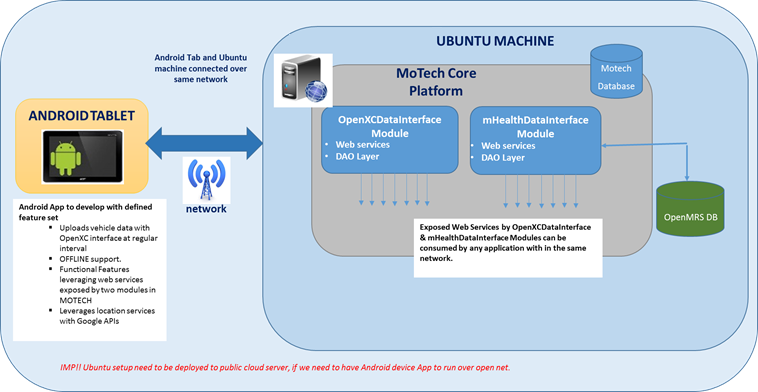
|  |
| --- |
| mHealth – App Feature set |

# Solution Overview

Diagram on overall integrated solution –



**Highlights on above framework –**

* Two modules named as **mHealthDataInterface** and **OpenXCDataInterface** are integrated in the MOTECH Backend framework.
* The MOTECH system deployed on a single server machine
* Both modules expose the APIs in form of web-services for other apps to consume – for vehicle data as well as for patient & health worked data.
* The Mobile App based on an Android device is written to interface with the MOTECH Backend via APIs exposed from both the new modules *mhealthDataInterface* and*OpenXCDataInterface*
* In the local development environment of HCL, both MOTECH server machine and Mobile device are connected to same network, which is internal network of HCL.
* ***To have the Mobile App running and accessing the MOTECH backend through public network internet, it will be necessary to deploy the MOTECH setup on a public cloud server (AWS, Azure,...)***

# Android App as a Demo Application

This app represents an end to end functional flow of some of the features of mHealth system that are being leveraged via the web interfaces that are provided by two modules developed as part of this project – OpenXCDataInterface Module and mHealthDataInterface Module.

**The App is supposed to run in a phone that is with health worker in the mHealth vehicle.**

The set of key features to be developed in Android App are mentioned in below sections –

## Startup Splash Screen and vehicle registration to backend

On startup app will show up a splash screen like below –



**Fig. Splash screen of App on start up**

The App will do the vehicle registration with OpenXCDataInterface module, which will be only first time of execution.

Here We are using the web service API ***registermHealthVehicle ( )*** from **OpenXCDataInterface** module for Register a vehicle.

## Offline Feature Support

The App will implement the OFFLINE mode support while it uploads the vehicle real-time data to MOTECH Backend.

**Key objective of feature is track the vehicle status from MOTECH server even for the duration when cellular network is lost at vehicle location.**

The App will keep monitoring on the cellular network status of phone. If signal is not available, vehicle data that is candidate for upload is put in a local cache of phone.

App will detect as cellular network is re-connected, on the same it will sync up with backend server via sending all the data packets that are in cache.

### Configuring the upload interval

App will allow selecting the upload interval (interval at which data packets will be uploaded to cloud server) on the startup screen for the first time.

Upload interval options can be 15 sec, 30 sec, 45 sec, and 60 sec. Once selected, app will not ask again for the same unless app data is cleared or reinstalled.

By default option is kept selected @ every 30 sec. per packet.

### Local cache capacity

The local cache on phone is created to hold a maximum defined no of packets; thus limiting the local cache capacity. At present it will cache a maximum of 120 packets. After cache is filled with all 120 packets, next packet will replace the oldest one in the cache.

As a notable point, if say upload interval is 30 sec. then 120 packets will be able to cache data for 1 Hr.

Thus feature will allow to track the vehicle for the last up to one hour duration (**assuming upload interval is selected as 30 sec**) whenever the cellular signal is re-stored. Of course if vehicle is having no signal for more than 1 hr, then tracking will be lost for additional duration.

Further the cache count and hence duration is easily customizable in the software.

In the source code of android client application, **Constants.java file has MAX\_ROW\_COUNT constant**

*public static final int MAX\_ROW\_COUNT = 120;*

*Developer just need to change the value of MAX\_ROW\_COUNT and build the new APK then.*

## Dashboard

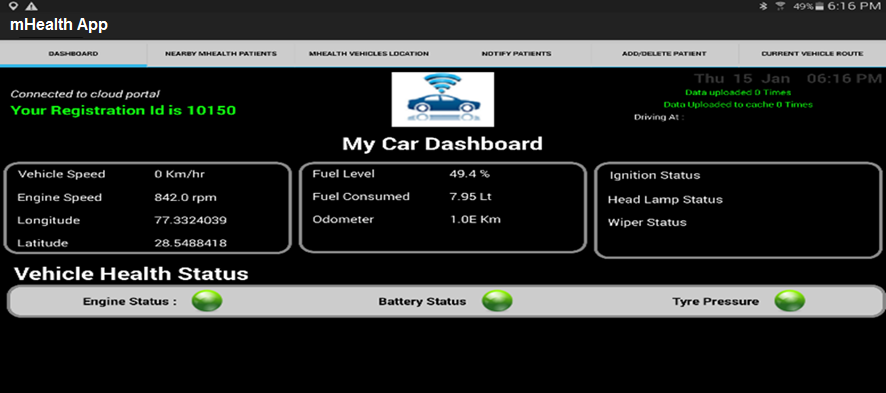
General Layout of the screen is Tab based layout. There are SIX Tabs developed at present –

