**RASPBIAN OS**

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

**Note:** Raspbian is not affiliated with the Raspberry Pi Foundation. Raspbian was created by a small, dedicated team of developers that are fans of the Raspberry Pi hardware, the educational goals of the Raspberry Pi Foundation and, of course, the Debian Project.

WLAN

A wireless LAN (or WLAN, for wireless local area network, sometimes referred to as LAWN, for local area wireless network) is one in which a mobile user can connect to a local area network (LAN) through a wireless (radio) connection. The IEEE 802.11 group of standards specify the technologies for wireless LANs. 802.11 standards use the Ethernet protocol and CSMA/CA (carrier sense multiple access with collision avoidance) for path sharing and include an encryption method, the Wired Equivalent Privacy algorithm.

High-bandwidth allocation for wireless will make possible a relatively low-cost wiring of classrooms in the United States. A similar frequency allocation has been made in Europe. Hospitals and businesses are also expected to install wireless LAN systems where existing LANs are not already in place.

Using technology from the Symbionics Networks, Ltd., a wireless LAN adapter can be made to fit on a Personal Computer Memory Card Industry Association (PCMCIA) card for a laptop or notebook computer.

Architecture[edit]

Stations[edit]

All components that can connect into a wireless medium in a network are referred to as stations (STA). All stations are equipped with wireless network interface controllers (WNICs). Wireless stations fall into one of two categories: wireless access points, and clients. Access points (APs), normally wireless routers, are base stations for the wireless network. They transmit and receive radio frequencies for wireless enabled devices to communicate with. Wireless clients can be mobile devices such as laptops, personal digital assistants, IP phones and other smartphones, or fixed devices such as desktops and workstations that are equipped with a wireless network interface.

Basic service set[edit]

The basic service set (BSS) is a set of all stations that can communicate with each other at PHY layer. Every BSS has an identification (ID) called the BSSID, which is the MAC address of the access point servicing the BSS.

There are two types of BSS: Independent BSS (also referred to as IBSS), and infrastructure BSS. An independent BSS (IBSS) is an ad hoc network that contains no access points, which means they can not connect to any other basic service set.

Extended service set[edit]

An extended service set (ESS) is a set of connected BSSs. Access points in an ESS are connected by a distribution system. Each ESS has an ID called the SSID which is a 32-byte (maximum) character string.

Distribution system[edit]

A distribution system (DS) connects access points in an extended service set. The concept of a DS can be used to increase network coverage through roaming between cells.

DS can be wired or wireless. Current wireless distribution systems are mostly based on WDS or MESH protocols, though other systems are in use.

## Types of wireless LANs[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=7" \o "Edit section: Types of wireless LANs)]

The [IEEE 802.11](https://en.wikipedia.org/wiki/IEEE_802.11) has two basic modes of operation: **infrastructure** and ***ad hoc*** mode. In *ad hoc* mode, mobile units transmit directly peer-to-peer. In infrastructure mode, mobile units communicate through an [access point](https://en.wikipedia.org/wiki/Wireless_access_point) that serves as a bridge to other networks (such as [Internet](https://en.wikipedia.org/wiki/Internet) or [LAN](https://en.wikipedia.org/wiki/Local_area_network)).

Since wireless communication uses a more open medium for communication in comparison to wired LANs, the 802.11 designers also included encryption mechanisms: [Wired Equivalent Privacy](https://en.wikipedia.org/wiki/Wired_Equivalent_Privacy) (WEP, now insecure), [Wi-Fi Protected Access](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access) (WPA, WPA2), to secure wireless computer networks. Many access points will also offer [Wi-Fi Protected Setup](https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup), a quick (but now insecure) method of joining a new device to an encrypted network.

### Infrastructure**[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=8" \o "Edit section: Infrastructure)]**

Most Wi-Fi networks are deployed in infrastructure mode.

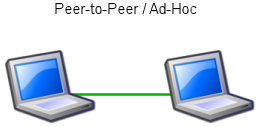
In infrastructure mode, a base station acts as a [wireless access point](https://en.wikipedia.org/wiki/Wireless_access_point) hub, and nodes communicate through the hub. The hub usually, but not always, has a wired or fiber network connection, and may have permanent wireless connections to other nodes.

Wireless access points are usually fixed, and provide service to their client nodes within range.

Wireless clients, such as laptops, smartphones etc. connect to the access point to join the network.

