbers configured on the screen of the mobile user, as shown in Figure-5.

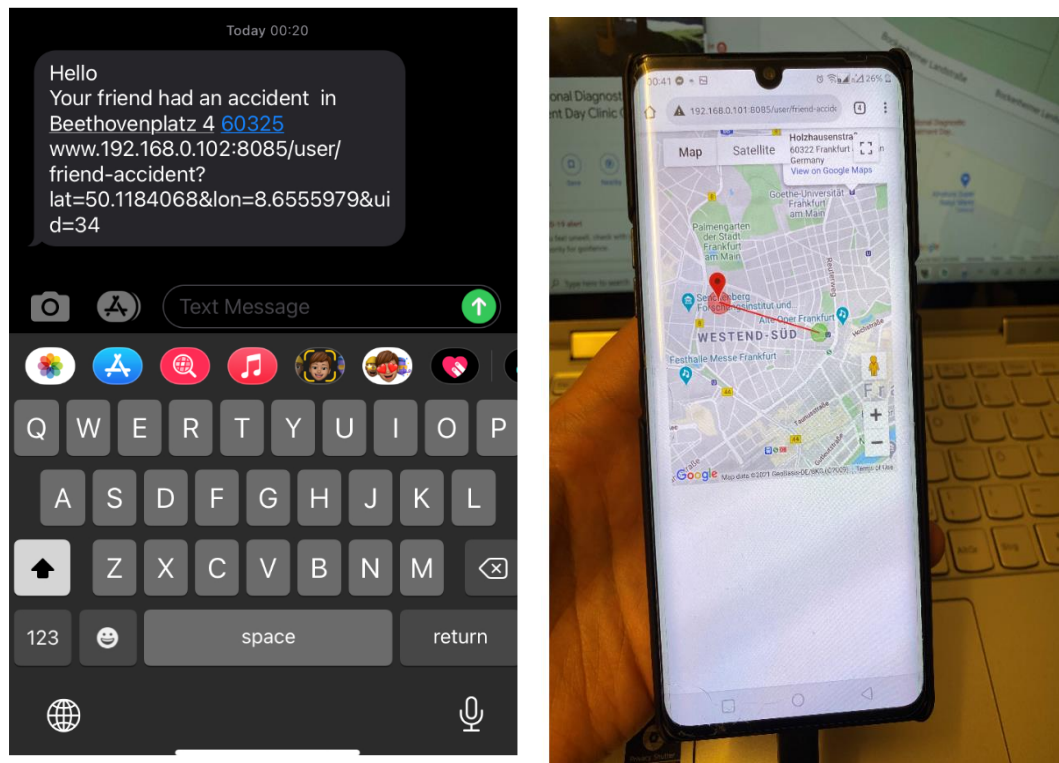


Figure 10 – SMS and Accident URL

21- E1 will receive a notification about the accident which happened with the address, while E2 will not as we set E1 nearby the accident area

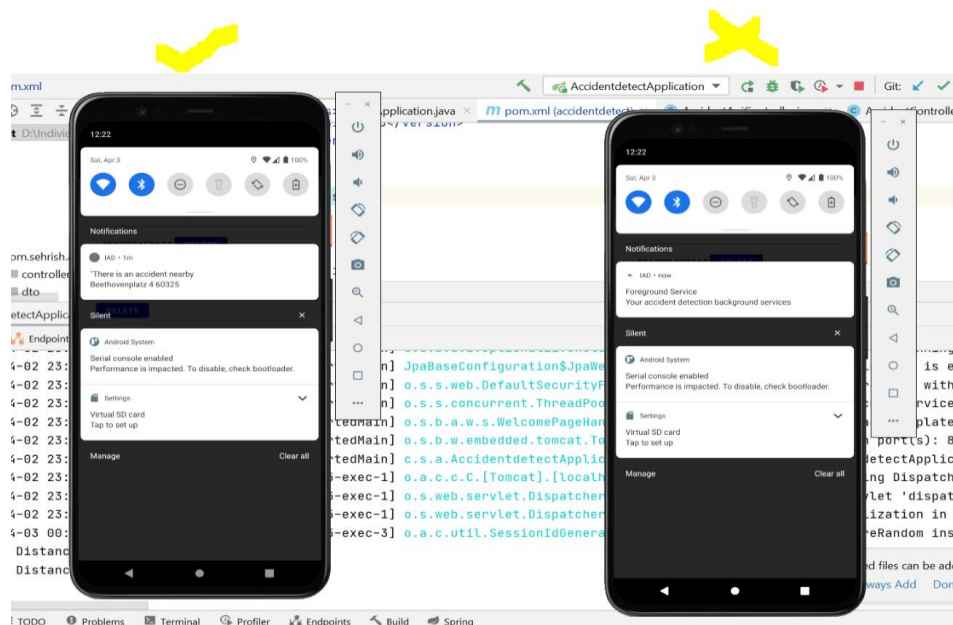
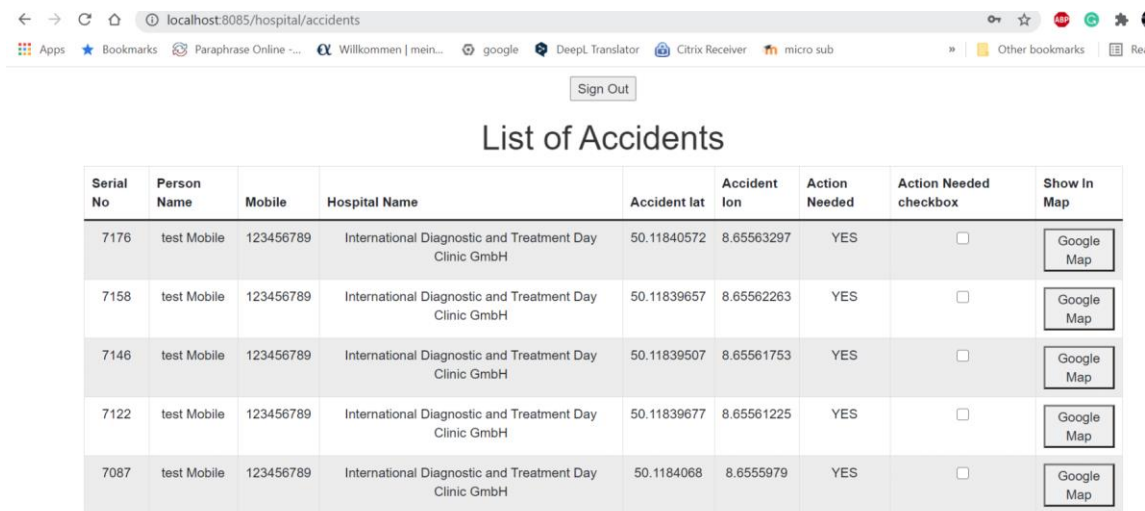


