BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE (BITS)-PILANI

A REPORT ON - Deriving LOGIC to validate ANDSF POLICY for USER EQUIPMENT

Project Report 1 Prepared for Partial Fulfillment of the Practice School-I Course No: BITS C221/BITS C231/BITS C241

Prepared by:

Name (s) of the Student (s)	ID. No (s)	Discipline (s)
ANURAG PANDA	2013A7PS129H	B.E.(Hons) Computer Science
ASHISH KUMAR	2013B3A7629G	M.Sc.(Hons) Economics B.E.(Hons) Computer Science
PARUL GOYAL	2013B4A7807H	M.Sc.(Hons) Mathematics B.E.(Hons) Computer Science

6/18/2015

AT

ELITECORE TECHNOLOGIES PVT. LTD.

A Practice School –I Station of Ahmedabad, Gujarat

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

(22th May 2015 – 16th July 2015)

Acknowledgements

We are highly indebted to our college, BITS Pilani, for providing us with this excellent learning opportunity which greatly helped us in honing our skills. We would also like thank Mr. Nikhil Jain, CEO, Elitecore Technologies Pvt. Ltd. for their guidance and constant supervision as well as for providing necessary information regarding the report and for their support in pursuing the project.

We extend our thanks to Mr. Mohit Gupta and Ms. Devanshi Sheth, for coordinating PS-I with our institution. We owe our profound gratitude to our project head Mr. Sumit Pandya and project guides, Mr. Virat Sinh Parmar who took keen interest in our project work and guided us all along till now.

We are thankful and fortunate enough to get constant encouragement, support and guidance from our instructor Dr. Debdulal Thakur and co-instructors Mr. Parth and Mr. Dhruvesh.

We would like to heartily thank all the employees in the Elitecore, especially the HR department for making our internship here smooth and problem-free.

The contribution of the peer group, as reviewers is also indispensable in completing our project.

Anurag Panda
Ashish Kumar
Parul Goyal

Abstract Sheet

Station	Elitecore Technologies Pvt. Ltd.
Centre	Ahmedabad
Duration	54 days
Date of Start	22-May 2015
Date of Submission	19-June 2015
Title of the Project	Deriving LOGIC to validate ANDSF POLICY for USER EQUIPMENT

Name of the Student (s)	ID No (s)	Discipline (s) of the Student (s)
Anurag Panda	2013A7PS129H	B.E.(Hons) Computer Science
Ashish Kumar	2013B3A7629G	M.Sc.(Hons) Economics B.E.(Hons) Computer Science
Parul Goyal	2013B4A7807H	M.Sc.(Hons) Mathematics B.E.(Hons) Computer Science

Name of the Expert (s)	Designation of the Expert (s)
Sumit Pandya	
Virat Sinh Parmar	Senior Software Engineer
Brijesh Mistry	Senior Software Engineer

Name of the PS Faculty: Dr. Debdulal Thakur, Assistant Professor, Department of Economics, BITS-Pilani, K.K. Birla Goa Campus

Key Words: Hotspot 2.0, Wifi Monetization, Android, ANDSF, ANDI, XML, Parsing, OMA DM, SyncML, Eliteconnect, NotificationManager, UE, MO, Policy.

Project Areas: Android Application Development, JAVA, ANDSF technology

Abstract: This report presents a strategic solution to mobile data offloading between 3GPP and non-3GPP access networks.

Signature of the Student (s)	Signature of the PS Faculty
Date:	Date:

Contents

1.1. *Introduction*

1.1.1. Course Description

Practice School-I is an industrial exposure program of eight weeks duration. It is a 5 Unit course which introduces the students to the real life work environment, which cannot be simulated in the classroom. Practice School-I provides a comprehensive first exposure to professional workplace, to learn organization structure and function, to develop personality traits, and to enhance communication and presentation (oral and written) skills.

OBJECTIVE

Practice School is an educational innovation seeking to link industry experience with university instruction. The objectives of PS are to :

(i) meet the rapidly changing needs and challenges of a professional workplace

(ii) (ii) enable students to acquire learning by applying the knowledge and skills they possess, in unfamiliar, open-ended real life situations

(iii) (iii) Bear an economic relevance to society.

These objectives are achieved by bringing the reality of the world of work into the process of education. PS creates the required setting for experiential and cooperative learning and education, by providing students with an opportunity to work on relevant assignments, under the guidance of professional experts and under the supervision of faculty. Consequently, Practice School serves as a platform that facilitates and promotes partnership and intellectual exchange between academia and industry.

SALIENT FEATURES

The scope or salient features of this PS-1 program are as follows:

(i) Institutionalized linkage between University and Industry

(ii) Applicable to all degree programs

(iii) Integral part of the curriculum

(iv) Student involvement in real-life projects

(v) Continuous Internal Evaluation system

(vi) Monitoring and evaluation by resident faculty member

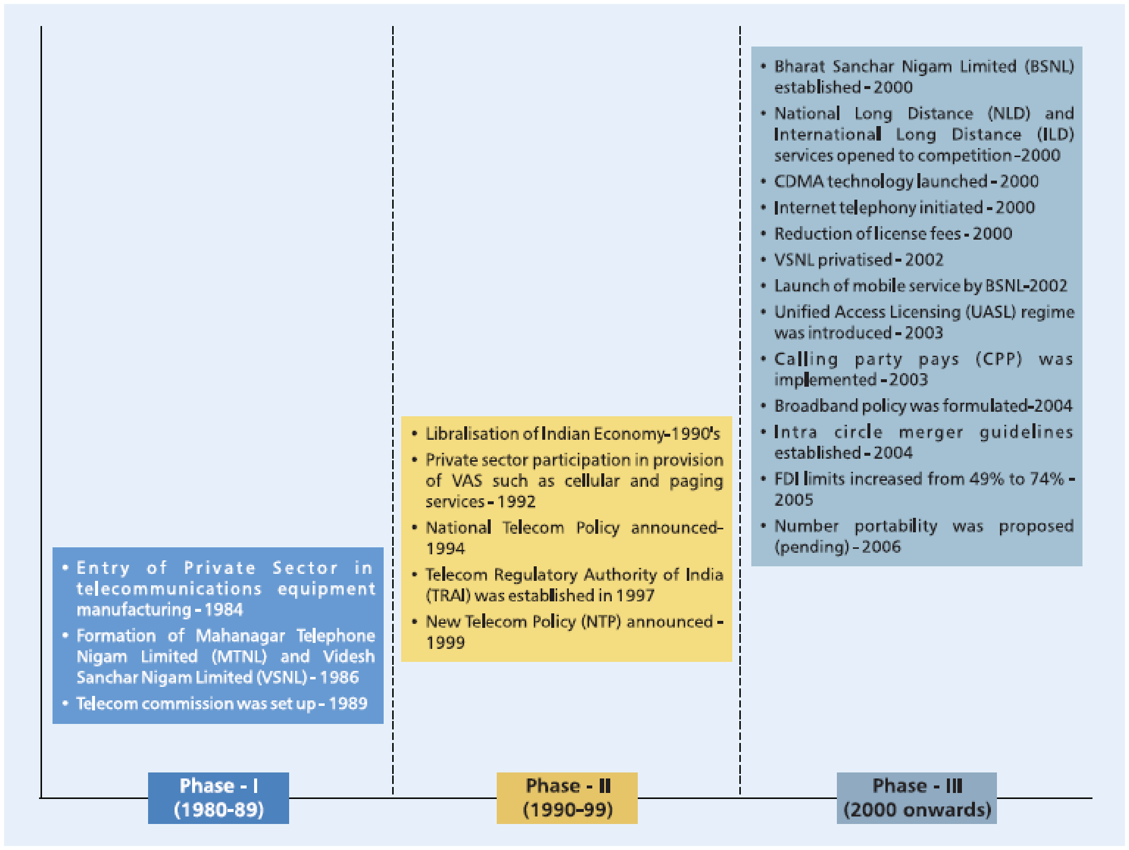
1.1.2. Industry Analysis – The Telecom Industry

In today's information age, the telecommunication industry has a vital role to play. Considered as the backbone of industrial and economic development, the industry has been aiding delivery of voice and data services at rapidly increasing speeds, and thus, has been revolutionising human communication.

Although the Indian telecom industry is one of the fastest-growing industries in the world, the current teledensity or telecom penetration is extremely low when compared with global standards. India's teledensity of 77.58% in December 2014 is the second highest in the world. Further, the urban teledensity is over 80%, while rural teledensity is less than 20%, and this gap is increasing. As majority of the population resides in rural areas, it is important that the government takes steps to improve rural teledensity. No doubt the government has taken certain policy initiatives, which include the creation of the Universal Service Obligation Fund, for improving rural telephony. These measures are expected to improve the rural teledensity and bridge the rural-urban gap in teledensity.

Evolution of Telecom Industry

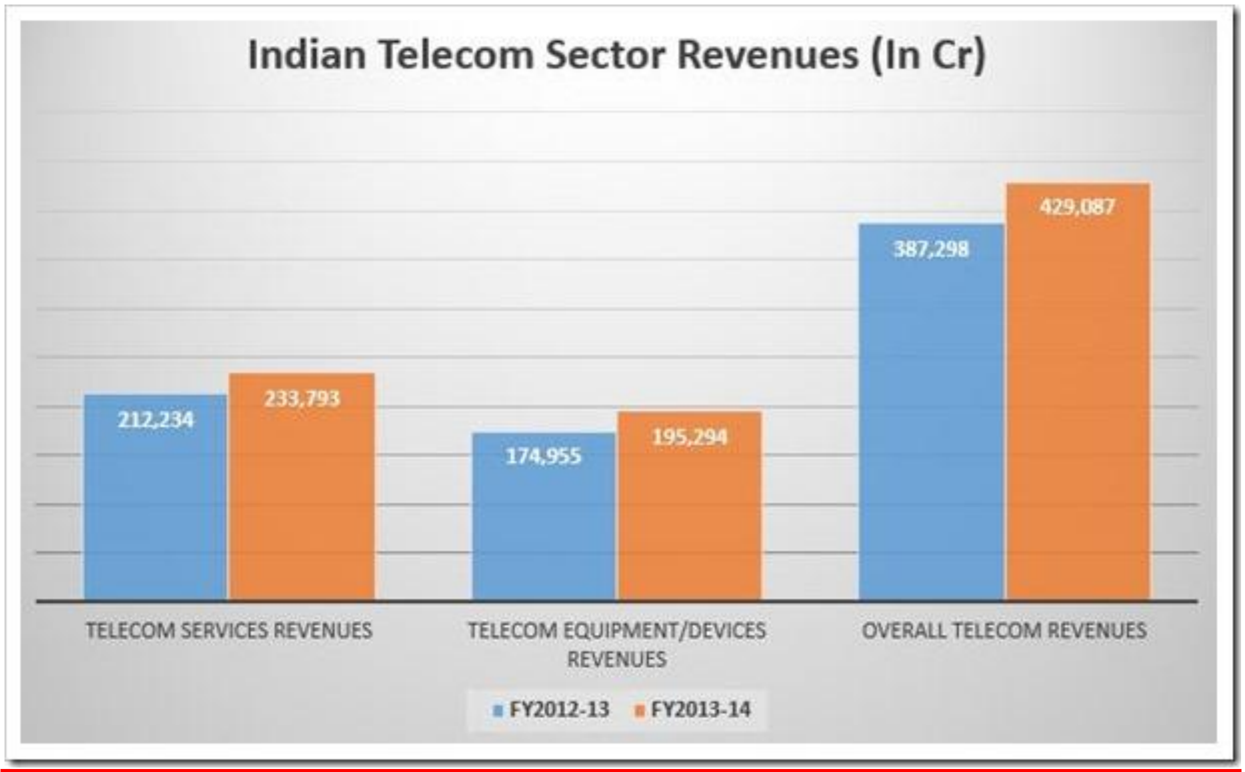
Exhibit 1.1: Evolution of Telecom Industry



Phase - I (1980-89)
• Entry of Private Sector in telecommunications equipment manufacturing - 1984
• Formation of Mahanagar Telephone Nigam Limited (MTNL) and Videsh Sanchar Nigam Limited (VSNL) - 1986
• Telecom commission was set up - 1989
Phase - II (1990-99)
• Libralisation of Indian Economy-1990's
• Private sector participation in provision of VAS such as cellular and paging services - 1992
• National Telecom Policy announced-1994
• Telecom Regulatory Authority of India (TRAI) was established in 1997
• New Telecom Policy (NTP) announced - 1999
Phase - III (2000 onwards)
• Bharat Sanchar Nigam Limited (BSNL) established - 2000
• National Long Distance (NLD) and International Long Distance (ILD) services opened to competition-2000
• CDMA technology launched - 2000
• Internet telephony initiated - 2000
• Reduction of license fees - 2000
• VSNL privatised - 2002
• Launch of mobile service by BSNL-2002
• Unified Access Licensing (UASL) regime was introduced - 2003
• Calling party pays (CPP) was implemented - 2003
• Broadband policy was formulated-2004
• Intra circle merger guidelines established - 2004
• FDI limits increased from 49% to 74% - 2005
• Number portability was proposed (pending) - 2006

Source: D&B Research

Revenue



CONTRIBUTION TO GDP

There is a direct correlation between the growth in mobile teledensity and the growth in GDP per capita in developing countries, which tend to have a high percentage of rural population. The share of the telecom services industry in the total GDP is around 3%.

EMPLOYMENT

Data Series	Feb 15	Mar 15	Apr 2015	May 2015
All employees	861.6	864.1	868.2	868.1
Production, nonsupervisionary	727.7	728.3	729.9	N.A.
Unemployment	1.9%	1.7%	2.8%	3.3%

1.1.3. Organization Analysis – Elitecore Technologies Pvt. Ltd.

Elitecore Technologies is a leading provider of OSS/BSS, Packet Core and Access Gateway Solutions. Elitecore was established in 1999 by highly experienced software technocrats with over 20 years of experience in IT consulting, who initially started the company's operations with billing software development. Today, it has come a long way to be ranked amongst a handful of product-centric IT companies from India, that thrive on innovation to deliver specialized solutions which are sold under well-known brand names for its global customer base.

Ever since its inception, Elitecore has been able to sustain a healthy growth rate. Elitecore has aptly envisioned as "Platform for Innovators" to describe the organization's core values. Elitecore is a dynamic player with domain expertise in offering products encompassing Signaling, Service Control, Policy Control, Real Time Charging & Billing.

Elitecore's has over 150+ network deployments in 52 Service Providers, with Presence across 30+ countries.

It is recognized as a Niche Player in Gartner MQ IRCM Report in 2013-14 and listed as one of the fastest growing IRCM Company for the year 2014.

SERVICES OFFERED

Elitecore Technologies is a Carlyle Group investee global IT product company providing Integrated Policy, Charging and Revenue Management solutions with flexibility of modular as well as pre-integrated offerings. Elitecore solutions are compatible to large vendor ecosystem leading to CSP's requirement of faster time to market and better TCO. Being a traditional IP solutions player, Elitecore products are highly responsive to next-generation services, fulfilling operator monetization needs across all access networks.

Telecom Business Segments

- Wireless Broadband / Packet Core – 3G, 4G/LTE, WIMAX
- Service Provider WIFI , Wi-Fi –Offload (3G, 4G)
- Convergent OSS BSS
- Fixed Line / Wire-line Broadband
- Pay TV
- Cloud Billing
- Enterprise Billing
- MVNO
- M2M
- NGN Transformation
- ISP Billing & Bandwidth Management

Services Portfolio

- Product Implementation
    - (i) Managed Services
    - (ii) Turnkey Project Implementation
    - (iii) Operations & Maintenance
- System Integration via our partners
- Education & Training - Complete Product Training to End user(Staff)& SI
- Support - 3 Levels of Support
- Effective Change Request Process

Value Proposition

- Offering Billing & Revenue Management, Charging, Policy Management & AAA solution
    1. Reduced TCO
    2. Faster Integration, Implementation & Launch of new service
- End to End Support from Single Team to address Consulting, Development & Implementation
- Ability to offer Modular & End to End solution [Pre-Integrated Product Stack] at competitive pricing

Product Portfolio

- Crestel Convergent Billing
- Crestel Prepaid Billing,
- Crestel RnC (OCS)
- Crestel CGF (Charging GW)
- Crestel Mediation
- Crestel Partner Management
- Crestel Interconnect Billing
- Crestel Voucher Management
- Crestel IP Log Management System
- Crestel Work Order Management

1.1.4. Major Departments

NAME OF THE DEPARTMENT	ROLE OF THE DEPARTMENT IN THE ORGANIZATION

Technical Department	The Technical Department is responsible for all the technical aspects related with the Elitecore Technologies Pvt. Ltd. This includes the maintaining the software and services provided and keeping it up-to-date. It also handles the updating of the licensing of the software and fixing any bugs that are incorporated with the product. It is a nucleus of engineers to ensure that the services provided are optimally, efficiently and effectively maintained. It acts as the backbone of the company.
Human Resource Department	The Human Resource Department provides Elitecore with structural stability and ability to meet business needs through managing the company’s most valuable resource – it’s employees. It manages the recruitment process, takes care of the needs of the employees, helps in maintaining the work environment peaceful, and takes care of the training and development process among lot of other important tasks. It focuses on improving or changing the knowledge skills and attitudes of the employees, inducing employee involvement and motivation, and guiding the company's workforce towards a constructive goal
Research and Development Department	Elitecore’s R&D Department plays an integral role in the life cycle of the services it provides. While this department usually is separate from sales, production and other divisions, the function of these areas are related and often require collaboration. The R&D is involved in new product research, new product development, existing product updates, quality checks and innovation.

1.2. *Process Mapping*

Elitecore is the provider of NG OSS/BSS & allied professional services to Telecom Operators/ ISP/ Residential / Enterprises /Small and Medium Service Providers across the globe. Elitecore meets the requirements for AAA, Real time Charging, PCRF, and Convergent Billing & Service Delivery Solutions across Wireline and Wireless Networks.

Headed by highly experienced Telco S/w Implementation head to address the complexities in the OSS/ BSS infrastructure and manageability Elitecore professional Services group, Elitecore addresses Project management and professional services to offer best practices and service to ensure efficient processes in each of our projects.

With A Single team to address Consulting, Development, Implementation & Support, Elitecore offers an advantage to Telecom Operator, Network Service Providers and System Integrators enabling faster service roll out.

Elitecore professional Services group comprises of Project management, Operations & Maintenance, Managed Services, Time and Material, Post Migration/Go live support & Turnkey project Implementation.

EXPERIENCE

- Consulting, Implementation and Managed Services for Elitecore Customers & Partners
    1. Professional Services Team having worked in operator environments
    2. PMP Certified team members, Java developer & DBA
    3. Dedicated Account manager
- Efficient Change Management Process
- Experience in Project Implementation for TEIR 1 to TIER 3 operators

DELVERY SERVICES

- **Project and Program Management**
  Right from winning the project to go live is coordinated by the Project Manager who takes end-to-end responsibility, from planning, migration and integration through to

issue resolution and preparation of progress reports.

- **Solution requirement Study**
  Business Requirements Analysis – Our team works on understanding and documenting the business objectives and requirements of the project surrounding the product /solution.
  Technical Requirements – Based on the business objective a spread sheet is made with the technical requirements needed after analyzing the current system.
  Solution Architecture – Proposed solution architecture is prepared keeping in mind the requirements for the overall solution with key interfaces and functionality

- **High level Solution Design**
  As a next step a high level solution design is made, mapping the requirements and proposed architecture to existing system and identifying customization work.

- **Software Development**
  Based on the HLD there will be Creation of source code needed for integrating separate systems or to provide functionality missing and formal documentation of interface specification for information exchange.