Sometimes a network will have a multiple access points, with the same 'SSID' and security arrangement. In that case connecting to any access point on that network joins the client to the network. In that case, the client software will try to choose the access point to try to give the best service, such as the access point with the strongest signal.

### Peer-to-peer**[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=9" \o "Edit section: Peer-to-peer)]**

[](https://en.wikipedia.org/wiki/File:Wlan_adhoc.png)

Peer-to-Peer or ad hoc wireless LAN

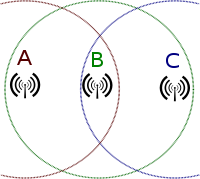
An [ad hoc network](https://en.wikipedia.org/wiki/Wireless_ad_hoc_network) (not the same as a [WiFi Direct network](https://en.wikipedia.org/wiki/Wi-Fi_Direct" \o "Wi-Fi Direct)[[4]](https://en.wikipedia.org/wiki/Wireless_LAN#cite_note-4)) is a network where stations communicate only peer to peer (P2P). There is no base and no one gives permission to talk. This is accomplished using the Independent Basic Service Set (IBSS).

A [WiFi Direct network](https://en.wikipedia.org/wiki/Wi-Fi_Direct" \o "Wi-Fi Direct) is another type of network where stations communicate peer to peer.

In a Wi-Fi P2P group, the group owner operates as an access point and all other devices are clients. There are two main methods to establish a group owner in the Wi-Fi Direct group. In one approach, the user sets up a P2P group owner manually. This method is also known as Autonomous Group Owner (autonomous GO). In the second method, also called negotiation-based group creation, two devices compete based on the group owner intent value. The device with higher intent value becomes a group owner and the second device becomes a client. Group owner intent value can depend on whether the wireless device performs a cross-connection between an infrastructure WLAN service and a P2P group, remaining power in the wireless device, whether the wireless device is already a group owner in another group and/or a received signal strength of the first wireless device.

A [peer-to-peer](https://en.wikipedia.org/wiki/Peer-to-peer) network allows wireless devices to directly communicate with each other. Wireless devices within range of each other can discover and communicate directly without involving central access points. This method is typically used by two computers so that they can connect to each other to form a network. This can basically occur in devices within a closed range.

If a signal strength meter is used in this situation, it may not read the strength accurately and can be misleading, because it registers the strength of the strongest signal, which may be the closest computer.

[](https://en.wikipedia.org/wiki/File:Wifi_hidden_station_problem.svg)

[Hidden node problem](https://en.wikipedia.org/wiki/Hidden_node_problem): Devices A and C are both communicating with B, but are unaware of each other

[IEEE 802.11](https://en.wikipedia.org/wiki/IEEE_802.11) defines the physical layer (PHY) and MAC (Media Access Control) layers based on [CSMA/CA](https://en.wikipedia.org/wiki/CSMA_CA) (Carrier Sense Multiple Access with Collision Avoidance). The 802.11 specification includes provisions designed to minimize collisions, because two mobile units may both be in range of a common access point, but out of range of each other.

### *Bridge***[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=10" \o "Edit section: Bridge)]**

''*A bridge can be used to connect networks, typically of different types. A wireless*[*Ethernet*](https://en.wikipedia.org/wiki/Ethernet)*bridge allows the connection of devices on a wired Ethernet network to a wireless network. The bridge acts as the connection point to the Wireless LAN.*

### Wireless distribution system**[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=11" \o "Edit section: Wireless distribution system)]**

*Main article:*[*Wireless Distribution System*](https://en.wikipedia.org/wiki/Wireless_Distribution_System)

A Wireless Distribution System enables the wireless interconnection of access points in an IEEE 802.11 network. It allows a wireless network to be expanded using multiple access points without the need for a wired backbone to link them, as is traditionally required. The notable advantage of WDS over other solutions is that it preserves the MAC addresses of client packets across links between access points.[[5]](https://en.wikipedia.org/wiki/Wireless_LAN#cite_note-5)

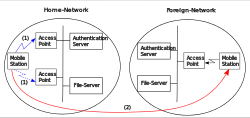
An access point can be either a main, relay or remote base station. A main base station is typically connected to the wired Ethernet. A relay base station relays data between remote base stations, wireless clients or other relay stations to either a main or another relay base station. A remote base station accepts connections from wireless clients and passes them to relay or main stations. Connections between "clients" are made using MAC addresses rather than by specifying IP assignments.