Figure 11 – Nearby User Notify



22- Now you can see the real-time monitoring on the hospital web, here you can see the accident information.



Serial No	Person Name	Mobile	Hospital Name	Accident lat	Accident lon	Action Needed	Action Needed checkbox	Show in Map
7176	test Mobile	123456789	International Diagnostic and Treatment Day Clinic GmbH	50.11840572	8.65563297	YES	<input type="checkbox"/>	Google Map
7158	test Mobile	123456789	International Diagnostic and Treatment Day Clinic GmbH	50.11839657	8.65562263	YES	<input type="checkbox"/>	Google Map
7146	test Mobile	123456789	International Diagnostic and Treatment Day Clinic GmbH	50.11839507	8.65561753	YES	<input type="checkbox"/>	Google Map
7122	test Mobile	123456789	International Diagnostic and Treatment Day Clinic GmbH	50.11839677	8.65561225	YES	<input type="checkbox"/>	Google Map
7087	test Mobile	123456789	International Diagnostic and Treatment Day Clinic GmbH	50.1184068	8.6555979	YES	<input type="checkbox"/>	Google Map

Figure 11 – Hospital Monitoring Screen

## 6 Future Development

This project was one of my best effort in a short time. I have tried my best to deliver the maximum amount of aspects of the project. However, I have delivered all the components I have been committed to. But there are much more possibilities to make it more efficient and more environment friendly

Following are my suggestion and thoughts for a future aspect of how we can make this product better:

- 1- The scope should not be restricted to a vehicle, it should be extended to different groups like pedestrians. As this proposed solution can work for any kind of transportation but pedestrian it might have different for example mobile dropping will have different speed.
- 2- Web service can be deployed to a remote server and use as a normal third-party service.
- 3- In an android phone there can be a configuration page where the user can configure basic parameters for example time to hold accident alarm screen, speed threshold.
- 4- It should be implemented on IOS as well for iPhone.
- 5- Research on real data from the past can help to train the system for more accuracy as for this project due to time constrain I have trained by few samples inside the car.
- 6- Accident detection can be more polished by trial and use more accurate calculations for accident shake differ than the normal one as for now it is achieved by few samples inside the car (could be research part)



The above points are from my vision that how can I see this product grow and how it can be better.

## **Conclusion**

This document provides complete detail about the "Android-based implementation of an emergency system". The idea behind this project was to try to provide a solution or the guideline for intelligent accident detection, as there is no such application that has been developed so far by using mobile. The car has some systems which provide this solution but have this functionality in mobile gives more benefit as we can identify and change the accident detection in future so that this solution will work better than the available functions in the car, as I believe by using mobile we can provide a better and practical solution to address this area.

Intelligent transportation is a huge field. This solution provides a solution toward the accident events on the road and opens a direction of research and realization of using this product as we do have products that detect human injury, like a smartwatch. But still many people are not aware of or do not have smartwatches, but a mobile is a device that almost everyone has. That was the main reason to have a project in this way.

## **References:**

1. [https://developer.android.com/guide/topics/sensors/sensors\\_motion](https://developer.android.com/guide/topics/sensors/sensors_motion)
2. <https://www.google.com/maps/place/Beethovenpl.+4,+60325+Frankfurt+am+Main/data=!4m2!3m1!1s0x47bd095ab2b0d153:0xcc570c511f195416?sa=X&ved=2ahUKEwiDuOGaqfnvAhUjgP0HHSKxBGAQ8gEwAHoECAMQAO>

## **Appendix**

### **Complete Project Code Files:**

<https://github.com/sehrishf/AccidentDetectionAndriod>

<https://github.com/sehrishf/AccidentDetectionJava>