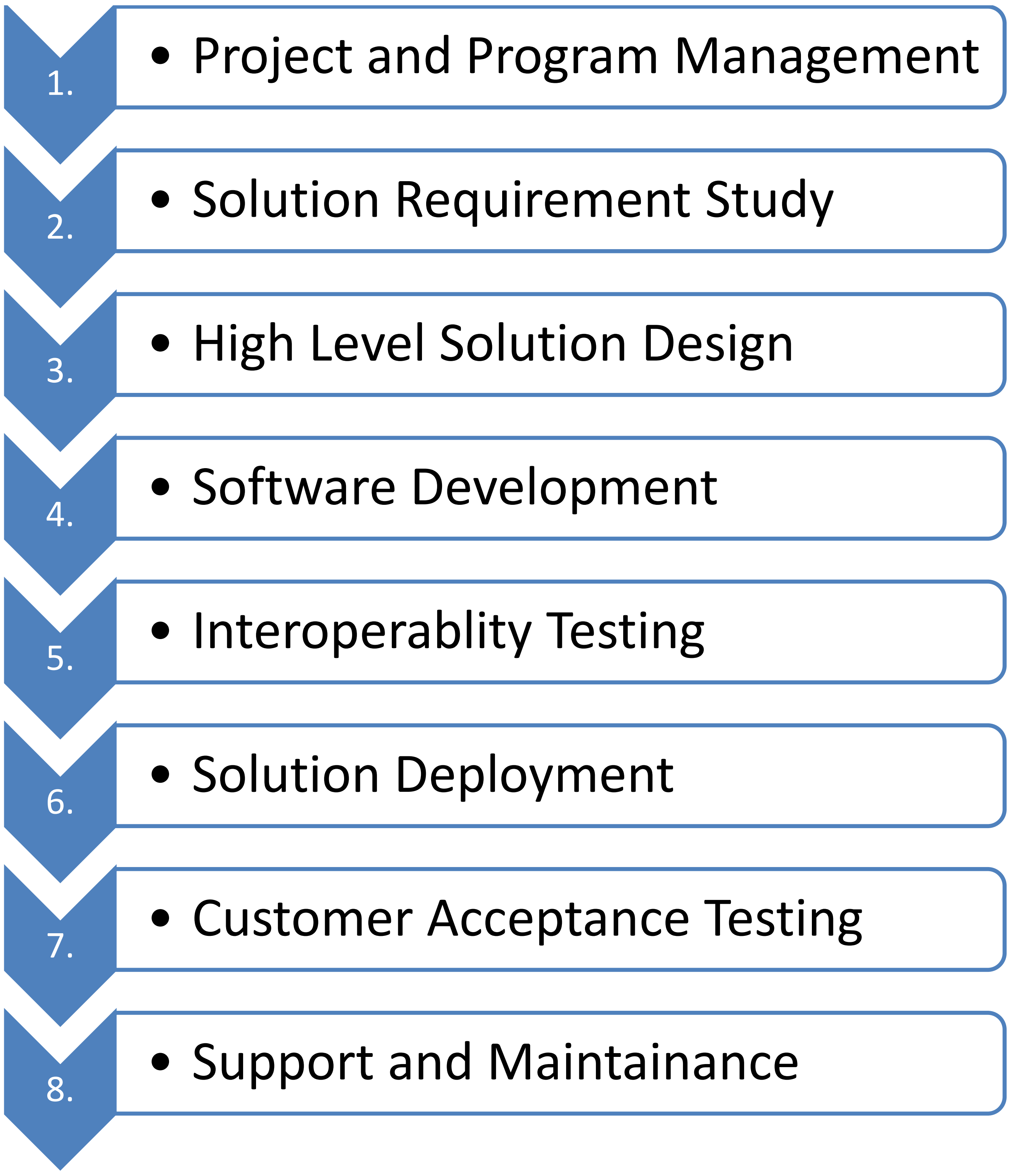
- **Interoperability Testing**
  IOT is done with various network elements present in the ecosystem to show conformance to system requirements.

- **Solution Deployment**
  It involves the set of activities related to site preparation, deployment and installation of the solution.

- **Customer Acceptance Testing**
  Formal testing of the delivered solution against the business requirements specified for the project.

- **Support and Maintenance**
  Post implementation support functions like troubleshooting, hot fixes and ongoing updates for the solution.

FLOW DIAGRAM



1.
• Project and Program Management
2.
• Solution Requirement Study
3.
• High Level Solution Design
4.
• Software Development
5.
• Interoperablity Testing
6.
• Solution Deployment
7.
• Customer Acceptance Testing
8.
• Support and Maintainance

1.3. Projects undertaken at Practice School

In 16 days of Practice School, we have had the opportunity to participate in multiple projects and assignments. It has been an amazing learning experience, where we were able to channel our knowledge productively into making a useful contribution to the company.

Our first task was to study about java xml parsers in android and to practice making them.

We were also asked to study about ANDSF and OMA DM.

We studied about different classes of the package android.net.wifi and implemented them in android, invoking web services in android, android background services and hotspot 2.0 features.

1.3.1. Mid-Term

1.3.1.1. XML parsers in android

What is XML Parsing?

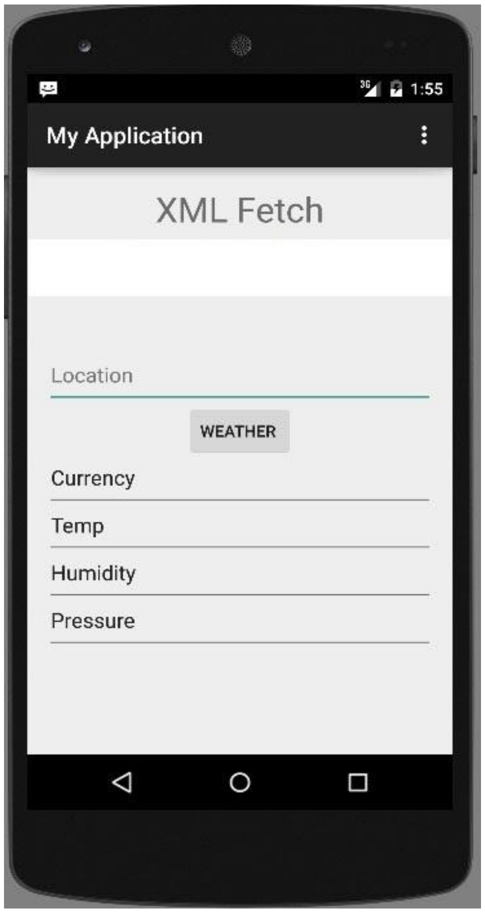
Parsing XML refers to going through XML document to access data or to modify data in one or other way.

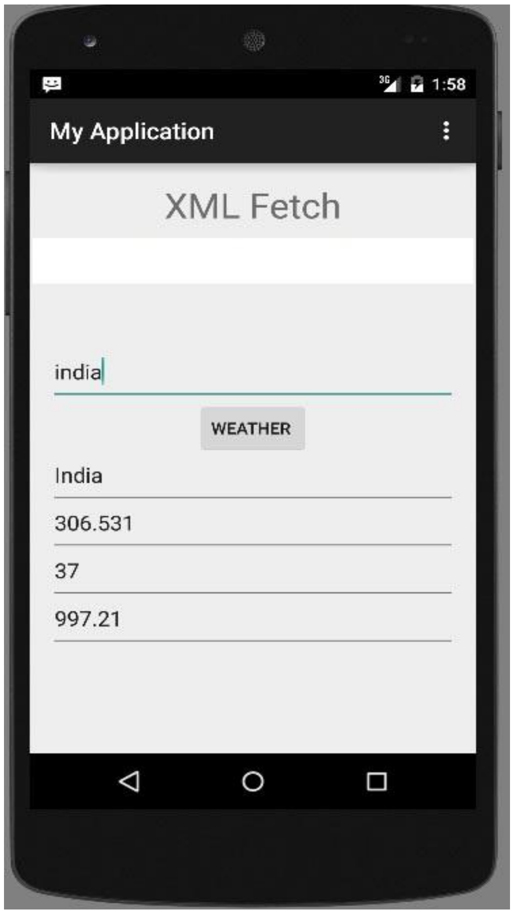
What is XML Parser?

XML Parser provides way how to access or modify data present in an XML document. Java provides multiple options to parse XML document. Following are various types of parsers which are commonly used to parse XML documents.

- **Dom Parser** - Parses the document by loading the complete contents of the document and creating its complete hierarchical tree in memory.
- **SAX Parser** - Parses the document on event based triggers. Does not load the complete document into the memory.
- **JDOM Parser** - Parses the document in similar fashion to DOM parser but in easier way.
- **StAX Parser** - Parses the document in similar fashion to SAX parser but in more efficient way.
- **XPath Parser** - Parses the XML based on expression and is used extensively in conjunction with XSLT.
- **DOM4J Parser** - A java library to parse XML, XPath and XSLT using Java Collections Framework, provides support for DOM, SAX and JAXP.

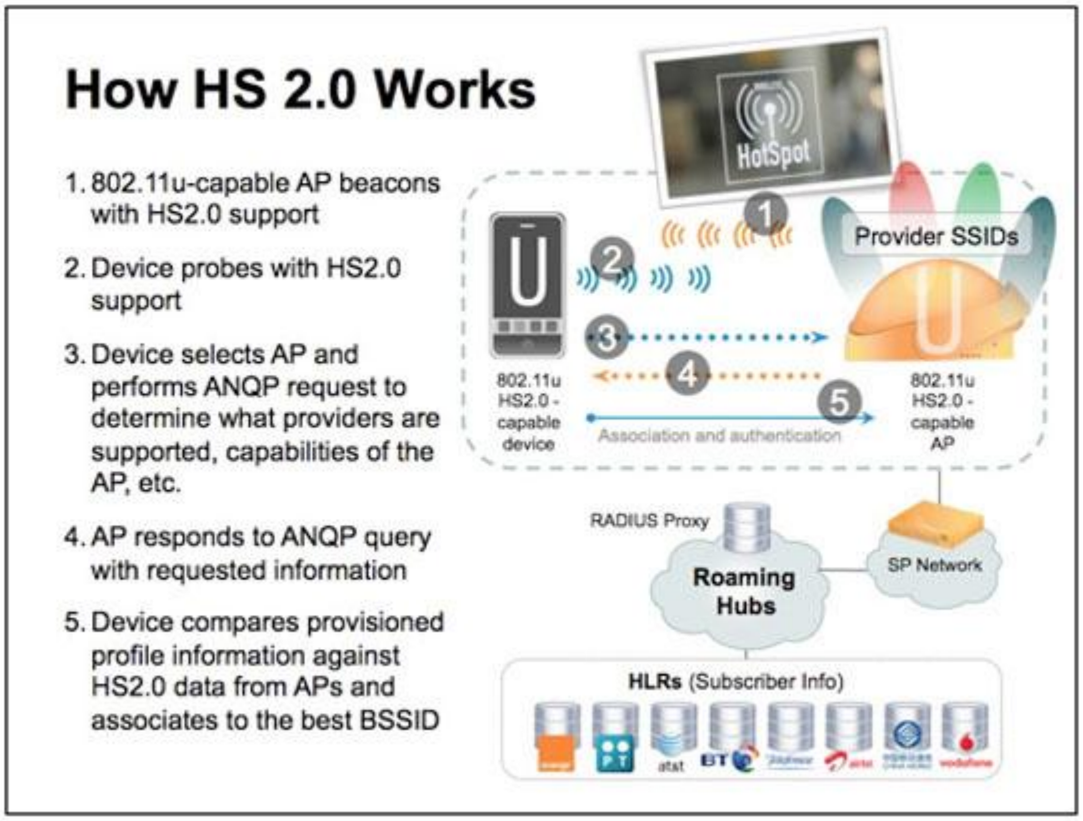
To practice we created a basic weather application that allows us to parse XML from google weather api and show the results (we have mentioned the code in appendix). Given below are the screen shots of the app that we made.





1.3.1.2. Hotspot 2.0

Release 1 introduced new capabilities for automatic Wi-Fi network discovery, selection, and 802.1X authentication all based on the Access Network Query Protocol (ANQP). With Hotspot 2.0, the client device and access point exchange information prior to association using ANQP. The access point advertises the “backend” service providers (SPs) who can process authentication requests that are reachable from this hotspot. The client than checks to see if it possesses a credential for one of those SPs, and if it does, it proceeds to associate and then authenticate to the AP using 802.1X and the provisioned credential. Supported client credentials include SIM cards, USIMs, X.509 certificates and username/password pairs. Each credential is associated with a specific EAP type. The primary benefits of Release 1 were automating the connection experience at hotspots where the client credential was accepted and providing a secure, encrypted air-link for public Wi-Fi. A secondary benefit is the ability to support multiple roaming partners over a single SSID, with SSID proliferation becoming a big issue for operators looking to expand their footprint through roaming relationships.



How HS 2.0 Works

1. 802.11u-capable AP beacons with HS2.0 support
2. Device probes with HS2.0 support
3. Device selects AP and performs ANQP request to determine what providers are supported, capabilities of the AP, etc.
4. AP responds to ANQP query with requested information
5. Device compares provisioned profile information against HS2.0 data from APs and associates to the best BSSID

Until Release 2 there was no standard format for managing a Hotspot 2.0 credential on a client device. Depending upon the OS or manufacturer, a text or XML file was typically used, but these might have different naming conventions, syntaxes, and locations within the file system. Release 2 leverages the Open Mobile Alliance's Device Management (OMA-DM) framework, which provides a standardized XML tree structure within which different kinds of information can be stored in a consistent manner. Release 2 specifies a new PerProviderSubscription Management Object (PPS-MO), which is one or more of the branches in the OMA-DM tree containing all of the information related to the Hotspot 2.0 credentials on the device.

A Release 2 client will see the Release 2 support in the Hotspot 2.0 indication element of the APs beacons and probe responses. The client then sends an ANQP query to the Release 2 AP. In the ANQP response, the AP indicates that Online Signup services are available and lists the OSU providers that are reachable from this hotspot. Since the client does not have a valid credential associated with this hotspot operator, or any of its roaming partners, it does not proceed to automatically associate and 802.1X authenticate. Instead, while it is still in the pre-association phase the user will be notified that Online Signup services are available. If the user elects to sign up, they will be presented with a list of the available Online Signup providers. The list is typically displayed as an icon, title, and description for each operator. The icon is actually embedded within certificate issued to the OSU server, thus ensuring that clients don't connect to "rogue" provisioning systems. Remember that everything described so far has happened while the client is not yet associated to any WLAN.

At this point, we should detour to discuss a new type of WLAN that is being introduced with Release 2. The OSU Server-only authenticated layer 2 Encryption Network (OSEN) is similar to the trusty old RSN, with the striking exception that OSENs only require authentication of the servers and expect that the client will remain anonymous during the session. OSEN is used exclusively as an option for the OSU WLAN, and is prohibited from being used in production WLANs. The other option for the OSU WLAN is to use Open Authentication. If OSEN is used, it is encrypted using Anonymous EAP-TLS, which again only authenticates the server/network and not the client using the PKR trusts. The intent is to ensure that the client is connecting to a valid/trusted OSU WLAN and that the registration and provisioning servers are authenticated. In order to accomplish this, there will be new Public Key Infrastructure (PKI) root trusts loaded into Release 2 clients. These will be used to validate OSU servers and the OSU WLAN if the OSEN option is used.

Once the user selects an OSU provider from the list, the connection manager on the device will connect to the OSU WLAN (Open or OSEN). It then triggers an HTTPS connection to the OSU server URI, which was received with the OSU providers list. The client validates the server certificate to ensure it is a trusted OSU server.

Table 1: Release 1 standards and new introductions with release 2

TABLE 1

Release 1	Release 2
Credential (or Credential Pointer)	Preferred Roaming Partners
Home Provider Domain	SSID Blacklist for Autonomous Selection
Roaming Consortium OIs	Home Provider AAA Server Trust Root
NAI Realm and/or PLMN ID	Subscription Update Parameters
Required TCP and UDP Services/Ports	Usage Limits (Time or Data)

At this point the client will be prompted to complete some type of online registration through their browser – this could be anything from registering as an existing subscriber to purchasing prepaid access for a few hours. When they successfully complete the registration, they will be pushed a credential, associated parameters and, optionally, the policy to apply for that credential. Release 2 specifies that this secure communication between the provisioning servers and the clients can be accomplished with either SOAP-XML or OMA-DM messages over HTTPS. These messages map to the PPS-MO structure, and the information is stored in the management object on the client. Finally, now that the client has a valid credential for the production HS2.0 WLAN, it disassociates from the OSU WLAN and connects to the HS2.0 WLAN using the standard ANQP mechanisms. The connection manager also factors any configured policies into its selection decisions when utilizing the credential. From then on, the credential provider can use this framework to update the credential, policy or subscription of the device by indicating via RADIUS messaging that the client needs to contact one of the provisioning servers. Additionally, the client also can track parameters stored it its PPS-MO, such as update intervals, subscription expiration, or data usage limits, and automatically contact the appropriate subscription or policy server when an update or renewal is needed.

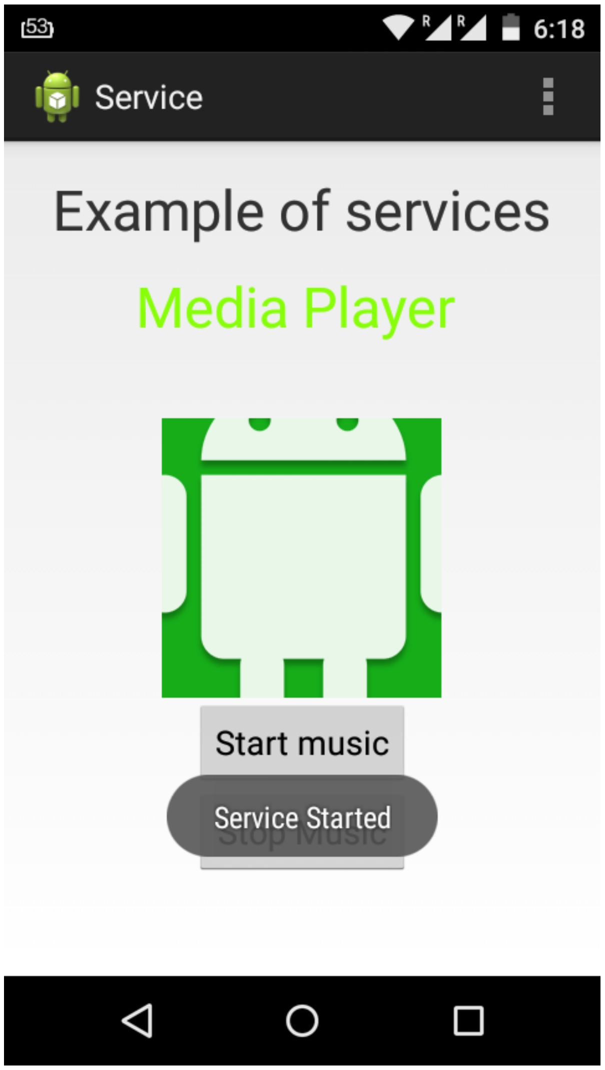
1.3.1.2. Android Services

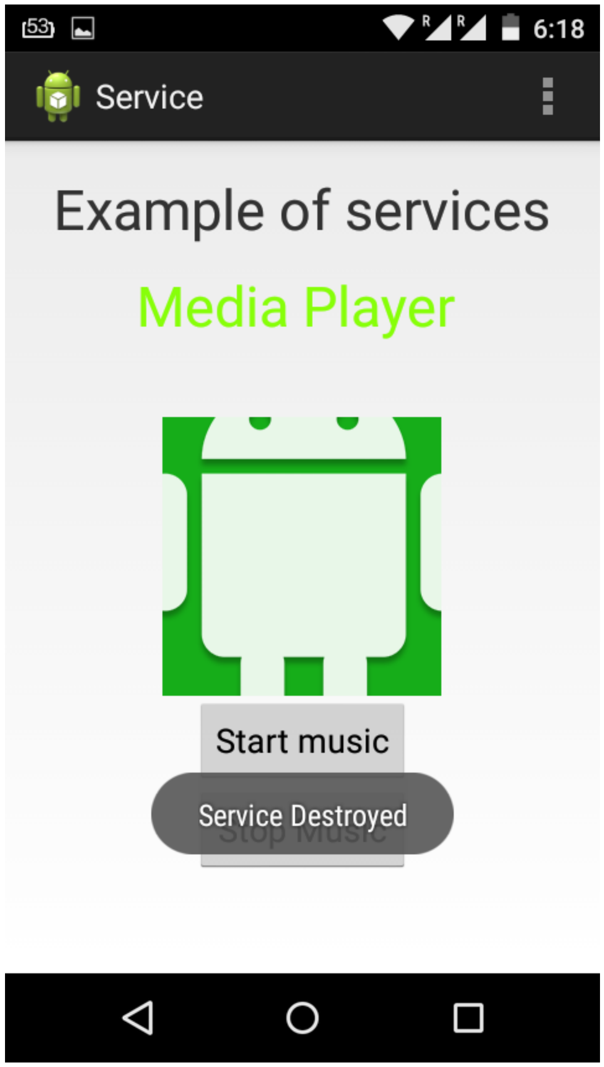
A **service** is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed. A service can essentially take two states –

State	Description
Started	A service is **started** when an application component, such as an activity, starts it by calling *startService()*. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.
Bound	A service is **bound** when an application component binds to it by calling *bindService()*. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC).

