**CHAPTER 1**

**INTRODUCTION**

Advertising is a means by which communication with the users of a product or service occurs. Advertising is always present, though people may not be aware of it. In present world, using every possible media advertising get’s its message through. This is done via television, print (newspapers, magazines, journals etc), radio, press, internet, events, direct selling, hoardings, posters, mailers, contests, clothes, sounds, visuals sponsorships, and even people (endorsements). However today’s advertising platforms are relatively expensive in terms of creative, production and airtime costs making it difficult for targeting your market. A professional has to be hired to design an efficient, well-crafted and effective script.

Our proposed idea helps in making the advertising more efficient and cost effective. This assists in the targeted marketing strategy. This report describes how the proposed idea is more effective than traditional advertising methods.

**CHAPTER**

**TECHNOLOGY USED**

To successfully develop any application, various software technologies and platforms have to be selected and interconnected such they seamlessly work along with each other. The language used to develop an application an application plays a crucial role and should be selected according to the requirements of the application. Real time applications require procedure oriented languages such as C, while flexible object oriented languages such as java can be used for other applications. Our application, which has been developed using java language, runs on the android platform. Various other technologies can be used depending on its ease of use and requirement of the application.

**Raspberry pi**

The Raspberry Pi is a series of credit card–sized single-board computers developed in the United Kingdom by the Raspberry Pi Foundation with the intent to promote the teaching of basic computer science in schools and developing countries. It is capable of doing everything a desktop does like browsing video streaming etc. It has a Broadcom BCM2836 Arm7 Quad Core Processor powered Single Board Computer running at 900MHz,1GB RAM so you can now run bigger and more powerful applications, Micro SD slot for storing information and loading your operating systems. You can now provide up to 1.2 AMP to the USB port – enabling you to connect more power hungry USB devices directly to the Raspberry PI (This feature requires a 2Amp micro USB Power Supply) and 10/100 Ethernet Port to quickly connect the Raspberry Pi to the Internet. Most boards have between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like I2C. Some models have an RJ45 Ethernet port and the Pi 3 has on board WiFi 802.11n and Bluetooth.

The system on a chip (SoC) used in the first generation Raspberry Pi is somewhat equivalent to the chip used in older smartphones (such as iPhone, 3G, 3GS). The Raspberry Pi is based on the Broadcom BCM2835 SoC, which includes an 700 MHz ARM1176JZF-S processor, VideoCore IV graphics processing unit (GPU), and RAM. It has a Level 1 cache of 16 KB and a Level 2 cache of 128 KB. The Level 2 cache is used primarily by the GPU. The SoC is stacked underneath the RAM chip, so only its edge is visible.The Raspberry Pi 2 uses a Broadcom BCM2836 SoC with a 900 MHz 32-bit quad-core ARM Cortex-A7 processor, with 256 KB shared L2 cache.