* Dashboard
* Nearby mHealth Patients
* Nearby mHealth Vehicles
* Fetch Details of Patients & Notify Patients
* Add/Delete Patients
* Current Vehicle Route

Dashboard is the first Tab, which will display a snapshot of vehicle status (real time data through OpenXC interface) and upload status to server.

Here We are using the web service API ***uploadmHealthVehicleData ( )*** from **OpenXCDataInterface** module for Upload location parameter and vehicle speed.

Layout of screen is as below –



The Backend API that is being used in this screen is Data Upload API.

Each time the App will be started, it will prompt to choose the source for vehicle data streaming.

Below three options are supported –

1. Vehicle Live Feed Selection
2. Custom Trace file Selection
3. Default Trace file Selection

### Vehicle LIVE Stream

Application can be used in a live scenario, where it can be connected to a car and can stream the live vehicle data to the backend cloud, instead of using the trace file.

In case of **Vehicle Live Stream**, device running the app need to be connected to a real vehicle and paired up with VI device connecting to the OBD-II port present in the vehicle. When the VI device is connected to the vehicle and paired up over **Bluetooth channel** to the device running the app, VI device will start streaming the data to the App. Streaming data can be visualized on the dashboard screen and real time data of the vehicle will start uploading to the cloud. For location of the vehicle, GPS should be kept on so that it can provide Latitude and Longitude values.

### Default Trace File

As a simulation option, the app has the ability to playback a previously recorded OpenXC trace file. The default trace is embedded in the app by default.

### Choose a specific Trace file

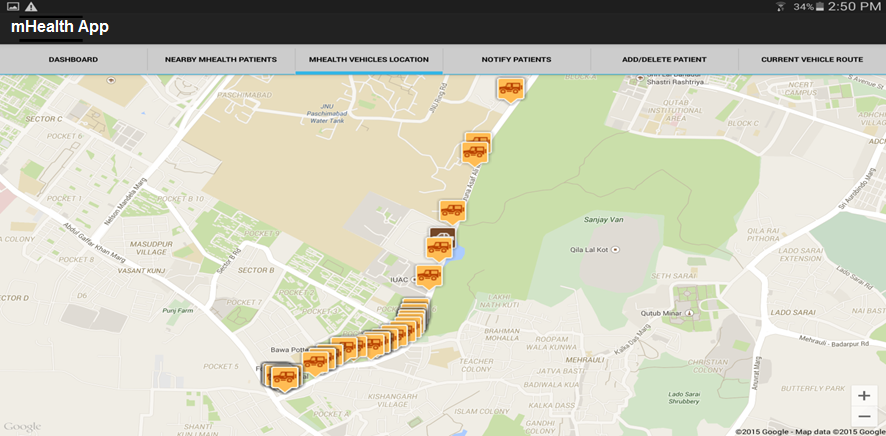
This option provides one to use any trace file that he has. Custom trace file can be selected in the mobile device itself from the file explorer.

## Vehicle Location tracking

Vehicle Location tracking is the feature that will display the location of current vehicle as well as other mHealth vehicles on the map.

Here we are using the web service exposed API ***getAllmHealthVehiclesLocation ( )*** from **OpenXCDataInterface** module . We get the list of all vehicle locations from this API, after that we use vehicle’s latitude, longitude to show up nearby all vehicles’ location to our vehicle.

A tentative layout of screen is as below –

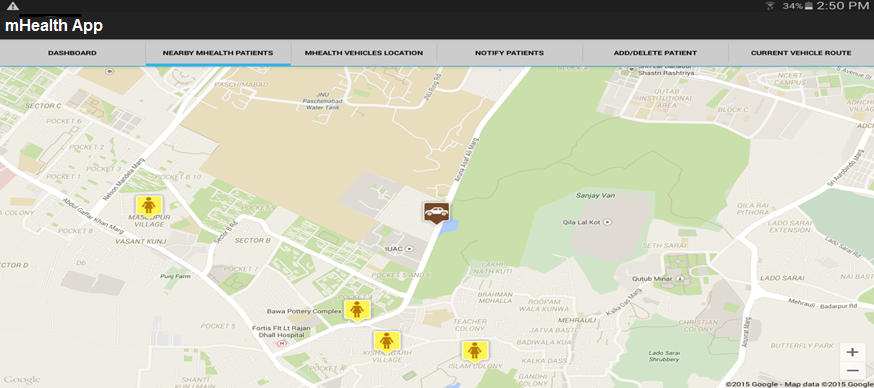
**

## Patient location tracking vs vehicle location

This will show up the registered patients that belong to the postal code location where vehicle is driving at present. That means it will show up nearby patients to vehicle, which will be probable set of patients whom health worker is going to visit.