For practice purpose, we made an app that plays music in the background using android services even when the activity is closed, also it displays toast saying “Service started” and plays music on clicking the button “Start music” and displays the toast “Service stopped” and stops the music on clicking the button “Stop music”.

The code for the app has been given in the appendix.





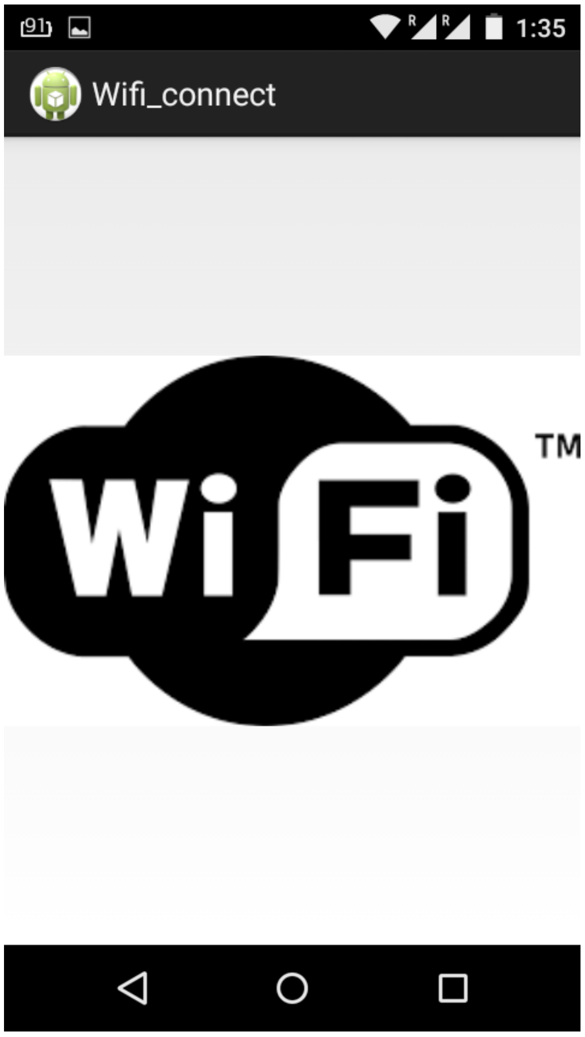
1.3.1.3. Wi-Fi Android app

Android allows applications to access the state of the wireless connections at very low level. Application can access almost all the information of a wifi connection.

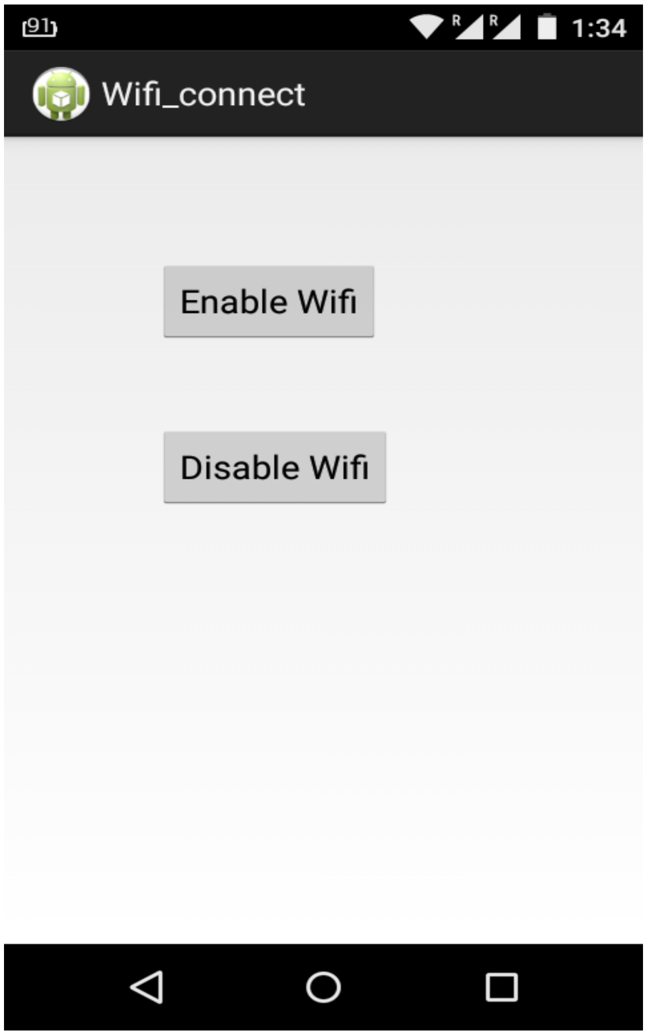
The information that an application can access includes connected network's link speed, IP address, negotiation state, other networks information. Applications can also scan, add, save, terminate and initiate Wi-Fi connections.

We made an app that first displays the splash screen, then asks for the option to enable or disable Wi-Fi. On clicking on the button "Enable Wi-Fi" it scans for the available Wi-Fi hotspots nearby the mobile network and show a list of them with their full details. On clicking the button "Disable Wi-Fi" it disables the Wi-Fi in the handset.

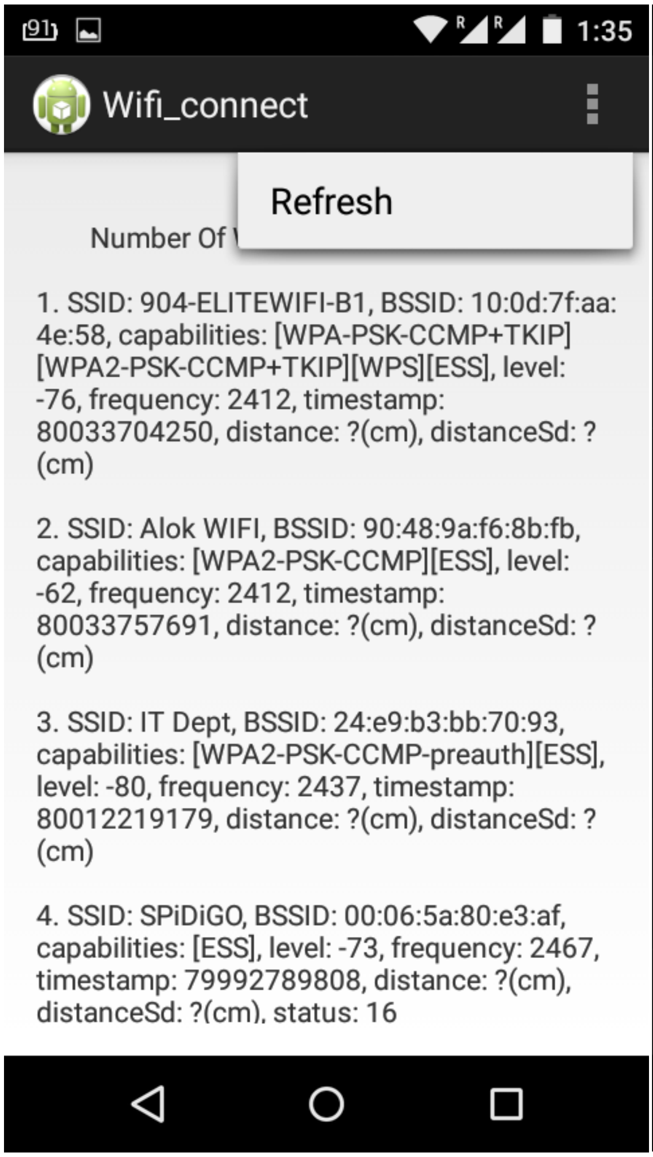
Here are the screen shots of our app which we modified further to provide the user an option to connect to the desired network. We have shared the code in appendix.



1:35
Wifi_connect
WiFi
TM



1:34
Wifi_connect
Enable Wifi
Disable Wifi


Wifi_connect

Refresh

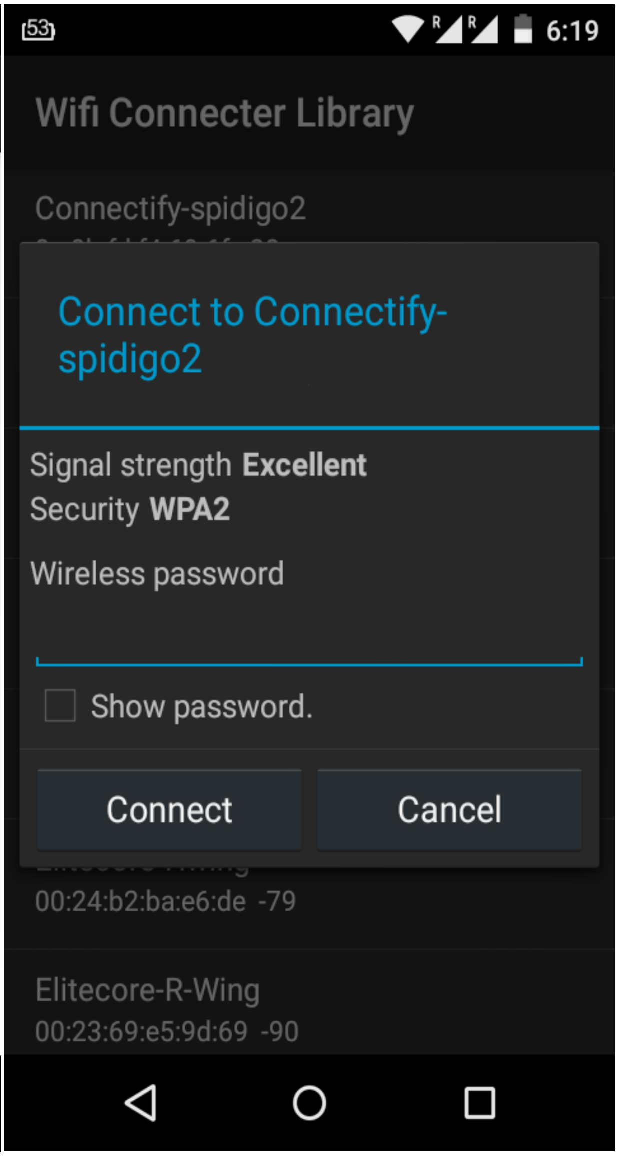
Number Of

1. SSID: 904-ELITEWIFI-B1, BSSID: 10:0d:7f:aa:4e:58, capabilities: [WPA-PSK-CCMP+TKIP][WPA2-PSK-CCMP+TKIP][WPS][ESS], level: -76, frequency: 2412, timestamp: 80033704250, distance: ?(cm), distanceSd: ?(cm)

2. SSID: Alok WIFI, BSSID: 90:48:9a:f6:8b:fb, capabilities: [WPA2-PSK-CCMP][ESS], level: -62, frequency: 2412, timestamp: 80033757691, distance: ?(cm), distanceSd: ?(cm)

3. SSID: IT Dept, BSSID: 24:e9:b3:bb:70:93, capabilities: [WPA2-PSK-CCMP-preauth][ESS], level: -80, frequency: 2437, timestamp: 80012219179, distance: ?(cm), distanceSd: ?(cm)

4. SSID: SPiDiGO, BSSID: 00:06:5a:80:e3:af, capabilities: [ESS], level: -73, frequency: 2467, timestamp: 79992789808, distance: ?(cm), distanceSd: ?(cm), status: 16


Wifi Connecter Library

Connectify-spidigo2

Connect to Connectify-spidigo2

Signal strength **Excellent**
Security **WPA2**
Wireless password

Show password.

Connect Cancel

00:24:b2:ba:e6:de -79

Elitecore-R-Wing
00:23:69:e5:9d:69 -90

1.3.2. Post Mid- term

1.3.2.2. Deriving logic to validate ANDSF policy for USER equipment

Abstract :

This report presents a strategic solution to mobile data offloading between 3GPP and non-3GPP access networks. As mobile data traffic is increasing rapidly, many 3G operators face massive challenges in managing the huge amount of mobile data traffic. Thus, mobile data offloading to non-3GPP access networks in 3G-access networks is necessary. In this report, we propose a novel ANDSF-assisted Wi-Fi control method based on user's high-level motion states, such as walking, driving, and recognizing whether the user is stationary or not, to avoid unnecessary Wi-Fi scanning and connections. The proposed method is compared to a client-based Wi-Fi connection manager, which performs periodic Wi-Fi scanning. According to the

performance results, the proposed mechanism shows significant performance improvement in terms of efficient Wi-Fi control and connectivity with 3G and WiFi.

Introduction:

According to the latest Cisco Visual Networking Index report, monthly global mobile data traffic will exceed 10 exabytes by 2016; Asia, in particular Japan, South Korea, Indonesia, and China, will account for 40% of that amount. Furthermore, the continuous advent of mobile services and the high speeds of mobile connections are the major factors in the growth of mobile
data traffic.

Today, mobile users frequently use many smartphone/tablet applications, such as social networking applications, location based service applications, and mobile messengers, to communicate with application servers and peers. It will be unlikely to change mobile user trends
for the use of such applications, and the traffic volume will be increase even more. Thus, growing mobile data traffic brings the requirements of mobile infrastructure improvement and a
new wireless technology, such as Long Term Evolution (LTE) to fulfill mobile user demand. Mobile network operators (MNOs) might have several approaches to deal with the mobile data traffic explosion. The approaches are additional spectrum acquisition, existing 3G network upgrades such as a multichannel assignment, LTE deployment, and alternative networks, such as
Wi-Fi. Since all the approaches, except the fourth one, are dependent on MNO policy, thus we limit the discussion to an alternative network as a solution for mobile data offloading between 3GPP and non-3GPP access networks in this report.

The solutions for mobile data offloading can be categorized into two solutions, such as a client-based solution and a server-based solution. Various Wi-Fi connection managers that are the applications for connecting the Wi-Fi more easily and quickly are available on user equipment (UE), and it can be a client-based solution for mobile data offloading. However, with the client-based solution, it is impossible for the MNO to gain visibility and control over Wi-Fi traffic and the user experience for better mobile traffic management. In addition, most Wi-Fi connection managers always require turning on a Wi-Fi interface on the UE, and thus it can cause unnecessary Wi-Fi scanning. Therefore, a number of unnecessary Wi-Fi connections might be established.

Recently, ANDSF has been specified in 3GPP standards 23.402 and 24.312as a server-based solution. The ANDSF is an entity within an Evolved Packet Core (EPC) to assist user equipment (UE) in discovering non-3GPP access networks and provide UE with rules and operator policies to connect to the non-3GPP access networks. However, the ANDSF server needs to acquire the current geographical location of the UE to provide discovery information, which is a list of available Wi-Fi networks near by the UE's current location. Thus, it can also cause an unnecessary energy drain on the UE owing to its power intensive location sensing and the synchronizing operations between the ANDSF server and the UE.

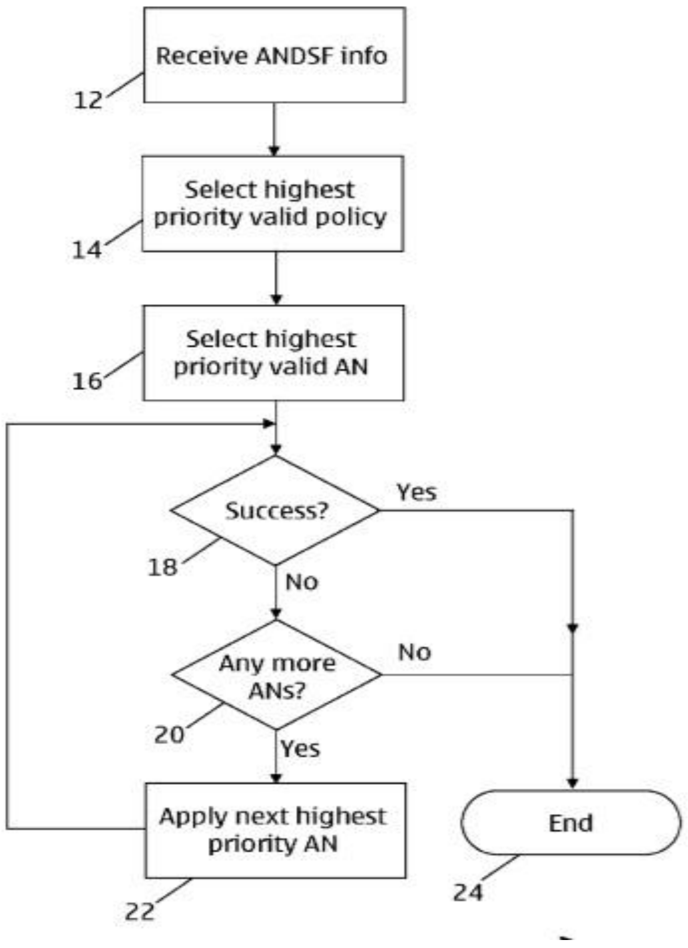
ANDSF:

Access network discovery and selection function (ANDSF) is an entity within an evolved packet core (EPC) of the system architecture evolution (SAE) for 3GPP compliant mobile networks. The purpose of the ANDSF is to assist user equipment (UE) to discover non-3GPP access networks – such as Wi-Fi or WIMAX – that can be used for data communications in addition to 3GPP access networks (such as HSPA or LTE) and to provide the UE with rules policing the connection to these networks. It defines three groups of information that can be sent to a handset:

1. Inter-system mobility policy (ISMP) – This specifies which network type to connect to when only one network connection will be used, for example a choice between LTE and Wi-Fi.

2. Inter-system routing policy (ISRP) – If multiple networks can be used simultaneously, this specifies which type of traffic should use which network.