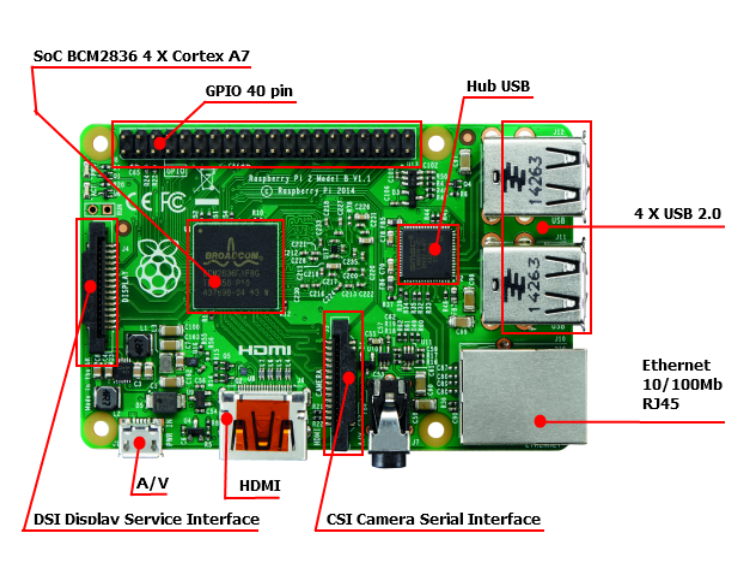


Fig 1. Architecture of raspberry pi 2

**General purpose input-output (GPIO) connector**

RPi A+, B+, 2B and Zero GPIO J8 have a 40-pin pinout.Model 3 has 40 pins as well, but someone will need to confirm that the pin layout is the same as its predecessor. Models A and B have only the first 26 pins.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **GPIO#** | **2nd func.** | **Pin#** |  | **Pin#** | **2nd func.** | **GPIO#** |
|  | +3.3 V | 1 |  | 2 | +5 V |  |
| 2 | SDA1 (I2C) | 3 |  | 4 | +5 V |  |
| 3 | SCL1 (I2C) | 5 |  | 6 | GND |  |
| 4 | GCLK | 7 |  | 8 | TXD0 (UART) | 14 |
|  | GND | 9 |  | 10 | RXD0 (UART) | 15 |
| 17 | GEN0 | 11 |  | 12 | GEN1 | 18 |
| 27 | GEN2 | 13 |  | 14 | GND |  |
| 22 | GEN3 | 15 |  | 16 | GEN4 | 23 |
|  | +3.3 V | 17 |  | 18 | GEN5 | 24 |
| 10 | MOSI (SPI) | 19 |  | 20 | GND |  |
| 9 | MISO (SPI) | 21 |  | 22 | GEN6 | 25 |
| 11 | SCLK (SPI) | 23 |  | 24 | CE0\_N (SPI) | 8 |
|  | GND | 25 |  | 26 | CE1\_N (SPI) | 7 |
| *(RPi 1 Models A and B stop here)* | | | | | | |
| EEPROM | ID\_SD | 27 |  | 28 | ID\_SC | EEPROM |
| 5 | N/A | 29 |  | 30 | GND |  |
| 6 | N/A | 31 |  | 32 |  | 12 |
| 13 | N/A | 33 |  | 34 | GND |  |
| 19 | N/A | 35 |  | 36 | N/A | 16 |
| 26 | N/A | 37 |  | 38 | Digital IN | 20 |
|  | GND | 39 |  | 40 | Digital OUT | 21 |

Model B rev. 2 also has a pad (called P5 on the board and P6 on the schematics) of 8 pins offering access to an additional 4 GPIO connections.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Function** | **2nd func.** | **Pin#** |  | **Pin#** | **2nd func.** | **Function** |
| N/A | +5 V | 1 |  | 2 | +3.3 V | N/A |
| GPIO28 | GPIO\_GEN7 | 3 |  | 4 | GPIO\_GEN8 | GPIO29 |
| GPIO30 | GPIO\_GEN9 | 5 |  | 6 | GPIO\_GEN10 | GPIO31 |
| N/A | GND | 7 |  | 8 | GND | N/A |

Models A and B provide GPIO access to the ACT status LED using GPIO 16. Models A+ and B+ provide GPIO access to the ACT status LED using GPIO 47, and the power status LED using GPIO 35.

**Operating systems**

The Raspberry Pi primarily uses Linux-kernel-based operating systems.

The ARM11 chip at the heart of the Pi (first generation models) is based on version 6 of the ARM. The primary supported operating system is Raspbian, although it is compatible with many others. The current release of Ubuntu supports the Raspberry Pi 2, while Ubuntu, and several popular versions of Linux, do not support the older Raspberry Pi 1 that runs on the ARM11. Raspberry Pi 2 can also run the Windows 10 IoT Core operating system, while no version of the Pi can run traditional Windows. The Raspberry Pi 2 currently also supports OpenELEC and RISC OS.

Raspbian – is maintained independently of the Foundation based on the Debian ARM hard-float (armhf) architecture port originally designed for ARMv7 and later processors (with Jazelle RCT/ThumbEE and VFPv3), compiled for the more limited ARMv6 instruction set of the Raspberry Pi. A minimum size of 4 GB SD card is required for the Raspbian images provided by the Raspberry Pi Foundation. There is a Pi Store for exchanging programs.