All base stations in a Wireless Distribution System must be configured to use the same radio channel, and share WEP keys or WPA keys if they are used. They can be configured to different service set identifiers. WDS also requires that every base station be configured to forward to others in the system as mentioned above.

WDS may also be referred to as repeater mode because it appears to bridge and accept wireless clients at the same time (unlike traditional bridging). It should be noted, however, that throughput in this method is halved for all clients connected wirelessly.

When it is difficult to connect all of the access points in a network by wires, it is also possible to put up access points as repeaters.

## Roaming[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=12" \o "Edit section: Roaming)]

[](https://en.wikipedia.org/wiki/File:Roaming01.svg)

Roaming among Wireless Local Area Networks

There are two definitions for wireless LAN roaming:

1. **Internal Roaming:** The Mobile Station (MS) moves from one access point (AP) to another AP within a home network because the signal strength is too weak. An authentication server (RADIUS) performs the re-authentication of MS via 802.1x (e.g. with [PEAP](https://en.wikipedia.org/wiki/Protected_Extensible_Authentication_Protocol)). The billing of QoS is in the home network. A Mobile Station roaming from one access point to another often interrupts the flow of data among the Mobile Station and an application connected to the network. The Mobile Station, for instance, periodically monitors the presence of alternative access points (ones that will provide a better connection). At some point, based on proprietary mechanisms, the Mobile Station decides to re-associate with an access point having a stronger wireless signal. The Mobile Station, however, may lose a connection with an access point before associating with another access point. In order to provide reliable connections with applications, the Mobile Station must generally include software that provides session persistence.[[6]](https://en.wikipedia.org/wiki/Wireless_LAN#cite_note-6)
2. **External Roaming:** The MS (client) moves into a WLAN of another Wireless Internet Service Provider (WISP) and takes their services (Hotspot). The user can independently of his home network use another foreign network, if this is open for visitors. There must be special authentication and billing systems for mobile services in a foreign network.

Applications[edit]

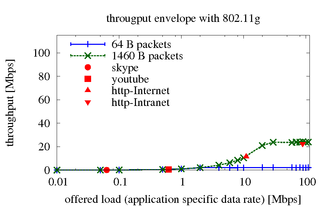
Wireless LANs have a great deal of applications. Modern implementations of WLANs range from small in-home networks to large, campus-sized ones to completely mobile networks on airplanes and trains.

Users can access the Internet from WLAN hotspots in restaurants, hotels, and now with portable devices that connect to 3G or 4G networks. Oftentimes these types of public access points require no registration or password to join the network. Others can be accessed once registration has occurred and/or a fee is paid.

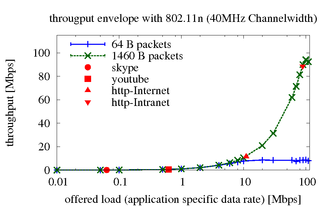
Existing Wireless LAN infrastructures can also be used to work as indoor positioning systems with no modification to the existing hardware.

Performance and throughput[[edit](https://en.wikipedia.org/w/index.php?title=Wireless_LAN&action=edit&section=14" \o "Edit section: Performance and throughput)]

WLAN, organised in various layer 2 variants (IEEE 802.11), has different characteristics. Across all flavours of 802.11, maximum achievable throughputs are either given based on measurements under ideal conditions or in the layer 2 data rates. This, however, does not apply to typical deployments in which data are being transferred between two endpoints of which at least one is typically connected to a wired infrastructure and the other endpoint is connected to an infrastructure via a wireless link.

[](https://en.wikipedia.org/wiki/File:Throughputenvelope80211g.png)

Graphical representation of [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) application specific (UDP) performance envelope 2.4 GHz band, with 802.11g

[](https://en.wikipedia.org/wiki/File:ThroughputEnvelope11n.png)

Graphical representation of [Wi-Fi](https://en.wikipedia.org/wiki/Wi-Fi) application specific (UDP) performance envelope 2.4 GHz band, with 802.11n with 40 MHz

This means that typically data frames pass an 802.11 (WLAN) medium and are being converted to 802.3 (Ethernet) or vice versa.

Due to the difference in the frame (header) lengths of these two media, the packet size of an application determines the speed of the data transfer. This means that an application which uses small packets (e.g. VoIP) creates a data flow with a high overhead traffic (e.g. a low goodput).