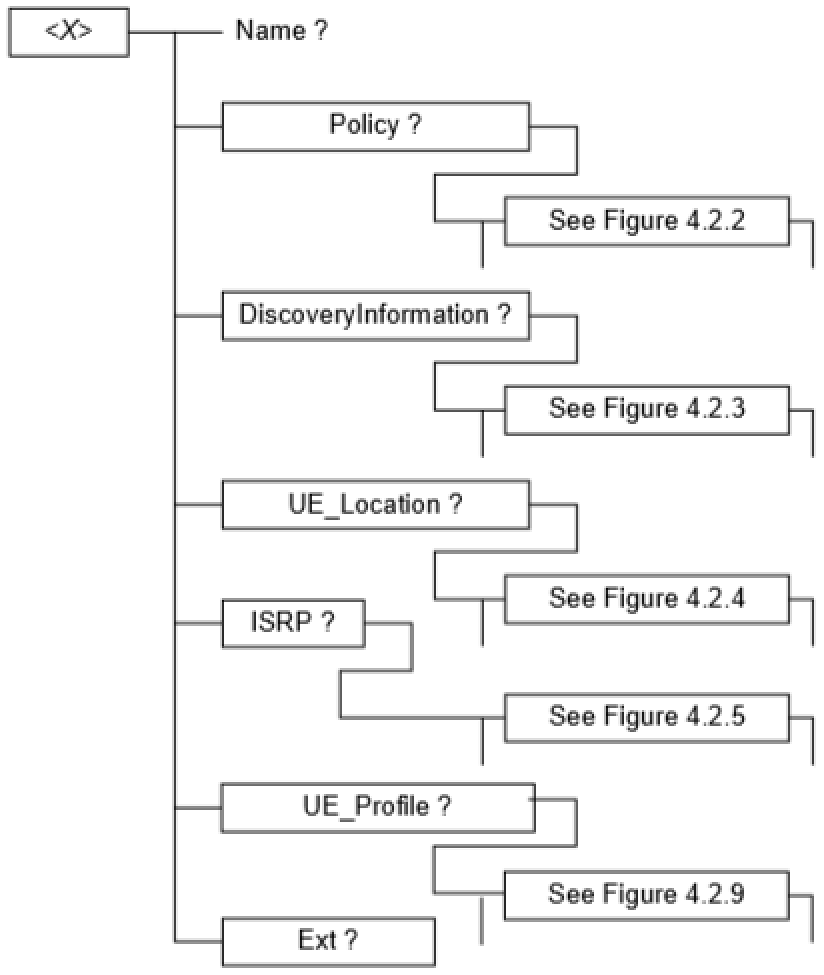
3. Discovery information (a list of non-3GPP) – This is information about networks that might be available in the handset's vicinity.


Receive ANDSF info
12
Select highest priority valid policy
14
Select highest priority valid AN
16
Success?
Yes
18
No
Any more ANs?
No
20
Yes
Apply next highest priority AN
22
End
24

This is a flow chart showing a network access selection algorithm. The algorithm starts at step 12, where ANDSF information is obtained. The ANDSF information may, for example, be provided (by the ANDSF server) on request from the mobile communication device (in a "pull" mode) or may be provided in a manner determined and initiated by the ANDSF server (in a "push" mode). The ANDSF server provides a number of policies for connecting the mobile communication device to networks. Priorities are assigned to the various policies and, at step 14 of the algorithm, the highest priority valid policy is applied by the mobile communication device. A policy is considered to be “valid” if it meets a number of validity conditions. Such conditions may, for example, relate to location or the time of day.

The policy selected at step 14 will have a number of access network options associated with it. The access net Work options will be prioritized within the policy. At step 16 of the algorithm, the highest priority access network option of the selected policy is selected at the mobile communication device. The algorithm moves to step 18 Where it is determined Whether or not the highest priority access network option of the selected policy has resulted in a successful connection to an available network. If a network connection has been made, then the algorithm terminates at step 24. If a connection has not been made then the algorithm moves to step 20, Where it is determined Whether or not the selected policy has any more (lower priority) access network options available. If no further access network options exist, the algorithm terminates at step 24. If further access network options do exist, then the next highest priority access network is selected at step 22. The algorithm then returns to step 18, Where it is determined Whether or not the newly selected access network results in a successful connection to an available network. If a connection is made, then the algorithm terminates at step 24. If a connection is not made, then the algorithm 10 moves to step 20, as discussed above. The algorithm 10 continues until either a successful connection is made or all access network options of the selected policy have been tried.

In some implementations of ANDSF systems, if the highest priority valid policy does not result in a connection being made, then no further efforts are made to make a connection. In other implementations, if the highest priority valid policy does not result in a connection being made, then the next highest priority policy (if any) is used and steps 16 to 24 of the algorithm 10 are repeated using that policy. In addition to providing network selection policies, ANDSF allows mobile operators to provide access network discovery information (ANDI) to assist user equipment (UE) in detecting access networks specified in the ANDSF policy rules. Policies are used to list preferred access networks in any given location or time.

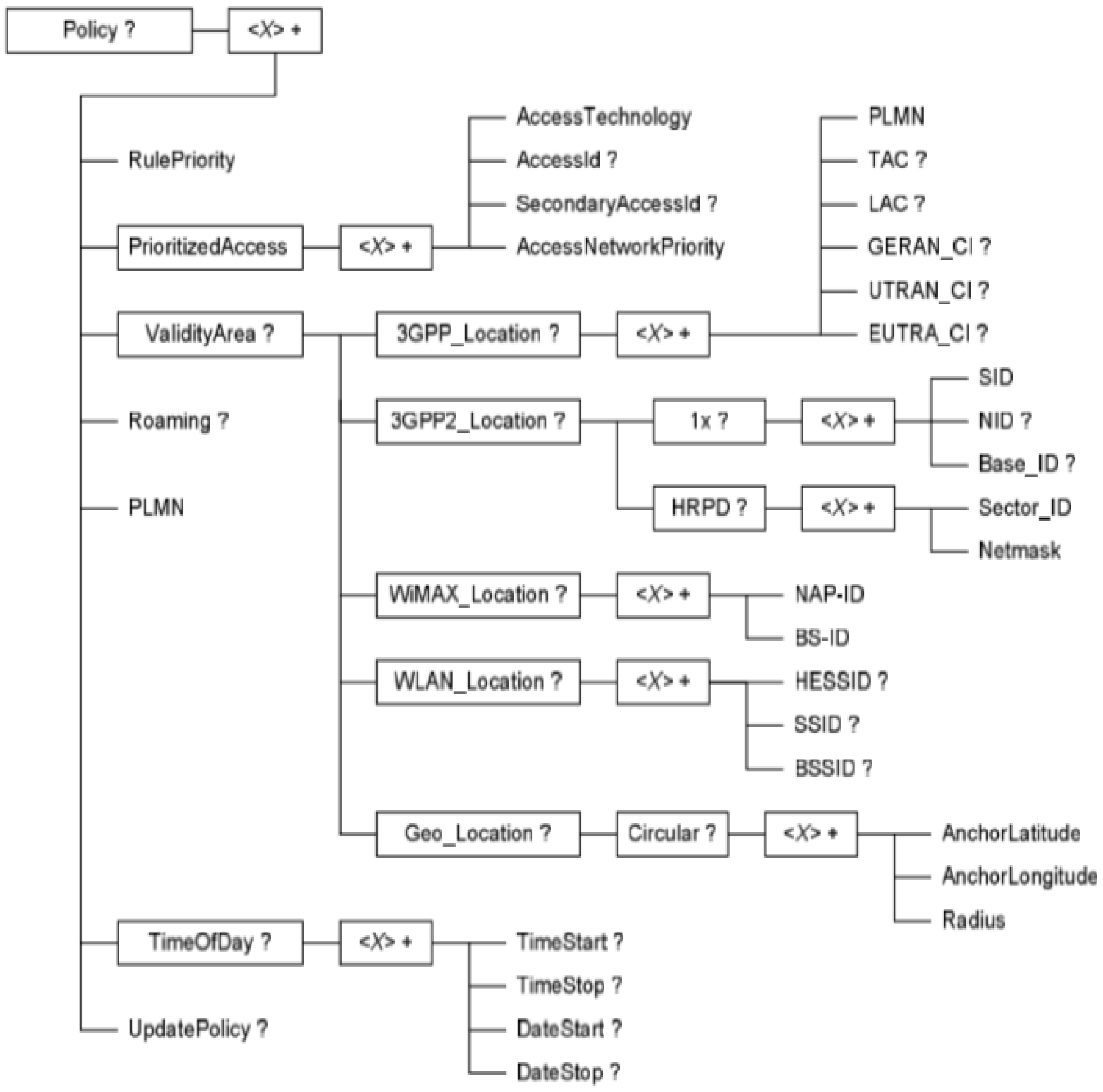


<u>VALIDATING ANDSF POLICIES WITH USER EQUIPMENT -</u>

Our main aim was to parse the ANDSF policies from the web server and validate it with the data parsed from the User Equipment. If the correct policy is found, the User Equipment is connected with the specified network.

SORTING

Once the policy is parsed using the “ksoap” class, they are sorted accoding to their Rule Priority which is specied inside the Valid Area class of the policy. The higher value of rule priority means lower is it's priority with respect to other policies. Hence we need to first validate a policy which has the lowest Rule Priority. The algorithm used in the code specified in the Appendix uses the Qucik Sort method to sort the priorities.

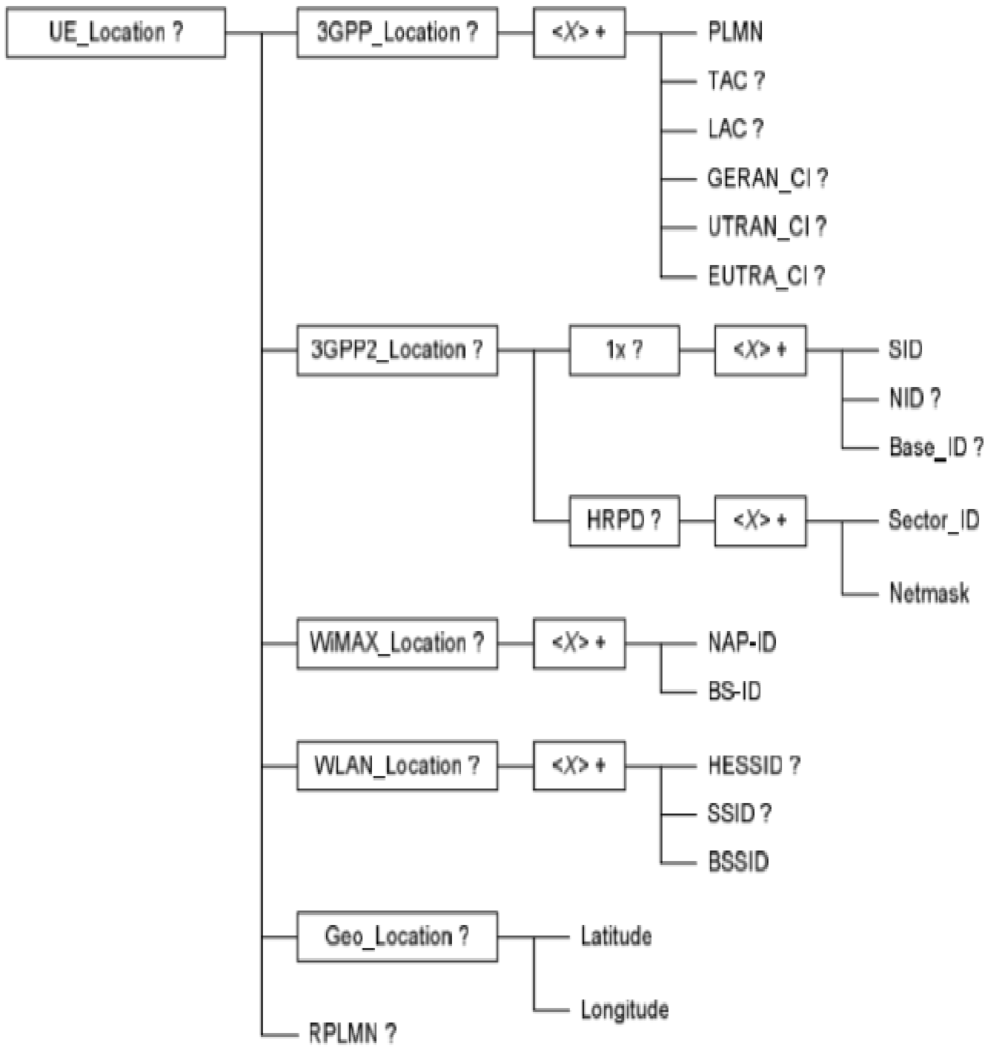


Following this, the location of the policies is parsed for the policy with minimum Rule Priority. Following options are available for the location and any one of the following can be used to validate the location -

1. Geographical Location (Latitude and Longitude) –
 First the Geographical Location of the policy is parsed. This is then used to calculate the distance between the location of User Equipment and the Policy location. If the User Equipment is found within the specified range or Radius, then we are successful and we don't need to find any other location parameters.

2. WLAN Location
3. 3GPP Location
4. 3GPP2 Location

Similar to Geo Location, any other of the above location parameter can be used to check if the User Equipment is within the range of the policy. If none of the above location returns a successful result, then no policy was found and we can turn off the wifi to save the energy



consumption.

However if one of the location parameters returns a successful result then we continue to validate the Time Of the Day.

TIME OF THE DAY

Once the location of the policy is evaluated, it is then taken to the next step where the Time of the policy is validated. Time of the day for the policy take four parameters -

1. Time Start
2. Time Stop
3. Date Start
4. Date Stop

Similary, two parameters are taken for the User Equipment -

1. Current Date
2. Current Time

To validate the Time, it is required to check that the current date should lie between the start date and end date. Also the current time shoild lie between start time and end time.
If the User Equipment satisfied these condition then the Time of Day is validated.

PRIORITIZED ACCESS

If the Location and Time of Day validates, the final steps is executed. This final step includes validating the correct Access Technology and Access Network Priority. Once these validity conditions are fulfilled, the highest Prioritized Access Object is selected.

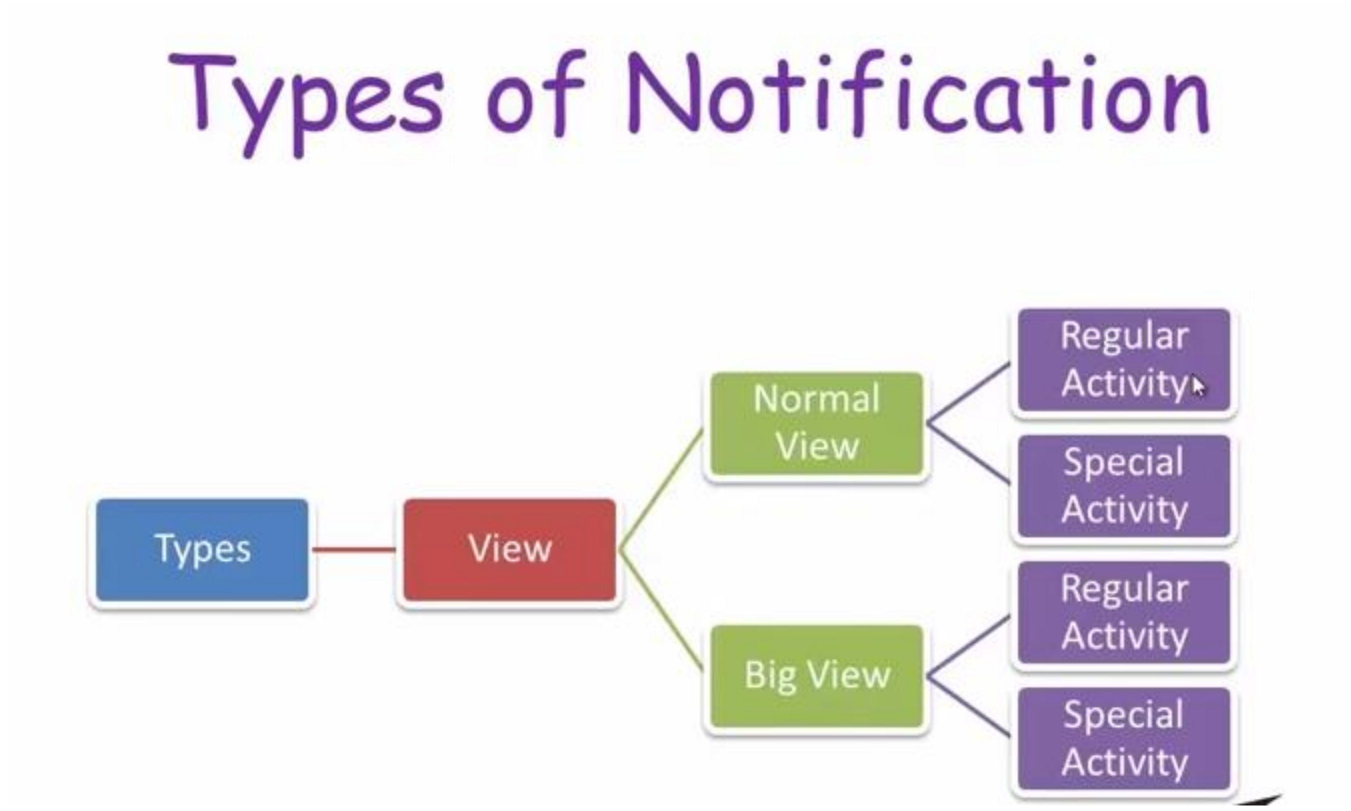
Once this is achieved successfully, the Access Id of the highest prioritized access is set to the Current Access Id which is then used as an argument of the Discovery Information method. Once all the parameters are valid, in the order stated we connect to the network to which the policy corresponds.

1.3.2.3. Custom Notification Manager Class

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

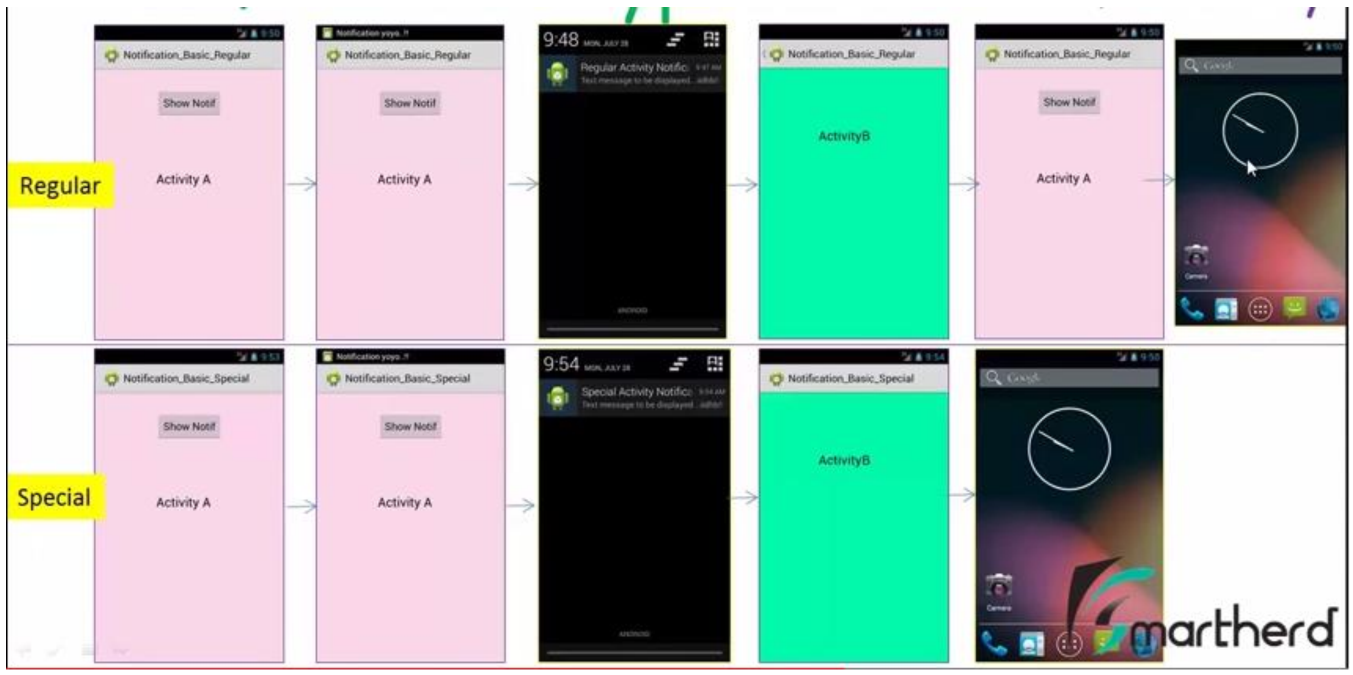
We made a custom Notification Manager class that could be used by the company as library in their apps. It contains all the possible types of notifications, which can be used directly by calling the methods without writing any code or changing the code of custom manager class. The code has been shared in the appendix.