**Driver APIs**

Raspberry Pi can use a VideoCore IV GPU via a binary blob, which is loaded into the GPU at boot time from the SD-card, and additional software, that initially was closed source. This part of the driver code was later released. However, much of the actual driver work is done using the closed source GPU code. Application software use calls to closed source run-time libraries (OpenMax, OpenGL ES or OpenVG) which in turn calls an open source driver inside the Linux kernel, which then calls the closed source VideoCore IV GPU driver code. The API of the kernel driver is specific for these closed libraries. Video applications use OpenMAX, 3D applications use OpenGL ES and 2D applications use OpenVG which both in turn use EGL. OpenMAX and EGL use the open source kernel driver in turn.

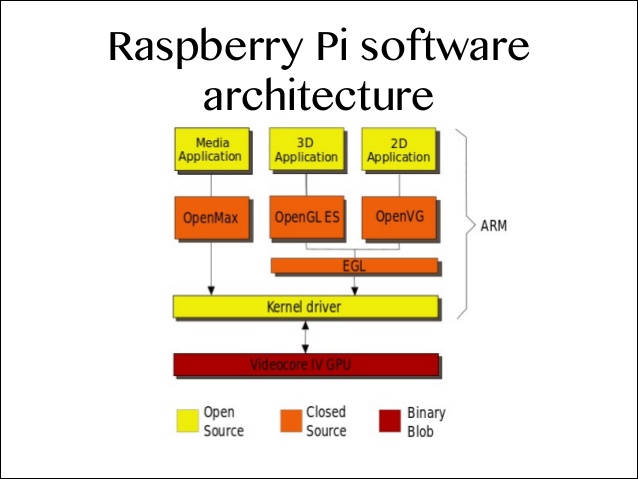


Fig 2. Raspberry pi software architecture

**Android**

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as Smartphone and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input.

**Android SDK**

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later. As of March 2015, the SDK is not available on Android itself, but the software development is possible by using specialized Android applications.

Until around the end of 2014, the officially supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, and NetBeans IDE also supports Android development via a plugin. As of 2015, Android Studio, made by Google and powered by IntelliJ, is the official IDE; however, developers are free to use others. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

**Java standards**

Obstacles to development include the fact that Android does not use established Java standards, that is, Java SE and ME. This prevents compatibility between Java applications written for those platforms and those written for the Android platform. Android only reuses the Java language syntax and semantics, but it does not provide the full class libraries and APIs bundled with Java SE or ME. However, there are multiple tools in the market from companies such as Myriad Group and UpOnTek that provide Java ME to Android conversion services.

**Java**

Java is a set of computer software and specifications developed by Sun Microsystems, which was later acquired by the Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While they are less common than standalone Java applications, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages.

Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages, including Ada, JavaScript, Python, and Ruby. In addition, several languages have been designed to run natively on the JVM, including Scala, Clojure and Groovy. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after Smalltalk and Objective-C. Java eschews certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM.

The Java platform is a suite of programs that facilitate developing and running programs written in the Java programming language. A Java platform will include an execution engine (called a virtual machine), a compiler and a set of libraries; there may also be additional servers and alternative libraries that depend on the requirements. Java is not specific to any processor or operating system as Java platforms have been implemented for a wide variety of hardware and operating systems with a view to enable Java programs to run identically on all of them. Different platforms target different classes of device and application domains.

**Java Virtual Machine**

The heart of the Java platform is the concept of a "virtual machine" that executes Java bytecode programs. This bytecode is the same no matter what hardware or operating system the program is running under. There is a JIT (Just In Time) compiler within the Java Virtual Machine, or JVM. The JIT compiler translates the Java bytecode into native processor instructions at run-time and caches the native code in memory during execution.

The use of bytecode as an intermediate language permits Java programs to run on any platform that has a virtual machine available. The use of a JIT compiler means that Java applications, after a short delay during loading and once they have "warmed up" by being all or mostly JIT-compiled, tend to run about as fast as native programs. Since JRE version 1.2, Sun's JVM implementation has included a just-in-time compiler instead of an interpreter.