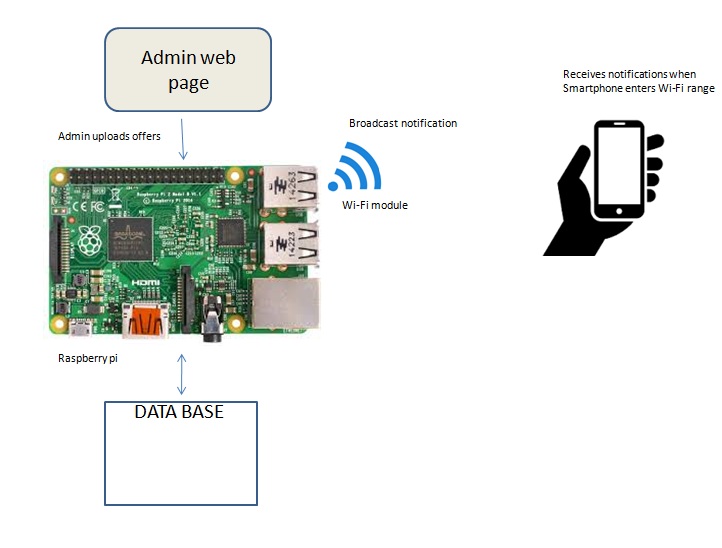
Other factors which contribute to the overall application data rate are the speed with which the application transmits the packets (i.e. the data rate) and the energy with which the wireless signal is received.

The latter is determined by distance and by the configured output power of the communicating devices.[[7]](https://en.wikipedia.org/wiki/Wireless_LAN#cite_note-7)[[8]](https://en.wikipedia.org/wiki/Wireless_LAN#cite_note-8)

Same references apply to the attached throughput graphs which show measurements of UDP throughput measurements. Each represents an average (UDP) throughput (the error bars are there, but barely visible due to the small variation) of 25 measurements.

Each is with a specific packet size (small or large) and with a specific data rate (10 kbit/s – 100 Mbit/s). Markers for traffic profiles of common applications are included as well. This text and measurements do not cover packet errors but information about this can be found at above references. The table below shows the maximum achievable (application specific) UDP throughput in the same scenarios (same references again) with various difference WLAN (802.11) flavours. The measurement hosts have been 25 meters apart from each other; loss is again ignored.

**SYSTEM DESIGN 17/05/2016**



An efficient advertising platform helps in effective marketing of the products. It should be cost effective and more responsive than the traditional methods. The design of the proposed model for the effective advertising platform is shown in figure.

**Tech Stack:**

1. Raspberry Pi

2. WLAN:

i. Wireless router.

ii. Wi-Fi module.

3. Android Smart Phone

**Raspberry Pi**: Raspberry pi is a credit card sized computer. The Raspberry Pi is connected the open Wi-Fi network of the shop. We use the Raspberry Pi to host the servers, maintain databases and to broadcast relevant data in the WLAN network which is explained in detail.

**PHP** is used as our server-side scripting language. Two important jobs are carried out by PHP.

1. **Pushing data into the database**: When the shopkeeper has to post about the offers or any information regarding the shop, he does it through a HTML web page - form which is designed for him. The data entered in the form are sent to the PHP using a POST request. PHP then takes the data and it feeds into the shopkeeper’s “offers” database.
2. **Echoing a JSON for the Android phone**: The next job of PHP is used to retrieve all the data from the database and broadcast / echo it as a JSON (JavaScript-Object Notation).

Example of a JSON :

**{**

"id"**:** **1,**

"name"**:** **"**T-Shirt**",**

"price"**:** **1000/-,**

**}**

**MySQL** is used to maintain the shopkeeper’s databases. All the databases required must also be created in the Raspberry Pi. The “offers” database which has all the offers is created. The owner pushes the relevant offers into this “offers” database created using MySQL. PHP and MySQL work hand in hand to handle all the data.

Finally the databases are queried and the data bits obtained regarding the offers are coined into JSONs using PHP and it is **broadcasted/echoed**. This JSON is available publicly in the WLAN network which can be accessed by any device. In this case an Android mobile phone.

**WLAN:** A wireless local area network (WLAN) is a wireless computer network that links two or more devices using a wireless distribution method.