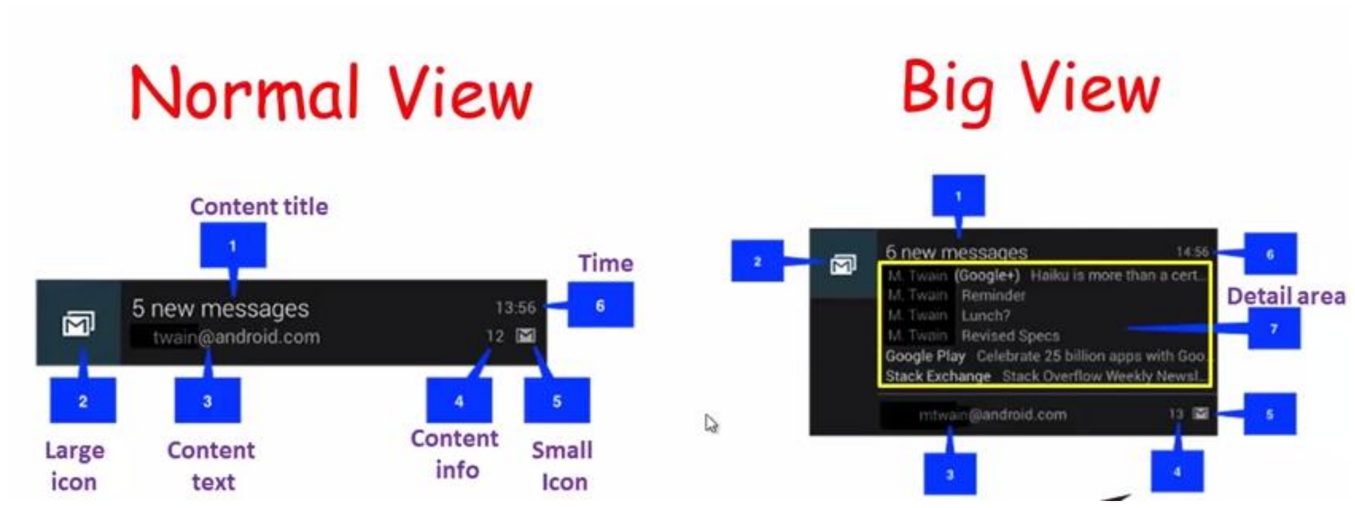
We specified the UI information and actions for a notification in a NotificationCompat.Builder object. To create the notification itself, you call NotificationCompat.Builder.build(), which returns a Notificationobject containing your specifications. To issue the notification, you pass the Notification object to the system by calling NotificationManager.notify().



Notifications can be classified based on -

1. View – Regular, Special
2. Activity – Normal View, Big View

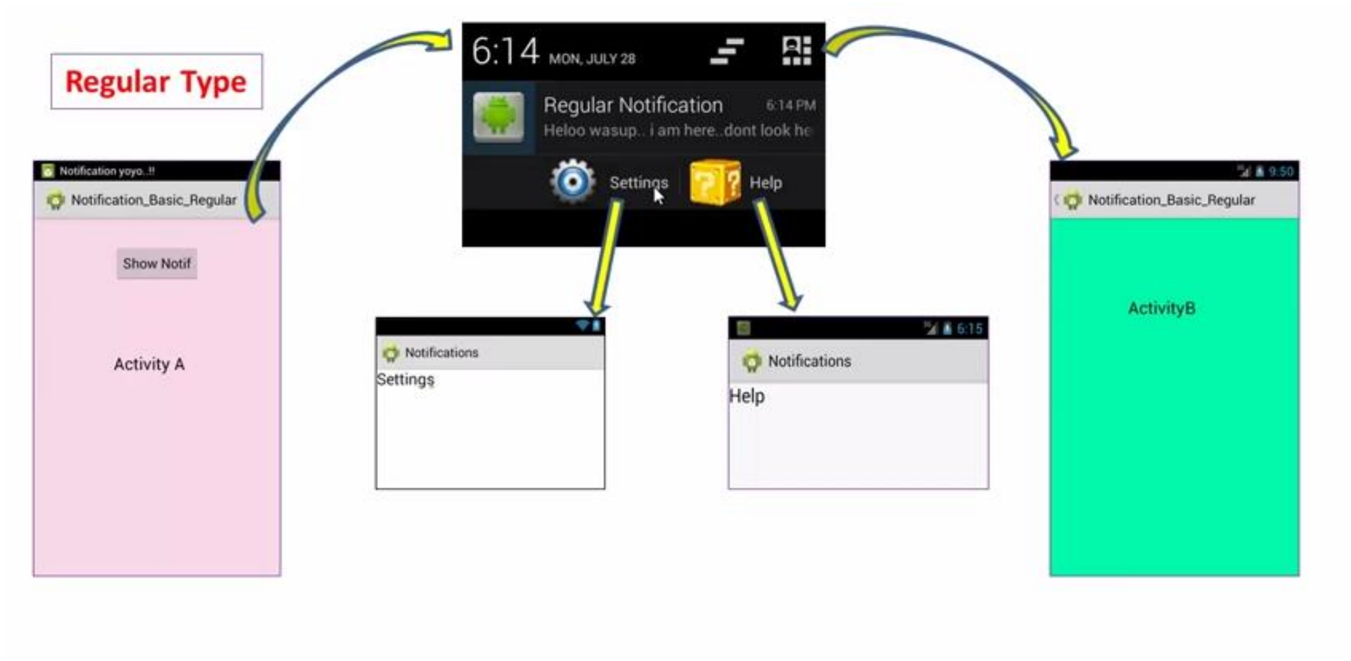




Big picture style, inbox style, big text style notifications fall under the category of Big View notifications.

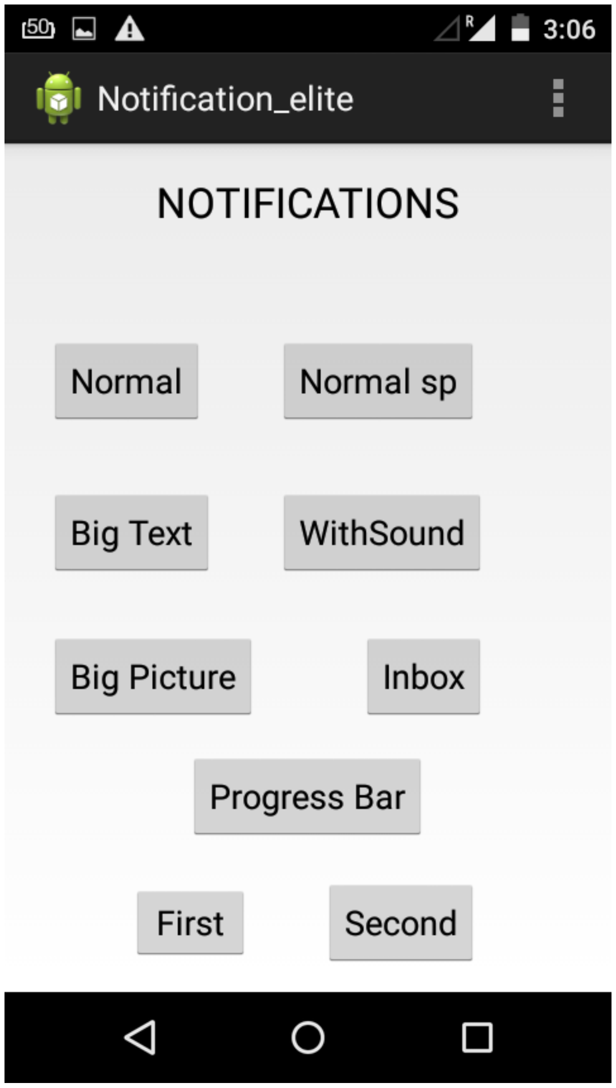
Also action buttons and activity can be added to a notification so that it functions as shown below. This can be achieved by adding intent and pending intents. Inside a Notification, the action itself is defined by a PendingIntent containing an Intent that starts anActivity in your application. To associate the PendingIntent with a gesture, call the appropriate method ofNotificationCompat.Builder. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling setContentIntent().

Starting an Activity when the user clicks the notification is the most common action scenario. You can also start an Activity when the user dismisses a notification. In Android 4.1 and later, you can start an Activityfrom an action button. To learn more, read the reference guide for NotificationCompat.Builder.

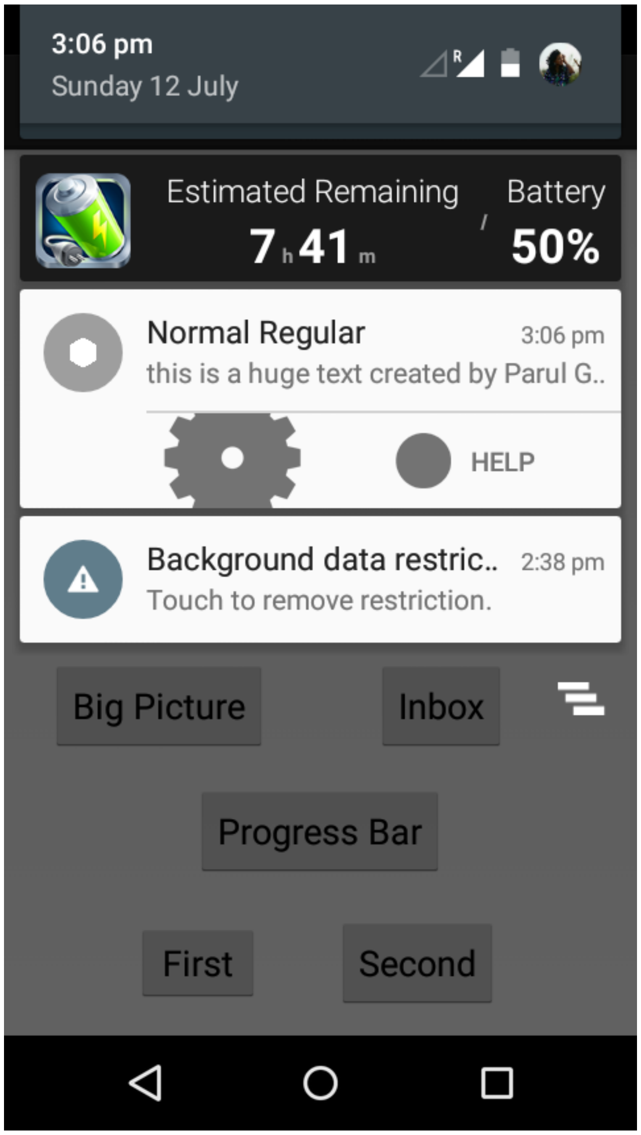


Given below are the screenshots of the android application that we created consisting of all the types of notifications.

First screen is the SPLASH screen. Then there is the home screen of the Notification_elite app.



Given below are the different screens for different type of notifications.



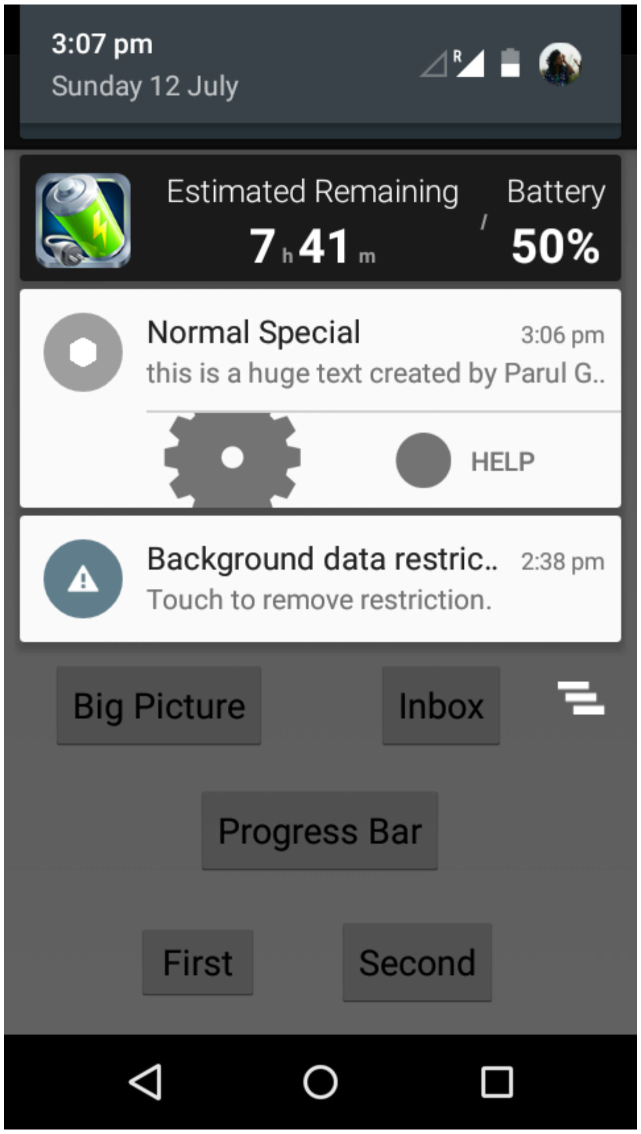
3:06 pm
Sunday 12 July

Estimated Remaining 7 h 41 m / Battery 50%

Normal Regular 3:06 pm
this is a huge text created by Parul G..

HELP

Background data restric.. 2:38 pm
Touch to remove restriction.



3:07 pm
Sunday 12 July

Estimated Remaining 7 h 41 m / Battery 50%

Normal Special 3:06 pm
this is a huge text created by Parul G..

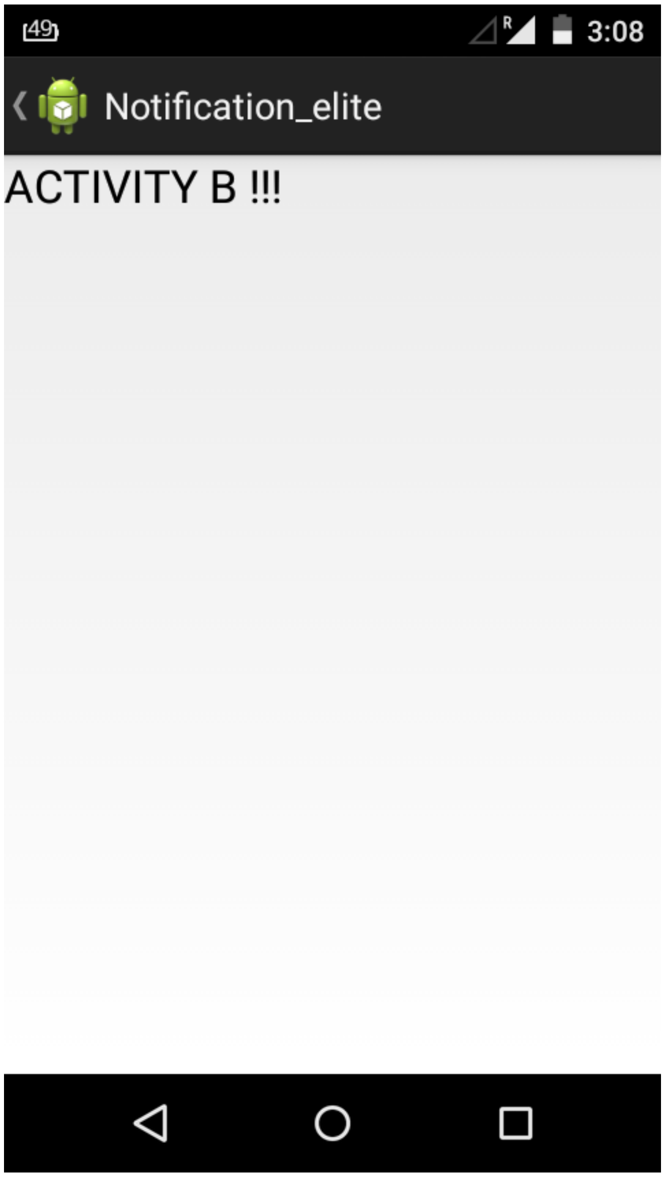
HELP

Background data restric.. 2:38 pm
Touch to remove restriction.

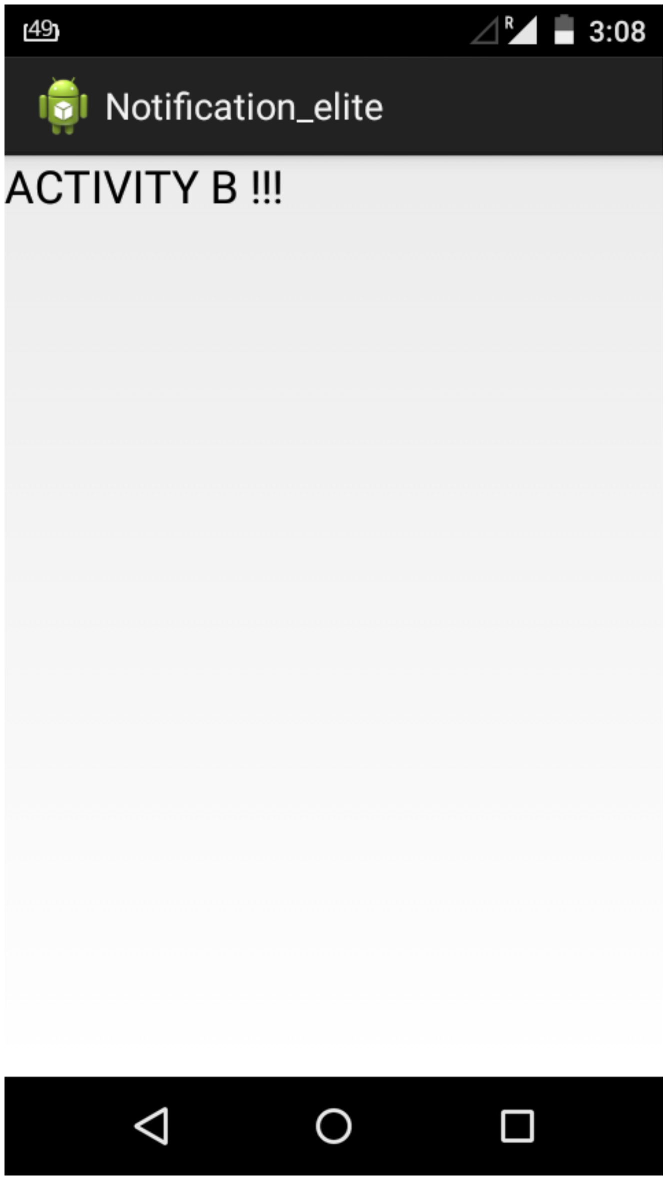
The difference between Normal View Regular Activity notification and Normal View Special Activity notification is that in regular activity, the activity that is displayed has an option to go back to the home screen whereas in case of Special Activity, such an option is not available and u have to exit from the app.