Although Java programs are cross-platform or platform independent, the code of the Java Virtual Machines (JVM) that execute these programs is not. Every supported operating platform has its own JVM.

**Extensible Markup Language (XML)**

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards.

The design goals of XML emphasize simplicity, generality and usability across the Internet.It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services. XML has come into common use for the interchange of data over the Internet. IETF RFC 7303 gives rules for the construction of Internet Media Types for use when sending XML. It also defines the media types application/xml and text/xml, which say only that the data is in XML, and nothing about its semantics. The use of text/xml has been criticized as a potential source of encoding problems and it has been suggested that it should be deprecated.

By definition, an XML document is a string of characters. Almost every legal Unicode character may appear in an XML document. The characters making up an XML document are divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup either begin with the character < and end with a >, or they begin with the character & and end with a; Strings of characters that are not markup are content. However, in a CDATA section, the delimiters <![CDATA [and]]> are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element is classified as markup.

XML documents may begin by declaring some information about themselves, as in the following example:

<? xml version="1.0" encoding="UTF-8"?>

In addition to being well-formed, an XML document may be valid. This means that it contains a reference to a Document Type Definition (DTD), and that its elements and attributes are declared in that DTD and follow the grammatical rules for them that the DTD specifies.

XML processors are classified as validating or non-validating depending on whether or not they check XML documents for validity. A processor that discovers a validity error must be able to report it, but may continue normal processing.

**Apache**

The Apache HTTP Server, is the world's most used web server software. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites.

Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. Most commonly used on a Unix-like system (usually Linux), the software is available for a wide variety of operating systems besides Unix, including eComStation, Microsoft Windows, NetWare, OpenVMS, OS/2, and TPF. Released under the Apache License, Apache is free and open-source software.

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP. Popular authentication modules include mod\_access, mod\_auth, mod\_digest, and mod\_auth\_digest, the successor to mod\_digest. A sample of other features include Secure Sockets Layer and Transport Layer Security support (mod\_ssl), a proxy module (mod\_proxy), a URL rewriting module (mod\_rewrite), custom log files (mod\_log\_config), and filtering support (mod\_include and mod\_ext\_filter).

Instead of implementing a single architecture, Apache provides a variety of MultiProcessing Modules (MPMs), which allow Apache to run in a process-based, hybrid (process and thread) or event-hybrid mode, to better match the demands of each particular infrastructure. This implies that the choice of correct MPM and the correct configuration is important. Where compromises in performance need to be made, the design of Apache is to reduce latency and increase throughput, relative to simply handling more requests, thus ensuring consistent and reliable processing of requests within reasonable time-frames.

**PHP**

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive backronym PHP: Hypertext Preprocessor.

PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management system and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

The PHP interpreter only executes PHP code within its delimiters. Anything outside its delimiters is not processed by PHP, although non-PHP text is still subject to control structures described in PHP code. The most common delimiters are <?php to open and ?> to close PHP sections. The shortened form <? also exists. PHP stores integers in a platform-dependent range, either a 64-bit or 32-bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behavior is different from other programming languages.

Integer variables can be assigned using decimal (positive and negative), octal, hexadecimal, and binary notations. Floating point numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation. The null data type represents a variable that has no value; NULL is the only allowed value for this data type. PHP defines a large array of functions in the core language and many are also available in various extensions; these functions are well documented in the online PHP documentation. Basic object-oriented programming functionality was added in PHP 3 and improved in PHP 4. This allowed for PHP to gain further abstraction, making creative tasks easier for programmers using the language.

**MySQL**

MySQL is an open-source relational database management system (RDBMS). MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Free-software open-source projects that require a full-featured database management system often use MySQL. Applications that use the MySQL database include: TYPO3, MODx, Joomla, WordPress, phpBB, MyBB, Drupal and other software.

MySQL is a database system used on the web. MySQL is a database system that runs on a server. MySQL is ideal for both small and large applications. MySQL is very fast, reliable, and easy to use. MySQL uses standard SQL. MySQL compiles on a number of platforms.