Here We are using the web service API ***getPatientsByPostalCode ()*** from ***mHealthDataInterface*** module. We get patients all details from this API, after that we use patient’s address and postal code to show up nearby patients to vehicle.

A tentative layout of screen is as below –



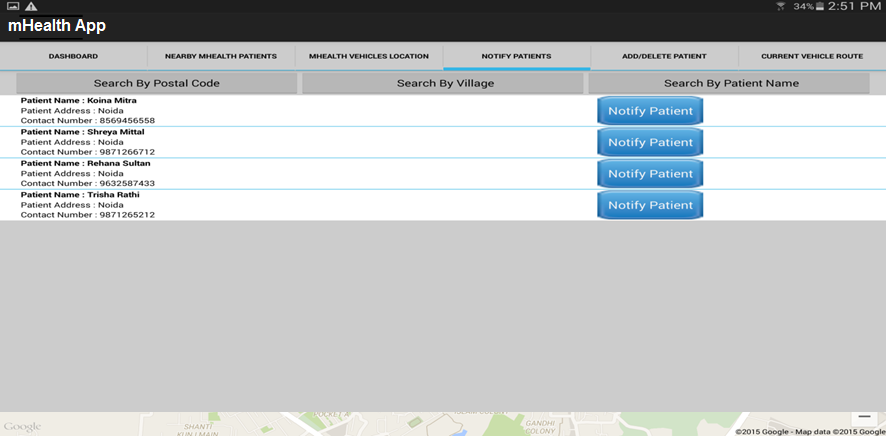
## Notification message to patients for Health Worker visit

This is an interesting feature where App will send an SMS message to registered mobile no of patient(s), whom health worker is going to visit shortly.

While being on the way, it will display the list of registered patients on current postal code. Health worker then can select some/all patients and send the message notification to them stating that HW is visiting shortly to them, for they could be available and ready for meeting.

Here We are using two web service API’s getPatientsByPostalCode ***(), getPatientsByVillage and getPatientsDetailByName()*** from ***mHealthDataInterface*** module. We get patients all details from these apis, after that App will send an SMS message to registered mobile no of patient(s).

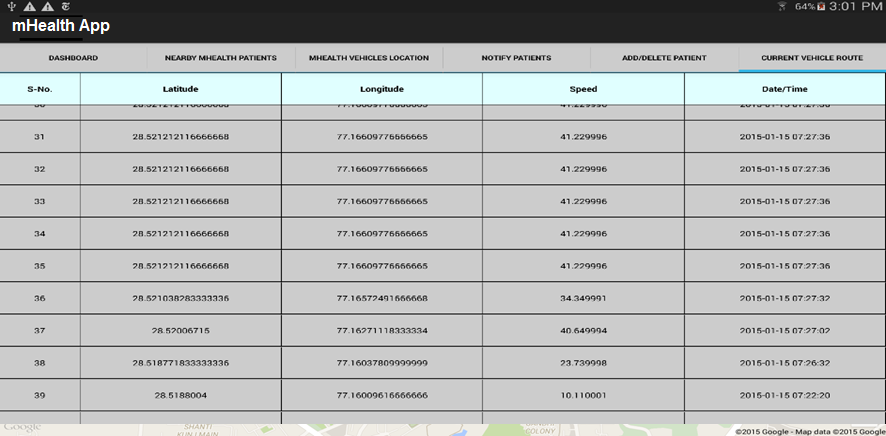
A tentative layout of screen is as below –



***As perquisite – we will use the SIM card service of cellular provider that is plugged on the Android phone/tablet.***

## Current Vehicle Route Data

This feature gives the facility of viewing the data that is being uploaded to the backend cloud server. At any time it fetches the data for the current vehicle. A layout of screen is as below –



The data packet that gets upload on the dashboard screen can be easily viewed here.

Latest 120 records are shown here. Most recent data feed displays on the top and then in descending order, the older ones are shown. This count is easily configurable on the server side code for mHealthDataInterface.

The Date/Time showed of the packet is the time when packet was uploaded from device to cloud server, in case the device is connected to a network.

If the network is unavailable and device is working under offline mode, then the packet will be uploaded to cache together with the time stamp and when the network regains, it will be uploaded. So, **all the date/ time are representing the real time actual values.**

**Date/Time is representing Standard GMT format, while uploading to the cloud. This will likely be different than the local time.**

# Perquisite for Demo/Test run of Solution

Below are the perquisites to run the Solution –

1. MOTECH backend setup to be deployed on a public network which could be AWS, Azure or any public cloud service provider, means it requires mHealthDataInterface and OpenXCDataInterface deployed and running on accessible network.
2. Android Device (phone or tablet)
   1. Connected to internet either via Wi-Fi or cellular network.
   2. Should have size 5” or more.
   3. For SMS notification, it must have active SIM connected
   4. Bluetooth enabled for LIVE streaming.
3. In car setup with VI device in case of running the app with vehicle live stream

**~~ End of doc ~~**