The difference isn't much but the entire code is different for these activities.

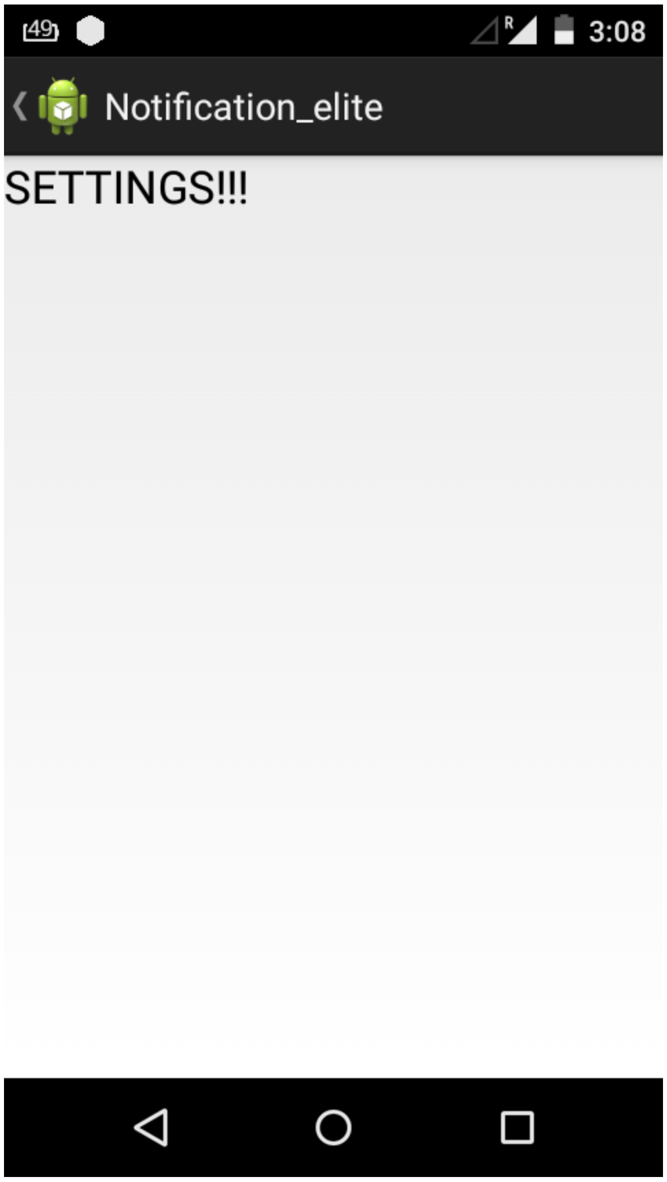
Also on clicking the action buttons Settings and Help, displays different activities containing the details.



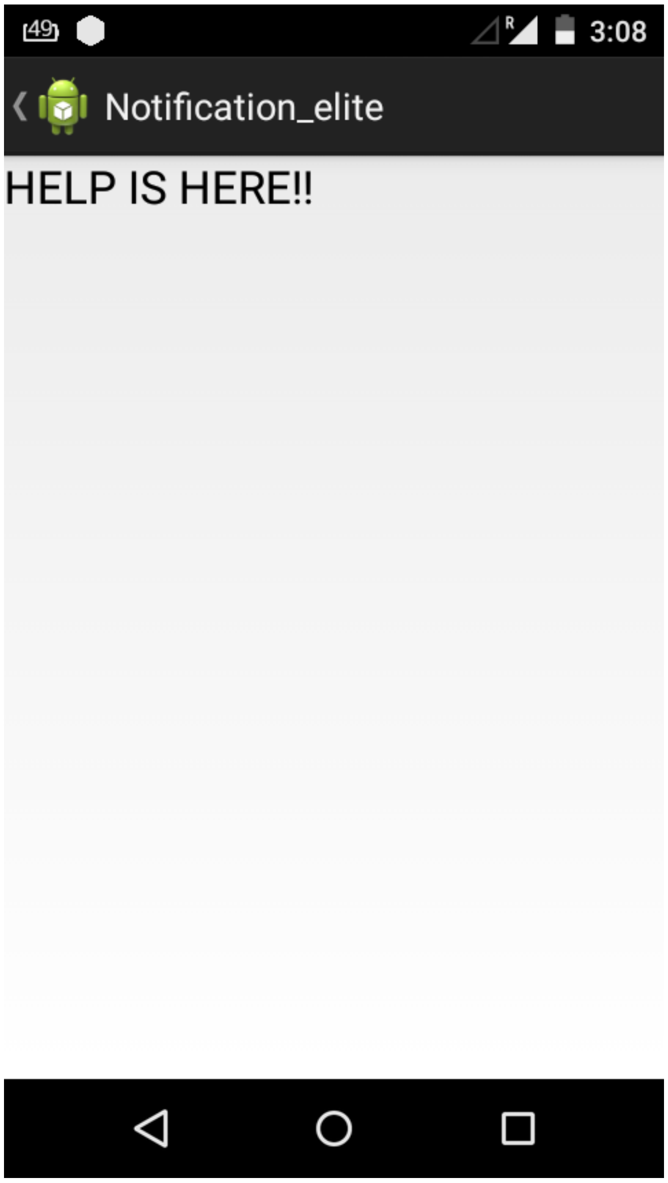
ACTIVITY B !!!



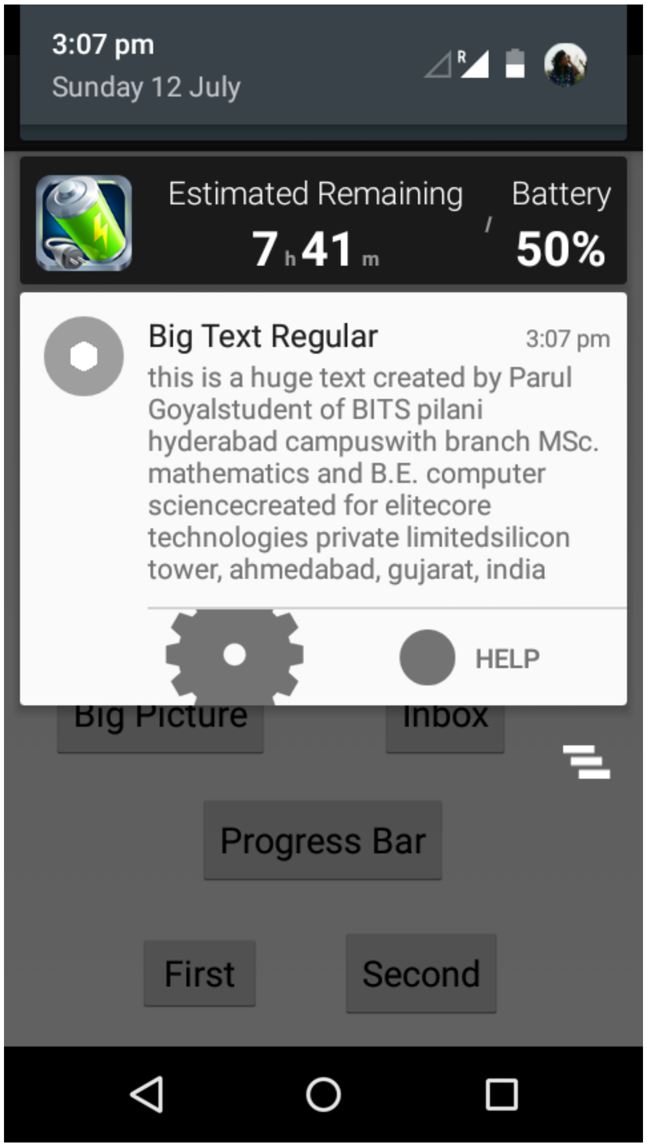
ACTIVITY B !!!



SETTINGS!!!



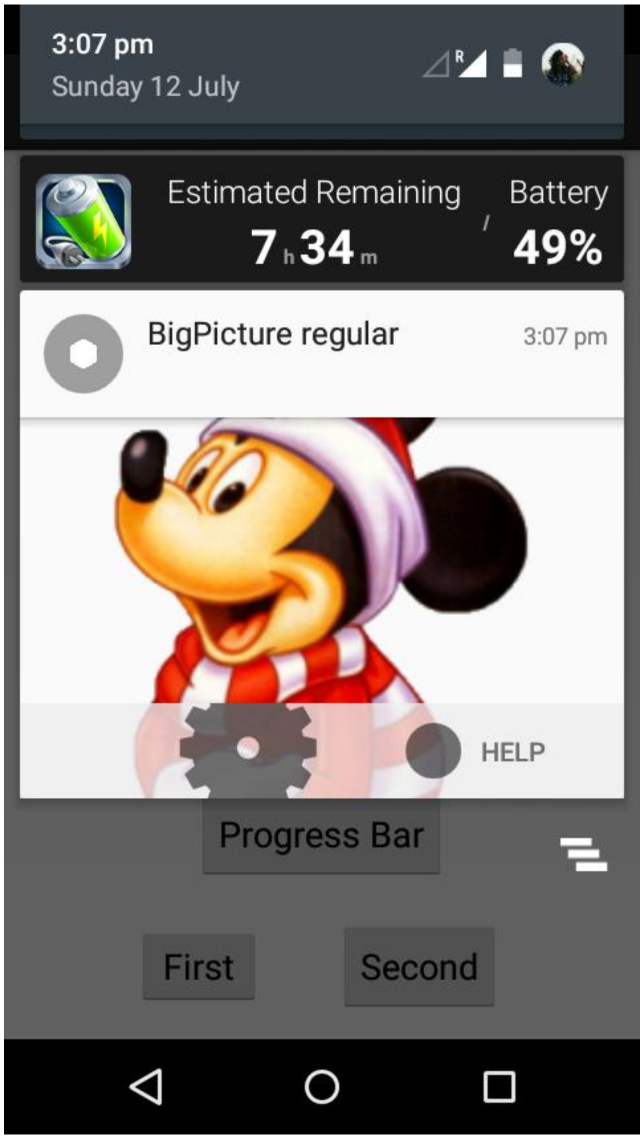
HELP IS HERE!!



3:07 pm
Sunday 12 July

Estimated Remaining 7h41m / Battery 50%

Big Text Regular 3:07 pm
this is a huge text created by Parul Goyalstudent of BITS pilani hyderabad campuswith branch MSc. mathematics and B.E. computer sciencecreated for elitecore technologies private limitedsilicon tower, ahmedabad, gujarat, india



3:07 pm
Sunday 12 July

Estimated Remaining 7h34m / Battery 49%

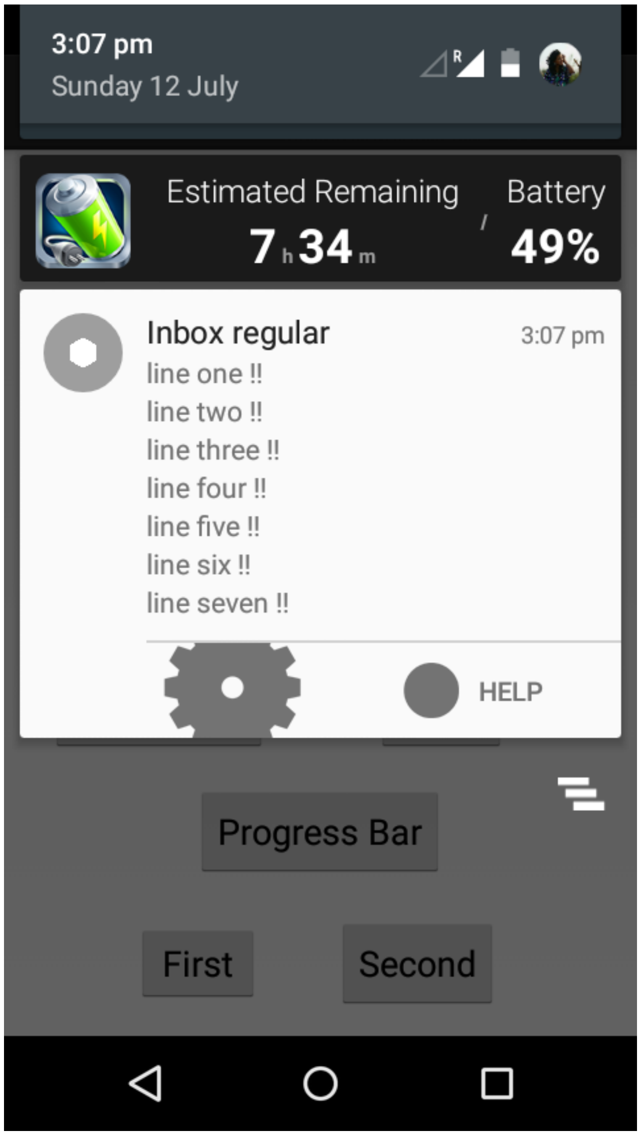
BigPicture regular 3:07 pm

Big Picture Style

The details area contains a bitmap up to 256dp tall in its detail section.

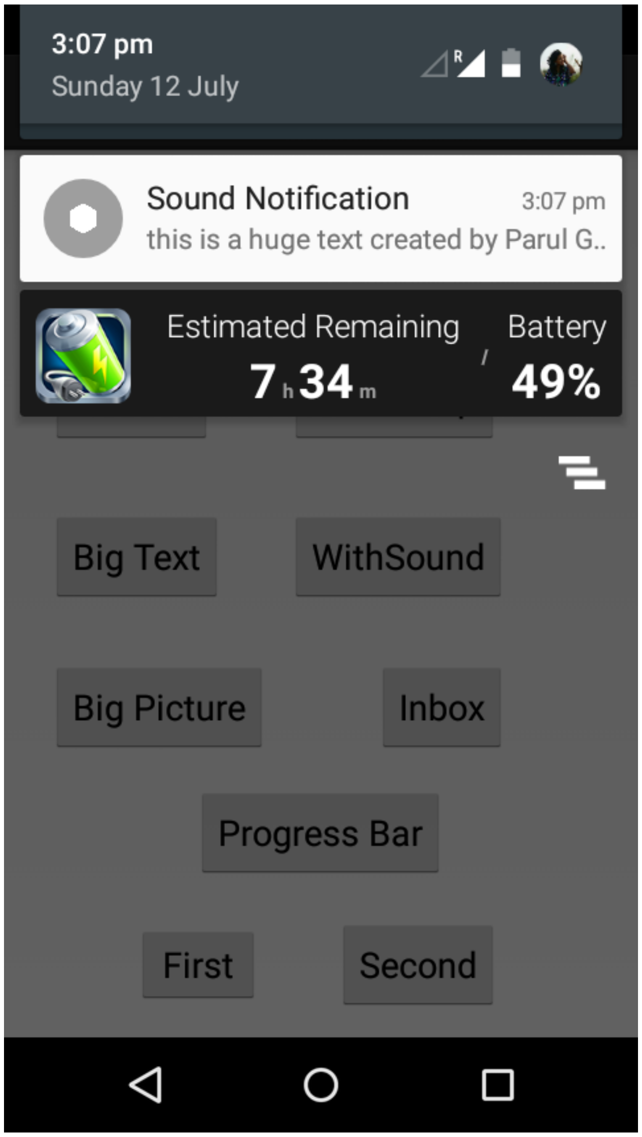
Big text style

Displays a large text block in its detail section.


3:07 pm
Sunday 12 July

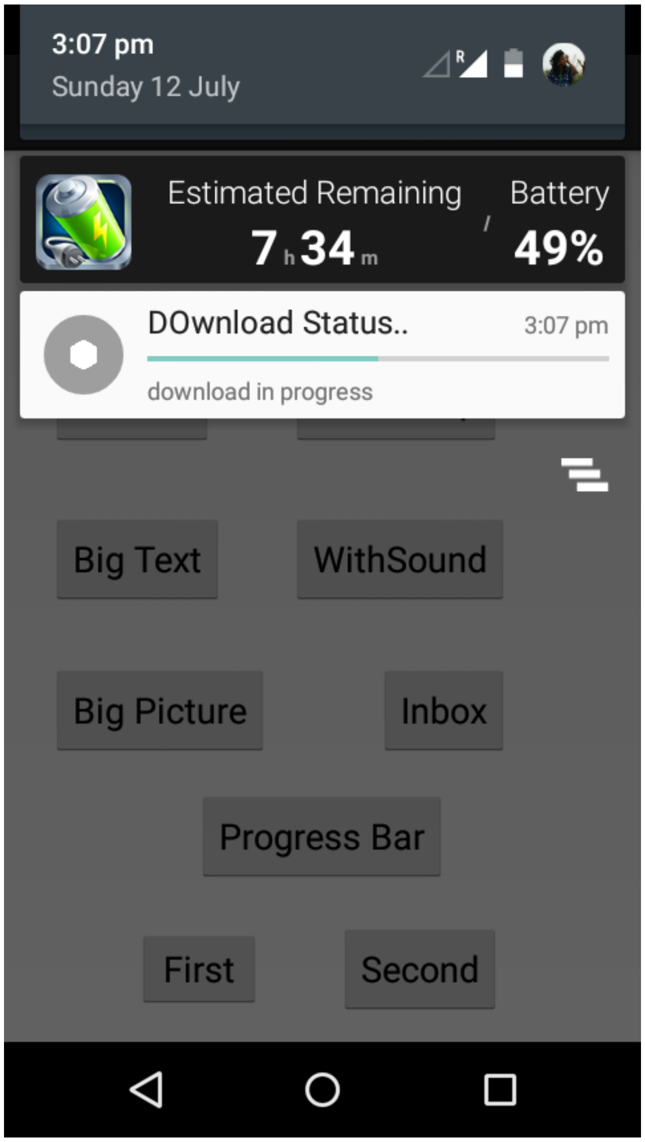
Estimated Remaining 7 h 34 m / Battery 49%

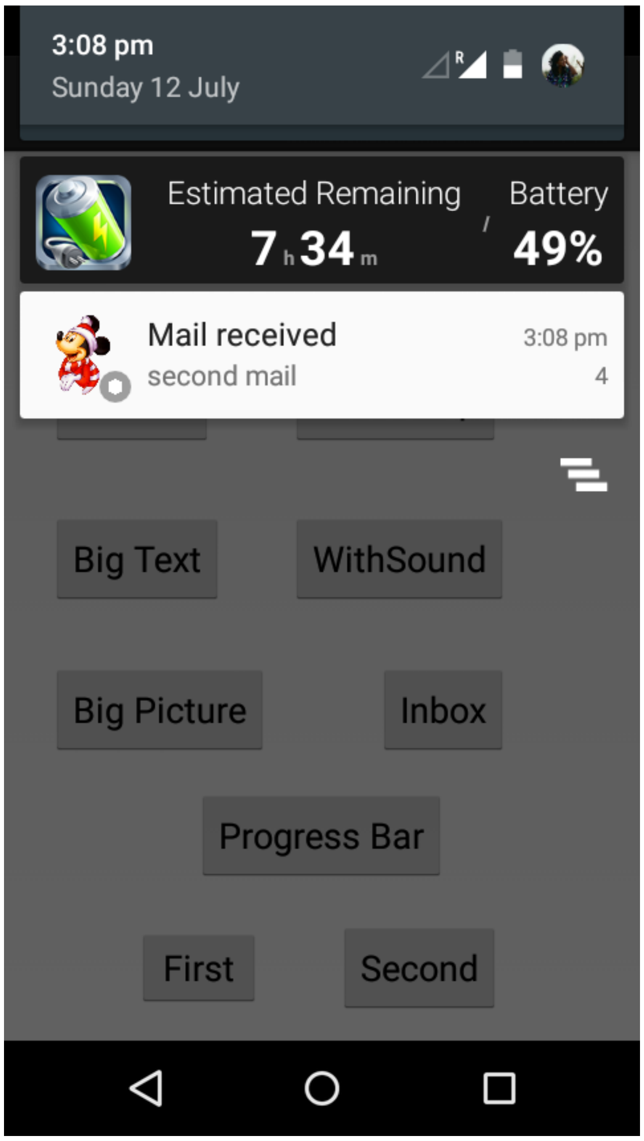
Inbox regular 3:07 pm
line one !!
line two !!
line three !!
line four !!
line five !!
line six !!
line seven !!