**WLAN**

A wireless local area network (WLAN) is a wireless computer network that links two or more devices using a wireless distribution method (often spread-spectrum or OFDM radio) within a limited area such as a home, school, computer laboratory, or office building. This gives users the ability to move around within a local coverage area and still be connected to the network, and can provide a connection to the wider Internet. Most modern WLANs are based on IEEE 802.11 standards, marketed under the Wi-Fi brand name.

Wireless LANs have become popular in the home due to ease of installation and use, and in commercial complexes offering wireless access to their customers; often for free. 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.

The 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 that serves as a bridge to other networks (such as Internet or LAN).

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 (WEP, now insecure), Wi-Fi Protected Access (WPA, WPA2), to secure wireless computer networks. Many access points will also offer Wi-Fi Protected Setup, a quick (but now insecure) method of joining a new device to an encrypted network.

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.

**Infrastructure**

Most Wi-Fi networks are deployed in infrastructure mode.In infrastructure mode, a base station acts as a 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**

Peer-to-Peer or ad hoc wireless LAN.

An ad hoc network (not the same as a WiFi Direct network[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 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 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.

IEEE 802.11 defines the physical layer (PHY) and MAC (Media Access Control) layers based on 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.

**CHAPTER 3**

**IMPLEMENTATION**

The section describes the implementation of the proposed advertising model.

**Android application part**

An android app is created and is to be installed by the customer in his smartphone to get the notification about the offers. The app is designed to get the notification about the offers whenever the customer enters the shop’s Wi-Fi range. Volley library is used for asynchronous background threads. Android volley is a networking library was introduced to make networking calls much easier, faster without writing tons of code. By default all the volley network calls works asynchronously, so we don’t have to worry about using asynctask anymore.

Volley comes with lot of features. Some of them are

1. Request queuing and prioritization

2. Effective request cache and memory management

3. Extensibility and customization of the library to our needs

4. Cancelling the requests

**Broadcast receiver**

The data (offers) in the raspberry pi is broadcasted through the Wi-Fi module. In the android phone a BroadcastReceiver is used to check the Wi-Fi state. Broadcast Receivers simply respond to broadcast messages from other applications or from the system itself. These messages are sometime called events or intents. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.

**Receiver Lifecycle**

A BroadcastReceiver object is only valid for the duration of the call to **onReceive(Context, Intent)**. Once your code returns from this function, the system considers the object to be finished and no longer active.

This has important repercussions to what you can do in an **onReceive(Context, Intent)** implementation: anything that requires asynchronous operation is not available, because you will need to return from the function to handle the asynchronous operation, but at that point the BroadcastReceiver is no longer active and thus the system is free to kill its process before the asynchronous operation completes.

In particular, you may not show a dialog or bind to a service from within a BroadcastReceiver. For the former, you should instead use the NotificationManager API. For the latter, you can use **Context.startService()** to send a command to the service.

**Process Lifecycle**

A process that is currently executing a BroadcastReceiver (that is, currently running the code in its **onReceive(Context, Intent)** method) is considered to be a foreground process and will be kept running by the system except under cases of extreme memory pressure.

Once you return from **onReceive(),** the BroadcastReceiver is no longer active, and its hosting process is only as important as any other application components that are running in it. This is especially important because if that process was only hosting the BroadcastReceiver (a common case for applications that the user has never or not recently interacted with), then upon returning from **onReceive()** the system will consider its process to be empty and aggressively kill it so that resources are available for other more important processes.

This means that for longer-running operations you will often use a Service in conjunction with a BroadcastReceiver to keep the containing process active for the entire time of your operation.

**JSON**

Whenever a Wi-Fi connection is detected, a service is made to run on the background to pick up the data from the raspberry pi. This data is in the form of JSONs. JSON is a lightweight data-interchange format. JSON uses JavaScript syntax, but the JSON format is text only, just like XML. Text can be read and used as a data format by any programming language.

The JSON format is syntactically identical to the code for creating JavaScript objects. Because of this similarity, instead of using a parser (like XML does), a JavaScript program can use standard JavaScript functions to convert JSON data into native JavaScript objects.