Here the data from the Raspberry Pi needs to be linked with the android phone. To achieve this we use :

**A wireless router** is a device that performs the functions of a **router** and also includes the functions of **a wireless** access point. It is used to provide access to the Internet or a private computer network.

This provides a wireless access point to connect to. Computers / Mobile phones can be connected to this access point.

A Wi-Fi module is plugged into the Raspberry-Pi so that the Raspberry Pi can connect to the wireless access point and can be accessed by other devices connected to the same WLAN.

In the design phase the software requirements for the proposed model is as follows:

**Android Smart Phone:** Android studio is used to build an android application which is installed in the user’s smartphone.

A broadcast receiver is used to monitor Wi-Fi changes and when the Wi-Fi is connected to an appropriate Wi-Fi network an android service is triggered.

The service receives JSON data from the server set up in the Raspberry pi. JSON parsing is a technique to retrieve the data from a JSON which is in the “name: value” pair format.

The JSON broadcasted over WLAN is parsed and then a notification is generated using Pending Intents in Android. The notification notifies the user about the offers. He can click on the notification to open for the details of the offers and other details which is broadcasted.

HTML is used to design the user interface for the shopkeeper to push the offers into the database.

**IMPLEMENTATION CODE 17/05/2016 :**

**MAINVOLLEYACTIVITY.**

**Initialising the activity.**

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//Initializing Views

recyclerView = (RecyclerView) findViewById(R.id.recyclerView);

recyclerView.setHasFixedSize(true);

layoutManager = new LinearLayoutManager(this);

recyclerView.setLayoutManager(layoutManager);

//Initializing our superheroes list

items = new ArrayList<>();

}

**Add Volley Request on application start :**

@Override

protected void onStart() {

//Showing a progress dialog

loading = ProgressDialog.show(this,"Loading Data", "Please wait...",false,false);

super.onStart();

mQueue = CustomVolleyRequestQueue.getInstance(this.getApplicationContext())

.getRequestQueue();

String url = getString(R.string.pi\_ip)+"TEST/getdata.php";

final CustomJSONObjectRequest jsonRequest = new CustomJSONObjectRequest(Request.Method

.GET, url,

new JSONObject(), this, this);

jsonRequest.setTag(REQUEST\_TAG);

mQueue.add(jsonRequest);

}

**JSON PARSING :**

**Parsing the JSON obtained from the raspbnerry pi**

private void parseData(Object response) {

String rpiResponse = response.toString();

try {

JSONObject jObj = new JSONObject(rpiResponse);

JSONArray jArr = jObj.getJSONArray("MegaMart");

for (int i = 0; i < jArr.length(); i++) {

JSONObject arJ = jArr.getJSONObject(i);

Item item = new Item();

item.setHeader(arJ.getString("header"));

item.setDescription(arJ.getString("description"));

item.setNew\_price(arJ.getString("newprice"));

item.setOld\_price(arJ.getString("oldprice"));

item.setImage(getString(R.string.pi\_ip) +"TEST/"+arJ.getString("url"));

Log.d("offers",getString(R.string.pi\_ip) +arJ.getString("url"));

items.add(item);

}

} catch (JSONException e) {

e.printStackTrace();

}

//Finally initializing our adapter

adapter = new CardAdapter(items, this);

//Adding adapter to recyclerview

recyclerView.setAdapter(adapter);

}

**BROADCAST RECEIVER CLASS :**

**Triggered WIFI STATE CHANGED in mobile:**

public void onReceive(Context context, Intent intent) {

NetworkInfo info = intent.getParcelableExtra(WifiManager.EXTRA\_NETWORK\_INFO);

if (info != null && info.isConnected()) {

// Do your work.

Log.d("ConnectionState", "Connected");

Intent i = new Intent(context,MyService.class);

context.startService(i);

// e.g. To check the Network Name or other info:

WifiManager wifiManager = (WifiManager) context.getSystemService(Context.WIFI\_SERVICE);

WifiInfo wifiInfo = wifiManager.getConnectionInfo();

String ssid = wifiInfo.getSSID();

} else {

Log.d("ConnectionState", "Disconnected");

Intent i = new Intent(context,MyService.class);

context.stopService(i);

}

}

**SERVICE CLASS:**

**Service is triggered by BroadcastReceiver**

public class MyService extends Service implements Response.Listener, Response.ErrorListener {

public static final String REQUEST\_TAG = "MyService";

private RequestQueue mQueue;