3:07 pm
Sunday 12 July

Sound Notification 3:07 pm
this is a huge text created by Parul G..

Estimated Remaining 7 h 34 m / Battery 49%





1.4. Appendix

1.4.1. Code for XML parser

```
package com.example.parser;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;


public class HandleXML {
   private String country = "county";
   private String temperature = "temperature";
   private String humidity = "humidity";
   private String pressure = "pressure";
   private String urlString = null;
   private XmlPullParserFactory xmlFactoryObject;
   public volatile boolean parsingComplete = true;

   public HandleXML(String url){
      this.urlString = url;
   }

   public String getCountry(){
      return country;
   }

   public String getTemperature(){
      return temperature;
   }

   public String getHumidity(){
      return humidity;
   }

   public String getPressure(){
      return pressure;
   }

   public void parseXMLAndStoreIt(XmlPullParser myParser) {
      int event;
      String text=null;

      try {
         event = myParser.getEventType();

         while (event != XmlPullParser.END_DOCUMENT) {
            String name=myParser.getName();

            switch (event){
               case XmlPullParser.START_TAG:
               break;

               case XmlPullParser.TEXT:
               text = myParser.getText();
               break;

               case XmlPullParser.END_TAG:
               if(name.equals("country")){
                  country = text;
               }

               else if(name.equals("humidity")){
                  humidity = myParser.getAttributeValue(null,"value");
               }

               else if(name.equals("pressure")){
                  pressure = myParser.getAttributeValue(null,"value");
               }

               else if(name.equals("temperature")){
                  temperature = myParser.getAttributeValue(null,"value");
               }
```

```
                  else{
                  }
                  break;
               }
               event = myParser.next();
            }
            parsingComplete = false;
         }

         catch (Exception e) {
            e.printStackTrace();
         }
      }

      public void fetchXML(){
         Thread thread = new Thread(new Runnable(){
            @Override
            public void run() {
               try {
                  URL url = new URL(urlString);
                  HttpURLConnection conn = (HttpURLConnection)url.openConnection();

                  conn.setReadTimeout(10000 /* milliseconds */);
                  conn.setConnectTimeout(15000 /* milliseconds */);
                  conn.setRequestMethod("GET");
                  conn.setDoInput(true);
                  conn.connect();

                  InputStream stream = conn.getInputStream();
                  xmlFactoryObject = XmlPullParserFactory.newInstance();
                  XmlPullParser myparser = xmlFactoryObject.newPullParser();

                  myparser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES,
false);
                  myparser.setInput(stream, null);

                  parseXMLAndStoreIt(myparser);
                  stream.close();
               }
               catch (Exception e) {
                  e.printStackTrace();
               }
            }
         });
         thread.start();
      }
}
```

1.4.2. *Android Services*

package com.example.service;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

```
import android.content.Intent;

import android.view.View;


public class MainActivity extends Activity {


  @Override

  public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

  }


  @Override

  public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.main, menu);

    return true;

  }


  // Method to start the service

  public void startService(View view) {

    startService(new Intent(getBaseContext(), MyService.class));

  }


  // Method to stop the service

  public void stopService(View view) {

    stopService(new Intent(getBaseContext(), MyService.class));

  }

}
```

```
package com.example.service;

import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.widget.Toast;

public class MyService extends Service {

   @Override
   public IBinder onBind(Intent arg0) {
      return null;
   }

   MediaPlayer mPlayer;

   @Override
   public int onStartCommand(Intent intent, int flags, int startId) {
      // Let it continue running until it is stopped.
      Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();

      mPlayer = MediaPlayer.create(getApplicationContext(), R.raw.interpol);
      mPlayer.start();
      return START_STICKY;
   }

   @Override
   public void onDestroy() {
      super.onDestroy();
      Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
      if(mPlayer!=null && mPlayer.isPlaying()){//If music is playing already
          mPlayer.stop();//Stop playing the music
      }
   }
}
```

1.4.3. Android Wi-Fi

Splash.java

```
package com.example.wifi_connect;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;


public class Splash extends Activity
{

      @Override
      protected void onCreate(Bundle savedInstanceState) {
            // TODO Auto-generated method stub
            super.onCreate(savedInstanceState);
            setContentView(R.layout.splash);
            Thread timer = new Thread()
            {
                  public void run()
                  {
                        try{
                              sleep(1500);
                        }catch(InterruptedException e){
                              e.printStackTrace();
                        }finally{
                              Intent openStartingPoint = new
Intent("com.example.wifi_connect.ENABLE");
                              startActivity(openStartingPoint);
                        }
                  }
            };
            timer.start();
      }

      @Override
      protected void onPause() {
            // TODO Auto-generated method stub
            super.onPause();
            finish();
      }

}
```

Enable.java

```
package com.example.wifi_connect;

import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Enable extends Activity{
      Button enableButton,disableButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.enable);

        enableButton=(Button)findViewById(R.id.button1);
        disableButton=(Button)findViewById(R.id.button2);

        enableButton.setOnClickListener(new OnClickListener(){
            public void onClick(View v){
                WifiManager wifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(true);
                Intent openStartingPoint = new
Intent("com.example.wifi_connect.MAINACTIVITY");
                        startActivity(openStartingPoint);

            }
        });
        disableButton.setOnClickListener(new OnClickListener(){
            public void onClick(View v){
                WifiManager wifi = (WifiManager)
getSystemService(Context.WIFI_SERVICE);
                wifi.setWifiEnabled(false);
            }
        });
    }
```

```
   /* @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.enable, menu);
        return true;
    } */
}
```

MainActivity.java

```
package com.example.wifi_connect;


import java.util.List;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {


      TextView mainText;
    WifiManager mainWifi;
    WifiReceiver receiverWifi;
    List<ScanResult> wifiList;
    StringBuilder sb = new StringBuilder();

    public void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity_main);
       mainText = (TextView) findViewById(R.id.mainText);

       // Initiate wifi service manager
       mainWifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);

       // Check for wifi is disabled
      /* if (mainWifi.isWifiEnabled() == false)
             {
                 // If wifi disabled then enable it
                 Toast.makeText(getApplicationContext(), "wifi is disabled..making
it enabled",
                 Toast.LENGTH_LONG).show();

                 mainWifi.setWifiEnabled(true);
             } */
       // wifi scaned value broadcast receiver
       receiverWifi = new WifiReceiver();

       // Register broadcast receiver
       // Broacast receiver will automatically call when number of wifi connections
changed
       registerReceiver(receiverWifi, new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
       mainWifi.startScan();
       //mainText.setText("Starting Scan...");
    }

    public boolean onCreateOptionsMenu(Menu menu) {
        menu.add(0, 0, 0, "Refresh");
        return super.onCreateOptionsMenu(menu);
    }

    public boolean onMenuItemSelected(int featureId, MenuItem item) {
        mainWifi.startScan();
       // mainText.setText("Starting Scan");
        return super.onMenuItemSelected(featureId, item);
    }

    protected void onPause() {
        unregisterReceiver(receiverWifi);
        super.onPause();
    }

    protected void onResume() {
        registerReceiver(receiverWifi, new
IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
        super.onResume();
    }

    // Broadcast receiver class called its receive method
    // when number of wifi connections changed

    class WifiReceiver extends BroadcastReceiver {

        // This method call when number of wifi connections changed
        public void onReceive(Context c, Intent intent) {

            sb = new StringBuilder();
            wifiList = mainWifi.getScanResults();
            sb.append("\n        Number Of Wifi connections
:"+wifiList.size()+"\n\n");

            for(int i = 0; i < wifiList.size(); i++){

                sb.append(new Integer(i+1).toString() + ". ");
                sb.append((wifiList.get(i)).toString());
                sb.append("\n\n");
            }

            mainText.setText(sb);
        }

    }
}
```

Enable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
     >

<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="76dp"
        android:layout_marginTop="67dp"
        android:text="Enable Wifi" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/button1"
        android:layout_below="@+id/button1"
        android:layout_marginLeft="76dp"
        android:layout_marginTop="44dp"
        android:text="Disable Wifi" />
</LinearLayout>
```

Activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.wifi_connect.MainActivity" >

    <ScrollView android:layout_width="fill_parent"
android:layout_height="fill_parent">

     <LinearLayout
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:orientation="vertical" >
    <TextView
        android:id="@+id/mainText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="no wifi available" />
    </LinearLayout>
    </ScrollView>

</RelativeLayout>
```

1.4.4. PolicyInformation.java for ANDSF validation

```java
package com.elitecorelib.andsf.pojo;

/**
 * Created by Ashish, Parul and Anurag on 30/6/15.
 */

import android.sax.StartElementListener;
import android.util.Log;

import com.elitecore.andsfparsing.utility.Utility;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import java.util.Vector;

public class PolicyInformation {

      public String Policy_Latitude;
      public String Policy_Longitude;
      public String Policy_Radius;

      public String UE_Latitude;
      public String UE_Longitude;

      public char unit = 'M';

      public String Policy_EUTRA_CI;
      public String Policy_GERAN_CI;
      public String Policy_LAC;
      public String Policy_PLMN;
      public String Policy_TAC;
      public String Policy_UTRAN_CI;

      public String UE_EUTRA_CI;
      public String UE_GERAN_CI;
      public String UE_LAC;
      public String UE_PLMN;
      public String UE_TAC;
      public String UE_UTRAN_CI;

      public String Policy_BSSID;
      public String Policy_HESSID;
      public String Policy_SSID;

      public String UE_BSSID;
      public String UE_HESSID;
      public String UE_SSID;

      public String Policy_netmask;
      public String Policy_sector_ID;

      public String UE_netmask;
      public String UE_sector_ID;
```

```
        public String Policy_NID;
        public String Policy_SID;
        public String Policy_base_ID;

        public String UE_NID;
        public String UE_SID;
        public String UE_base_ID;

        public String dateStart;
        public String dateStop;
        public String timeStart;
        public String timeStop;

        public int accessTechnology;
        public String accessId;
        public int accessNetworkPriority;
        public String secondaryAccessId;

        public static String currentAccessId;
        int geoi = 0;

        public static DiscoveryInformation currentDiscoveryInformation;

        public static Vector<Policy> policy = new Vector<Policy>();
        public static Policy validPolicy = null;
        // public static Vector<DiscoveryInformation> discoveryInformation = new
        // Vector<DiscoveryInformation>();

        private static PolicyInformation policyInformationObj = new
    PolicyInformation();

        private PolicyInformation() {
              // EMPTY CONSTRUCTOR
              // Private so that no other class can access
        }

        public static PolicyInformation getPolicyInformation() {

              if (policyInformationObj == null) {
                    policyInformationObj = new PolicyInformation();
              }
              return policyInformationObj;
        }

        public void setPolicy(ArrayList<Policy> policyArrayList) {
              // Log.d("set policy", "success");
              for (int i = 0; i < policyArrayList.size(); i++) {
                    policy.addElement(policyArrayList.get(i));
                    // Log.d("Policy : " + i + 1, policy.elementAt(i).policyName);
              }
        }

        /*
         * public void setDiscoveryInformation(ArrayList<DiscoveryInformation>
         * discoveryInformationArrayList){ for (int i=0;i <
         * discoveryInformationArrayList.size(); i++){
         * discoveryInformation.addElement(discoveryInformationArrayList.get(i));
         * //Log.d("Discovery Information : " + i+1,
         * discoveryInformation.get(i).toString());
         *
         * } }
         */

        public void quickSort() {
              quickSort(0, getSizePolicy() - 1);
        }

        public void quickSort(int lowerIndex, int higherIndex) {
              int i = lowerIndex;
              int j = higherIndex;
              int pivot = getPolicyObject(lowerIndex + (higherIndex - lowerIndex) /
    2).rulePriority;
              while (i <= j) {
                    while (getPolicyObject(i).rulePriority < pivot) {
                          i++;
                    }
                    while (getPolicyObject(j).rulePriority > pivot) {
                          j--;
                    }
                    if (i <= j) {
                          exchangeElements(policy, i, j);
                          i++;
                          j--;
                    }
              }
              if (lowerIndex < j) {
                    quickSort(lowerIndex, j);
              }

              if (i < higherIndex) {
                    quickSort(i, higherIndex);

              }

        }

        private void exchangeElements(Vector<Policy> policy, int i, int j) {
              Collections.swap(policy, i, j);
        }

        public boolean evaluateTimeOfDay(Policy validPolicy) throws ParseException {

              ArrayList<TimeOfDay> timearray = validPolicy.timeOfDay;
              for(int i=0; i< timearray.size(); i++){
                    TimeOfDay date = timearray.get(i);
                    dateStart = date.dateStart;
                    dateStop = date.dateStop;
                    timeStart = date.timeStart;
                    timeStop = date.timeStop;



              SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
              SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");

              Date dateStart1 = dateFormat.parse(dateStart);
              Date dateStop1 = dateFormat.parse(dateStop);

              Date timeStart1 = timeFormat.parse(timeStart);
              Date timeStop1 = timeFormat.parse(timeStop);

              boolean time;
```

```
            Log.d("date start policy", dateStart);
            Log.d("date stop", dateStop);

            time = evaluateTimeOfDay(dateStart1, dateStop1, timeStart1, timeStop1);

            if (time == false) {
                  validPolicy = null;
                  return false;
            }
            }
            return true;
      }

      public boolean evaluateTimeOfDay(Date DateStart, Date DateStop,
                  Date TimeStart, Date TimeStop) throws ParseException {


            // Calender cal = Calender.getInstance();


            /*
            // get current date time with date()

            Date date = new Date();

            Date currentDate = dateFormat.parse(dateFormat.format(date));
            Date currentTime = timeFormat.parse(dateFormat.format(date)); */

            TimeOfDay date1 = new TimeOfDay();
            date1.dateStart = "2015-01-01";
            date1.timeStart = "01:15:45";

            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");

            Date currentDate1 = dateFormat.parse(date1.dateStart);
            Date currentTime1 = timeFormat.parse(date1.timeStart);



            /*SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
            SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");

            Date currentDate = dateFormat.parse(currentDate1);


            Date currentTime = timeFormat.parse(currentTime1); */

            Log.d("current date xyz", currentDate1.toString());
            Log.d("current time xyz", currentTime1.toString());



            if (currentDate1.after(DateStart) && currentDate1.before(DateStop)) {
                  if (currentTime1.after(TimeStart) &&
      currentTime1.before(TimeStop)) {
                        return true;
                  } else if (currentTime1.equals(TimeStart)) {
                        return true;
                  } else {
                        return false;
```

```
                    }
                }

                else if (currentDate1.equals(DateStart) ||
        currentDate1.equals(DateStop)) {
                    if (currentTime1.after(TimeStart) &&
        currentTime1.before(TimeStop)) {
                        return true;
                    } else if (currentTime1.equals(TimeStart)) {
                        return true;
                    } else {
                        return false;
                    }
                }

                else {
                    return false;
                }
            }



            public void evaluateAccessNetworkPriority(Policy validPolicy) {


                    ArrayList<PrioritizedAccess> PrioritizedAccessArrayList =
        validPolicy.prioritizedAccess;
                    PrioritizedAccess highPriorityObj = PrioritizedAccessArrayList
                            .get(0);
                    for (int j = 0; j < PrioritizedAccessArrayList.size(); j++) {
                        PrioritizedAccess prioritizedAccessObj =
        PrioritizedAccessArrayList
                                .get(j);

                        accessTechnology = prioritizedAccessObj.accessTechnology;
                        accessId = prioritizedAccessObj.accessId;
                        secondaryAccessId = prioritizedAccessObj.secondaryAccessId;
                        accessNetworkPriority =
        prioritizedAccessObj.accessNetworkPriority;
                        boolean validityResult = prioritizedAccessObj.validate();
                        if (validityResult) {
                            int priorityCheck = prioritizedAccessObj
                                    .compareTo(highPriorityObj);
                            if (priorityCheck == 1) {
                                highPriorityObj = prioritizedAccessObj;
                                connectToNetwork(highPriorityObj);
                            }

                        }

                    }

            }

            public void connectToNetwork(PrioritizedAccess prioritizedAccessObj) {
                currentAccessId=prioritizedAccessObj.accessId;
            }

            public boolean evaluate3GPP() {

                Location3GPP location3GPP = new Location3GPP();
            UE_EUTRA_CI = location3GPP.EUTRA_CI;
            UE_GERAN_CI = location3GPP.GERAN_CI;
            UE_LAC = location3GPP.LAC;
            UE_PLMN = location3GPP.PLMN;
            UE_TAC = location3GPP.TAC;
            UE_UTRAN_CI = location3GPP.UTRAN_CI;

            boolean range = false;

            for (int i = 0; i < policy.size(); i++) {
                  ValidityArea validityAreaObj = policy.get(i).validityArea;
                  ArrayList<Location3GPP> location3GPPArrayList =
validityAreaObj.location_3GPP;
                  for (int j = 0; j < location3GPPArrayList.size(); j++) {
                        Location3GPP location3GPPObj =
location3GPPArrayList.get(j);
                        Policy_EUTRA_CI = location3GPPObj.EUTRA_CI;
                        Policy_GERAN_CI = location3GPPObj.GERAN_CI;
                        Policy_LAC = location3GPPObj.LAC;
                        Policy_PLMN = location3GPPObj.PLMN;
                        Policy_TAC = location3GPPObj.TAC;
                        Policy_UTRAN_CI = location3GPPObj.UTRAN_CI;
                        range = evaluate3GPP(Policy_EUTRA_CI, Policy_GERAN_CI,
                                    Policy_LAC, Policy_PLMN, Policy_TAC,
Policy_UTRAN_CI,
                                    UE_EUTRA_CI, UE_GERAN_CI, UE_LAC, UE_PLMN,
UE_TAC,
                                    UE_UTRAN_CI);
                        if (range) {
                              return range;
                        }
                  }
            }
            return range;
      }

      public boolean evaluate3GPP(String Policy_EUTRA_CI, String Policy_GERAN_CI,
                  String Policy_LAC, String Policy_PLMN, String Policy_TAC,
                  String Policy_UTRAN_CI, String UE_EUTRA_CI, String UE_GERAN_CI,
                  String UE_LAC, String UE_PLMN, String UE_TAC, String UE_UTRAN_CI)
{

            if (Policy_PLMN.equalsIgnoreCase(UE_PLMN)
                        && (Policy_GERAN_CI.equalsIgnoreCase(UE_GERAN_CI)
                                    || Policy_LAC.equalsIgnoreCase(UE_LAC)
                                    || Policy_TAC.equalsIgnoreCase(UE_TAC)
                                    ||
Policy_UTRAN_CI.equalsIgnoreCase(UE_UTRAN_CI) || Policy_EUTRA_CI
                                    .equalsIgnoreCase(UE_EUTRA_CI))) {
                  return true;
            }

            return false;
      }

      public boolean evaluate3GPP2() {
            Location3GPP2 location3GPP2 = new Location3GPP2();
            Hrpd hrpd = location3GPP2.hrpd;
            Rat1X rat1X = location3GPP2.rat1x;

            UE_netmask = hrpd.netmask;
            UE_sector_ID = hrpd.sector_ID;
            UE_NID = rat1X.NID;
            UE_SID = rat1X.SID;
            UE_base_ID = rat1X.base_ID;

            boolean range = false;

            for (int i = 0; i < policy.size(); i++) {
                  ValidityArea validityArea = policy.get(i).validityArea;
                  ArrayList<Location3GPP2> location3GPP2ArrayList =
validityArea.location_3GPP2;
                  for (int j = 0; j < location3GPP2ArrayList.size(); j++) {

                        Location3GPP2 location3GPP2Obj =
location3GPP2ArrayList.get(j);
                        Hrpd hrpdObj = location3GPP2Obj.hrpd;
                        Rat1X rat1X1Obj = location3GPP2Obj.rat1x;
                        Policy_netmask = hrpdObj.netmask;
                        Policy_sector_ID = hrpdObj.sector_ID;
                        Policy_NID = rat1X1Obj.NID;
                        Policy_SID = rat1X1Obj.SID;
                        Policy_base_ID = rat1X1Obj.base_ID;

                        range = evaluate3GPP2(Policy_netmask, Policy_sector_ID,
                                    Policy_NID, Policy_SID, Policy_base_ID);
                        if (range) {
                              return range;
                        }
                  }
            }
            return range;
      }

      public boolean evaluate3GPP2(String Policy_netmask,
                  String Policy_sector_ID, String Policy_NID, String Policy_SID,
                  String Policy_base_ID) {

            if (Policy_netmask.equals(UE_netmask)
                        && Policy_sector_ID.equals(UE_sector_ID)) {
                  return true;
            } else if (Policy_SID.equals(UE_SID)
                        && (Policy_NID.equals(UE_NID) || Policy_base_ID
                                    .equals(UE_base_ID))) {
                  return true;
            }
            return false;
      }

      public boolean evaluateWLAN() {

            WlanLocation wlanLocation = new WlanLocation();
            UE_BSSID = wlanLocation.BSSID;
            UE_SSID = wlanLocation.SSID;
            UE_HESSID = wlanLocation.HESSID;

            if (UE_HESSID == null && UE_BSSID == null && UE_SSID == null) {
                  return false;
            }

            boolean range = false;
            for (int i = 0; i < policy.size(); i++) {
                  ValidityArea validityArea = policy.get(i).validityArea;
                  ArrayList<WlanLocation> wlanLocationArrayList = validityArea.
                              WLAN_Location;
                  for (int j = 0; j < wlanLocationArrayList.size(); j++) {
                        WlanLocation wlanLocationObj =
      wlanLocationArrayList.get(j);
                        Policy_HESSID = wlanLocationObj.HESSID;
                        Policy_SSID = wlanLocationObj.SSID;
                        Policy_BSSID = wlanLocationObj.BSSID;

                        range = evaluateWLAN(Policy_BSSID, Policy_HESSID,
      Policy_SSID,
                                    UE_BSSID, UE_HESSID, UE_SSID);
                        if (range) {
                              return range;
                        }
                  }
            }
            return range;
      }

      public boolean evaluateWLAN(String Policy_BSSID, String Policy_HESSID,
                  String Policy_SSID, String UE_BSSID, String UE_HESSID,
                  String UE_SSID) {

            if (Policy_BSSID.equals(UE_BSSID) || Policy_HESSID.equals(UE_HESSID)
                        || Policy_SSID.equals(UE_SSID)) {
                  return true;
            }
            return false;
      }

      public boolean isWithinRange(float UE_Latitude, float UE_Longitude,
                  float policy_Latitude, float policy_Longitude, float radius,
                  char unit) {
            double dist = Utility.getDistanceBetweenLocations(UE_Latitude,
                        UE_Longitude, policy_Latitude, policy_Longitude, unit);
            if (dist <= radius) {
                  return true;
            } else {
                  return false;
            }

      }

      public boolean evaluateGeoLocation() {

            UeLocation ueLocationObj = new UeLocation();
            //Geo_Location geoLocationObj = ueLocationObj.geoLocation;
            //UE_Latitude = geoLocationObj.latitude;
            //UE_Longitude = geoLocationObj.longitude;

            //          if (UE_Latitude == null || UE_Longitude == null) {
            //                return false;
            //          }

            float ue_latitude = Float.parseFloat("23.039568000000003");
            float ue_longitude = Float.parseFloat("72.56600399999999");

            boolean range = false;
            for (int i = geoi; i < policy.size(); i++) {

                  ValidityArea validityAreaObj = policy.get(i).validityArea;
                  ArrayList<Geo_Location_1> geo_location_1ArrayList =
      validityAreaObj.
                              geo_Location_;
                  for (int j = 0; j < geo_location_1ArrayList.size(); j++) {
                        Vector<Circular> cirObj = geo_location_1ArrayList.get(j);
                        for (int k = 0; k < cirObj.size(); k++) {
                              Policy_Latitude = cirObj.get(k).latitude;
                              Policy_Longitude = cirObj.get(k).longitude;
                              Policy_Radius = cirObj.get(k).radius;
                              float policy_latitude =
      Float.parseFloat(Policy_Latitude);
                              float policy_longitude =
      Float.parseFloat(Policy_Longitude);
                              float policy_radius =
      Float.parseFloat(Policy_Radius);
                              range = isWithinRange(ue_latitude, ue_longitude,
                                          policy_latitude, policy_longitude,
      policy_radius,
                                          unit);
                              if (range) {
                                    validPolicy = policy.get(i);
                                    geoi = i + 1;
                                    return range;
                              }
                        }
                  }
            }
            return range;
      }

      public int evaluateLocation() {

            // if flag = 0 ---> No location found
            // if flag = 1 ---> geoLocation found
            // if flag = 2 ---> WLAN Location found
            // if flag = 3 ---> 3GPP Location found
            // if flag = 4 ---> 3GPP2 Location found

            int flag = 0;
            boolean geo = evaluateGeoLocation();
            if (geo) {
                  flag = 1;
                  return flag;
            } else {
                  boolean WLAN = evaluateWLAN();
                  if (WLAN) {
                        flag = 2;
                        return flag;
                  } else {
                        boolean location3GPP = evaluate3GPP();
                        if (location3GPP) {
                              flag = 3;
                              return flag;
                        } else {
                              boolean location3GPP2 = evaluate3GPP2();
                              if (location3GPP2) {
```

```
                                        flag = 4;
                                        return flag;
                                    }
                               }
                          }
                     }
                     return flag;

                }

                public void evaluate(){
                     int location = evaluateLocation(); // to check whether UE in range or
                     // not
                     boolean time = false;
                     if (location == 0) {
                          Log.d("STATUS", "NO POLICIES AVAILABLE");
                     }

                     else {
                          /*try {
                               time = evaluateTimeOfDay(validPolicy);
                          } catch (java.text.ParseException e) {
                               e.printStackTrace();
                          }*/
                          if (true) {
                               Log.d("valid Policy", validPolicy.policyName);
                               //ArrayList<PrioritizedAccess> temp =
           validPolicy.prioritizedAccess;
                               evaluateAccessNetworkPriority(validPolicy);
                               Log.d("Current Access ID",currentAccessId);
                          }
                          else{
                               //evaluate();
                          }
                     }
                }

                public void evaluatePolicy(ArrayList<Policy> policyArrayList) {

                     setPolicy(policyArrayList); // to set the policies
                     quickSort(); // to sort the policies on the basis of rule priority
                     evaluate();

                }

                // public void getValidPolicy(obj){
                // Log.d("policy INFORMATION", obj.toString())
                // }

                public int getSizePolicy() {
                     return policy.size();
                }

                public Policy getPolicyObject(int i) {
                     return policy.elementAt(i);
                }

                public void findDiscoveryInformation(PolicyResponse policyResponse) {
```

```
                              List<DiscoveryInformation> discoveryInformationList
=policyResponse.discoveryInformation;


            if(discoveryInformationList != null &&
                        !discoveryInformationList.isEmpty() &&
currentAccessId!=null &&
                        !"".equals(currentAccessId)){


                  for(DiscoveryInformation
                              discoveryInformation:discoveryInformationList){


                        if(discoveryInformation.accessNetworkInformationWLAN!=null
&&

      discoveryInformation.accessNetworkInformationWLAN.SSIDName!=null &&

      discoveryInformation.accessNetworkInformationWLAN.SSIDName.equalsIgnoreCase(c
urrentAccessId)){

                              currentDiscoveryInformation = discoveryInformation;

                              break;

                        }

                  }

            }

      }


}
```