JSON Example:

{"employees":[  
    {"firstName":"John", "lastName":"Doe"},  
    {"firstName":"Anna", "lastName":"Smith"},  
    {"firstName":"Peter", "lastName":"Jones"}  
]}

The data got is populated into Recycler Views in android with custom adapters and getters and setters. Many apps need to display collections of the same type (such as messages, contacts, images, or songs); often, this collection is too large to fit on the screen, so the collection is presented in a small window that can smoothly scroll through all items in the collection. RecyclerView is an Android widget that displays a collection of items in a list or a grid, enabling the user to scroll through the collection.

**Recyclerview**

RecyclerView introduces an additional level of abstraction between the **RecyclerView.Adapter** and **RecyclerView.LayoutManager** to be able to detect data set changes in batches during a layout calculation. This saves LayoutManager from tracking adapter changes to calculate animations. It also helps with performance because all view bindings happen at the same time and unnecessary bindings are avoided.

For this reason, there are two types of position related methods in RecyclerView:

* layout position: Position of an item in the latest layout calculation. This is the position from the LayoutManager's perspective.
* adapter position: Position of an item in the adapter. This is the position from the Adapter's perspective.

These two positions are the same except the time between dispatching adapter.notify\* events and calculating the updated layout.

Methods that return or receive \*LayoutPosition\* use position as of the latest layout calculation (e.g. **getLayoutPosition(), findViewHolderForLayoutPosition(int)**). These positions include all changes until the last layout calculation. You can rely on these positions to be consistent with what user is currently seeing on the screen. For example, if you have a list of items on the screen and user asks for the 5th element, you should use these methods as they'll match what user is seeing.

The other set of position related methods are in the form of \*AdapterPosition\*. (e.g. **getAdapterPosition(), findViewHolderForAdapterPosition(int)**) You should use these methods when you need to work with up-to-date adapter positions even if they may not have been reflected to layout yet. For example, if you want to access the item in the adapter on a ViewHolder click, you should use **getAdapterPosition().** Beware that these methods may not be able to calculate adapter positions if **notifyDataSetChanged**() has been called and new layout has not yet been calculated. For this reasons, you should carefully handle NO\_POSITION or null results from these methods.

When writing a **RecyclerView.LayoutManager** you almost always want to use layout positions whereas when writing an **RecyclerView.Adapter**, you probably want to use adapter positions.

**Website interface**

The shopkeeper has to upload the offers using the admin page or the dashboard in the raspberry pi. The raspberry pi contains a sql database where all the offers are stored. The admin page or the dashboard is created using the HTML, CSS and bootstrap. The offers entered here are then uploaded to the database using the post method of the php. A php server is running on the raspberry pi and it uses which contains the database that includes all the offers uploaded.

A Wi-Fi module connected to the raspberry pi is used to broadcast the offers to the customer’s smartphones. Whenever the customers come into the Wi-Fi range of the shop the customer’s smartphones gets the notification of the offers. An android application is installed in the customers Smartphone’s to get offers which is broadcasted through the Wi-Fi. Figure 3 shows the website interface of the proposed model.

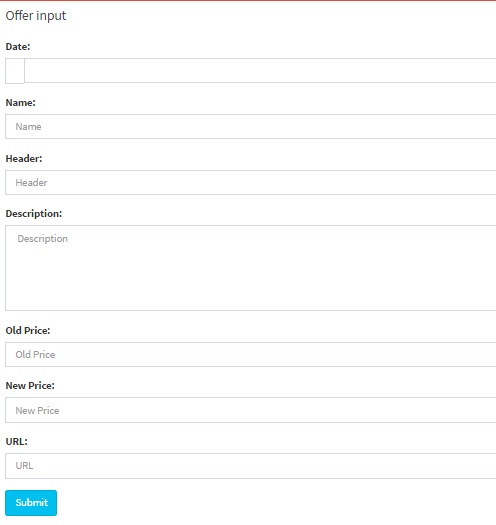


Fig 3. Website interface