String companyGlobal =null;

MediaPlayer mp;

@Override

public void onDestroy() {

super.onDestroy();

Log.d("Service", "Destroyed");

Toast.makeText(getApplicationContext() , "Destroyed" , Toast.LENGTH\_LONG).show();

if (mQueue != null) {

mQueue.cancelAll(REQUEST\_TAG);

}

}

@Override

public void onCreate() {

super.onCreate();

Log.d("Service","Kappa");

mp = MediaPlayer.create(this, R.raw.tweeters);

setUpRequest();

Toast.makeText(getApplicationContext() , "creaated" , Toast.LENGTH\_LONG).show();

}

private void setUpRequest() {

mQueue = CustomVolleyRequestQueue.getInstance(this.getApplicationContext())

.getRequestQueue();

String url = getString(R.string.pi\_ip)+"/TEST/getdata.php";

final CustomJSONObjectRequest jsonRequest = new CustomJSONObjectRequest(Request.Method

.GET, url,

new JSONObject(), this, this);

jsonRequest.setTag(REQUEST\_TAG);

mQueue.add(jsonRequest);

}

@Nullable

@Override

public IBinder onBind(Intent intent) {

return null;

}

@Override

public void onResponse(Object response) {

//Json Parsing

JSONObject job = null;

try {

job = new JSONObject(response.toString());

} catch (JSONException e) {

e.printStackTrace();

}

String dateTime = null,message = null, company = null;

try {

company = job.getString("company");

companyGlobal = company;

} catch (JSONException e) {

e.printStackTrace();

}

Log.d("response Json" , dateTime + "LOL" + message + " " + company);

//Show Notification

popUpNotification();

}

private void popUpNotification() {

mp.start();

Notification mNotification;

Intent resultIntent = new Intent(this, MainVolleyActivity.class);

// Because clicking the notification opens a new ("special") activity, there's

// no need to create an artificial back stack.

PendingIntent resultPendingIntent =

PendingIntent.getActivity(

this,

0,

resultIntent,

PendingIntent.FLAG\_UPDATE\_CURRENT

);

NotificationManager mNotificationManager = (NotificationManager) this.getSystemService(Context.NOTIFICATION\_SERVICE);

Notification.Builder builder = new Notification.Builder(this)

.setSmallIcon(R.mipmap.ic\_launcher)

.setAutoCancel(true)

.setContentTitle(companyGlobal)

.setContentText("AMAZING SUPER MEGA OFFERS HURRY !!")

.setContentIntent(resultPendingIntent);

//Get current notification

mNotification = builder.getNotification();

//Show the notification

mNotificationManager.notify(1, mNotification);

}

@Override

public void onErrorResponse(VolleyError error) {

}

}

**PHP :**

**PI echoing the Offers json for the android phones :**

**<?php**

**$con = mysqli\_connect("localhost", "root", "sqlmy" ,"roleDb");**

**$sql = "select \* from offersAvailable";**

**$result = mysqli\_query($con,$sql);**

**$i =0;**

**if($result -> num\_rows >0){**

**while($row = $result->fetch\_assoc()){**

**$myObject["dateTime"] = $row["dateTime"];**

**$myObject["name"] = $row["name"];**

**$myObject["header"] = $row["header"];**

**$myObject["description"] = $row["description"];**

**$myObject["oldprice"] = $row["oldprice"];**

**$myObject["newprice"] = $row["newprice"];**

**$myObject["url"] = $row["url"];**

**$arrayOb["MegaMart"][$i++] = $myObject;**

**}**

**$arrayOb["company"] = "MegaMart";**

**}**

**echo json\_encode($arrayOb);**

**mysqli\_close($con);**

**?>**

**Populating the database with offers ( By the owner ) :**

**<?php**

**$dateTime=$\_POST['dateTime'];**

**$name=$\_POST['name'];**

**$description = $\_POST['description'];**

**$header=$\_POST['header'];**

**$oldprice=$\_POST['oldprice'];**

**$newprice=$\_POST['newprice'];**

**$url=$\_POST['url'];**

**$con = mysqli\_connect("localhost", "root", "sqlmy" ,"roleDb");**

**$sql = "insert into offersAvailable values('$dateTime','$name','$header','$description','$oldprice','$newprice','$url')";**

**mysqli\_query($con,$sql);**

**echo "NOT SURE .$name";**

**?>**