1.4.5. CustomNotificationManager.java

```
package com.example.notification_elite;

import java.lang.reflect.Array;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.net.Uri;
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.TaskStackBuilder;
import android.support.v7.app.ActionBarActivity;

public class CustomNotificationManager extends ActionBarActivity {

      int version = Integer.valueOf(android.os.Build.VERSION.SDK);
```

```
	private static CustomNotificationManager customNotificationObj;

	private CustomNotificationManager() {

	}

	public static CustomNotificationManager getCustomNotificationManager() {
		if (customNotificationObj == null) {
			customNotificationObj = new CustomNotificationManager();
		}
		return customNotificationObj;
	}

	public void showBigTextNotification_regularActivity(String contentTitle,
			String contentText, int smallIcon_id, String BigContent, String
bigTitle,
			int largeIcon_id, String ticker, boolean autoCancel, String
classOnClick)
					throws ClassNotFoundException {
		// assign big text style
		NotificationCompat.BigTextStyle style = new
NotificationCompat.BigTextStyle();
		style.setBigContentTitle(bigTitle);
		style.bigText(BigContent);
		Bitmap bmp = BitmapFactory.decodeResource(this.getResources(),
				largeIcon_id);
		// building content
		NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
		builder.setContentTitle(contentTitle);
		builder.setContentText(contentText);
		builder.setSmallIcon(smallIcon_id);
		builder.setLargeIcon(bmp);
		builder.setTicker(ticker);
		builder.setAutoCancel(autoCancel);

		builder.setStyle(style);
		Class<?> c = Class.forName(classOnClick);
		// provide intent for notification
		Intent i = new Intent(this, c);

		// task builder and pending intent
		TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
		stackBuilder.addParentStack(c);
		stackBuilder.addNextIntent(i);
		PendingIntent pi_main = stackBuilder.getPendingIntent(0,
				PendingIntent.FLAG_UPDATE_CURRENT);

		builder.setContentIntent(pi_main);

		// notification manager
		Notification notification = builder.build();
		NotificationManager manager = (NotificationManager) this
				.getSystemService(NOTIFICATION_SERVICE);
		manager.notify(222, notification);
	}

	public void showInboxStyleNotification_regularActivity(String contentTitle,
			String contentText, int smallIcon_id, int numberOfLines,
			int largeIcon_id, String ticker, boolean autoCancel,
                Array arrayOfLines, String classOnClick)
                throws ClassNotFoundException {
            // assign inbox style
            NotificationCompat.InboxStyle style = new
NotificationCompat.InboxStyle();
            style.setBigContentTitle(contentTitle);
            for (numberOfLines = 0; numberOfLines < Array.getLength(arrayOfLines);
numberOfLines++) {
                style.addLine((CharSequence) Array.get(arrayOfLines,
numberOfLines));
            }

            Bitmap bmp_icon = BitmapFactory.decodeResource(this.getResources(),
                    largeIcon_id);
            // building content
            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                    this);
            builder.setContentTitle(contentTitle);
            builder.setContentText(contentText);
            builder.setSmallIcon(smallIcon_id);
            builder.setLargeIcon(bmp_icon);
            builder.setTicker(ticker);
            builder.setAutoCancel(autoCancel);

            builder.setStyle(style);

            // provide intent for notification
            Class<?> c = Class.forName(classOnClick);
            Intent i = new Intent(this, c);

            // task builder and pending intent
            TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
            stackBuilder.addParentStack(c);
            stackBuilder.addNextIntent(i);
            PendingIntent pi_main = stackBuilder.getPendingIntent(0,
                    PendingIntent.FLAG_UPDATE_CURRENT);

            builder.setContentIntent(pi_main);
            // notification manager
            Notification notification = builder.build();
            NotificationManager manager = (NotificationManager) this
                    .getSystemService(NOTIFICATION_SERVICE);
            manager.notify(444, notification);
        }

        public void showBigPictureNotification_regularActivity(String contentTitle,
                String contentText, int bigPicture_id, int smallIcon_id,
                int largeIcon_id, String ticker, boolean autoCancel,
                String classOnClick) throws ClassNotFoundException {
            // assign big picture style
            Bitmap bmp = BitmapFactory.decodeResource(this.getResources(),
                    bigPicture_id);
            NotificationCompat.BigPictureStyle style = new
NotificationCompat.BigPictureStyle();
            style.setBigContentTitle(contentTitle);
            style.bigPicture(bmp);
            Bitmap bmp_icon = BitmapFactory.decodeResource(this.getResources(),
                    largeIcon_id);
            Class<?> c = Class.forName(classOnClick);
            // building content
            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                    this);
            builder.setContentTitle(contentTitle);
            builder.setContentText(contentText);
            builder.setSmallIcon(smallIcon_id);
            builder.setLargeIcon(bmp_icon);
            builder.setTicker(ticker);
            builder.setAutoCancel(autoCancel);

            builder.setStyle(style);

            // provide intent for notification
            Intent i = new Intent(this, c);

            // task builder and pending intent
            TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
            stackBuilder.addParentStack(c);
            stackBuilder.addNextIntent(i);
            PendingIntent pi_main = stackBuilder.getPendingIntent(0,
                    PendingIntent.FLAG_UPDATE_CURRENT);

            builder.setContentIntent(pi_main);
            // notification manager
            Notification notification = builder.build();
            NotificationManager manager = (NotificationManager) this
                    .getSystemService(NOTIFICATION_SERVICE);
            manager.notify(333, notification);
        }

        public void showNormalNotification_regularActivity(String contentTitle,
                String contentText, int smallIcon_id, int largeIcon_id,
                String ticker, boolean autoCancel, String classOnClick)
                throws ClassNotFoundException {
            // building content
            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                    this);
            builder.setContentTitle(contentTitle);
            builder.setContentText(contentText);
            builder.setSmallIcon(smallIcon_id);
            Bitmap bmp_large = BitmapFactory.decodeResource(this.getResources(),
                    largeIcon_id);
            builder.setLargeIcon(bmp_large);
            builder.setTicker(ticker);
            builder.setAutoCancel(autoCancel);
            Class<?> c = Class.forName(classOnClick);
            // provide intent for notification
            Intent i = new Intent(this, c);

            // task builder and pending intent
            TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
            stackBuilder.addParentStack(ActivityB.class);
            stackBuilder.addNextIntent(i);
            PendingIntent pi_main = stackBuilder.getPendingIntent(0,
                    PendingIntent.FLAG_UPDATE_CURRENT);

            builder.setContentIntent(pi_main);

            // notification manager
            Notification notification = builder.build();
            NotificationManager manager = (NotificationManager) this
                    .getSystemService(NOTIFICATION_SERVICE);
            manager.notify(111, notification);
        }

        public void showNormalNotification_specialActivity(String contentTitle,
                String contentText, int smallIcon_id, int largeIcon_id,
                String ticker, boolean autoCancel, String classOnClick)
                throws ClassNotFoundException {
            Bitmap bmp_large = BitmapFactory.decodeResource(this.getResources(),
                    largeIcon_id);
            Class<?> c = Class.forName(classOnClick);
            // building content
            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                    CustomNotificationManager.this);
            builder.setContentTitle(contentTitle);
            builder.setContentText(contentText);
            builder.setSmallIcon(smallIcon_id);
            builder.setLargeIcon(bmp_large);
            builder.setTicker(ticker);
            builder.setAutoCancel(autoCancel);

            // set intents and pending intents
            Intent main_intent = new Intent(this, c);
            PendingIntent pi_main;
            if (version >= 11) {
                main_intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
                            | Intent.FLAG_ACTIVITY_CLEAR_TASK);
                pi_main = PendingIntent.getActivity(getApplicationContext(), 0,
                            main_intent, PendingIntent.FLAG_UPDATE_CURRENT);
            } else {
                pi_main = PendingIntent.getActivity(this, 0, main_intent, 0);
            }
            builder.setContentIntent(pi_main);

            // notification manager
            Notification notification = builder.build();
            NotificationManager manager = (NotificationManager) this
                    .getSystemService(NOTIFICATION_SERVICE);
            manager.notify(110, notification);
        }

        public void addActionButton(int icon, String label, String classOnClick)
                throws ClassNotFoundException {

            Class<?> c = Class.forName(classOnClick);
            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                    CustomNotificationManager.this);

            Intent intent_action = new Intent(this, c);
            TaskStackBuilder stackBuilder_action = TaskStackBuilder.create(this);
            stackBuilder_action.addParentStack(c);
            stackBuilder_action.addNextIntent(intent_action);
            PendingIntent pi_action = stackBuilder_action.getPendingIntent(0,
                    PendingIntent.FLAG_UPDATE_CURRENT);
            builder.addAction(icon, label, pi_action);

        }

        public void showIncrementalNotification(int largeIcon_id, int smallIcon_id,
                String contentTitle, String contentText, String ticker) {
            int counter = 0;

            // building content
```

```
            Bitmap bmp_large = BitmapFactory.decodeResource(this.getResources(),
                        largeIcon_id);

            NotificationCompat.Builder builder = new NotificationCompat.Builder(
                        this);
            builder.setContentTitle(contentTitle);
            builder.setContentText(contentText);
            builder.setSmallIcon(smallIcon_id);
            builder.setLargeIcon(bmp_large);
            builder.setTicker(ticker);
            builder.setAutoCancel(true);
            builder.setNumber(++counter);

            // set intent and pending intent

            // notification manager
            Notification notification = builder.build();
            NotificationManager manager = (NotificationManager) this
                        .getSystemService(NOTIFICATION_SERVICE);
            manager.notify(000, notification);
      }

      public void setNotificationSound() {
            NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
            builder.setVibrate(new long[] { 100, 100, 100, 100, 100, 100 });
            Uri soundUri = RingtoneManager
                        .getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
            builder.setSound(soundUri);
      }

      public void showProgressBarNotification(int icon_Id, String contentTitle,
                  String contentText) {

            final NotificationCompat.Builder builderP = new
NotificationCompat.Builder(
                        this);

            final NotificationManager NM = (NotificationManager) this
                        .getSystemService(NOTIFICATION_SERVICE);
            builderP.setSmallIcon(icon_Id);
            builderP.setContentTitle(contentTitle);
            builderP.setContentText(contentText);
            new Thread(new Runnable() {

                  @Override
                  public void run() {
                        // TODO Auto-generated method stub
                        int icr;
                        for (icr = 0; icr <= 100; icr += 5) {
                              builderP.setProgress(100, icr, false);
                              NM.notify(101, builderP.build());
                              try {
                                    Thread.sleep(1000);
                              } catch (InterruptedException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                              }
                        }
                        builderP.setProgress(0, 0, false);
                        builderP.setContentText("Download Complete");
                        NM.notify(101, builderP.build());
                    }
                }).start();
            }

    }
```

1.5. References

Practice School Documentation

- https://www.dnb.co.in/IndianTelecomIndustry/OverviewTI.asp
- http://www.ibef.org/industry/telecommunications.aspx
- http://www.elitecore.com/company.html

Elitecore Corporate Profile

- https://en.wikipedia.org/wiki/Telecommunications_in_India
- http://www.elitecore.com/telecompractices/services.html

Hotspot 2.0

- http://www.ruckuswireless.com/technology/hotspot2
- http://www.ruckuswireless.com/asset/watch/494