Bachelor's Thesis (UAS)

Degree Program: Information Technology

Specialization: Android Application development

2013

Amrit Poudel

MOBILE APPLICATION DEVELOPMENT FOR ANDROID OPERATING SYSTEM

–Case: NepGuide Mobile

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme | Information Technology

June 2013 | 45 pages

Instructor: Patric Granholm

'NepGuide Mobile' is a mobile application developed for NepGuide Pvt. Ltd. NepGuide Pvt. Ltd. is a newly established service providing company in Nepal. NepGuide Pvt. Ltd. provides online business and telephone directory of companies in Nepal. The objective of this thesis was to develop a mobile application for android mobile devices for NepGuide Pvt. Ltd. This thesis aims to help Android app developer and anyone interested in database application to understand the basic fundamentals of mobile application development process. The created application is capable of fetching arrays of information based on users query from the database server and logically display them on the mobile devices in the most comprehensible way possible. The thesis describes different aspects of mobile application development for Android devices.

'NepGuide Mobile' application was designed and developed for Android supportive mobile devices in Eclipse IDE using the Java programming language. As a result of this thesis, an Android mobile app was developed and tested successfully. This application offers online business directory of companies in Nepal, which is one of the prime online services provided by the NepGuide Pvt. Ltd. Nevertheless, more services and features can be added to better commercialize this application in the future.

KEYWORDS:

Android, Eclipse and Database

TABLE OF CONTENTS

FIGURES

ACRONYMS AND ABBREVIATIONS

OS	Operating System
App	Application
UI	User Interface
VM	Virtual Machine
ADT	Android Development Tools
SDK	Software Development Kit
API	Application Programming Interface
IDE	Integrated Development Environment
UML	Unified Modeling Language

1. INTRODUCTION

The mobile phone industry has been developing and growing rapidly during the last couple of years. Old mobile devices with limited capabilities are being replaced by new and advanced mobile technology supporting a wide range of mobile services. A mobile application (mobile app) is a software application designed to run on smartphones, tablet computers, and mobile devices. “Mobile app” has become a very familiar term in the world today. The popularity of mobile applications has continued to rise, as their usage has become increasingly prevalent across mobile phone users. Public demand of mobile apps and the availability of sophisticated developer tools, libraries, and frameworks have made mobile app development easy, fast and productive. Mobile apps can facilitate users with a wide range of services besides normal phone functionality such as GPS, mobile games, banking, online ticketing and many more.

NepGuide Pvt. Ltd. provides a sophisticated online business and telephone directory in Nepal. Businesses in the company’s database are categorized by business type, location and services. It also provides different types of service tools for users to create, delete, modify, edit their profile on its website. The information of the companies used in this application are either provided by the company itself or registered on its consent.

NepGuide Pvt. Ltd. runs the website nepguide.com and ‘NepGuide Mobile’ is a mobile application for this website, which provides all the features and services nepguide.com offers. The potential of the services this application can offer is immense, however, this thesis covers one of the very important services provided by NepGuide Pvt. Ltd. This application fetches general information (Name, Address, Contacts, Services, Image/Video, map) about businesses from the company’s database and displays them on mobile devices. Businesses in company’s database are categorized by business type, location and services.

The ‘NepGuide Mobile’ development process can be severed into three sections i.e., design, development, and implementation. The application design process requires good knowledge of Android operating system architecture and general understanding of the application’s intent. The second part of the process, development, is the most important and crucial part in the whole process; it requires knowledge of several programming languages, environment setup, and code debugging. Implementation is the final part of the development process which focuses on deploying the application on mobile devices after it has successfully been tested on virtual devices.

2. ANDROID OPERATING SYSTEM

Android is a comprehensive open source platform designed for mobile devices. It is championed by Google and owned by Open Handset Alliance. The goal of the alliance is to accelerate innovation in mobile computing and offer consumers a richer, less expensive, and better mobile experience. Android is the vehicle to do so. Android is a Linux-based operating system mainly used for running mobile devices such as smart phones and tablet computers. Its usability is not limited to mobile devices. Because of its open and customizable features, it is used in a wide range of electronics devices, like laptops, smart TV, cameras, headphones, wristwatches, game consoles, car CD and DVD players, home automations and many more [Marko Gargenta]. Android OS is hardware independent and runs on devices from different vendors, unlike other proprietary operating systems such as iOS (Apple Inc. products), Blackberry OS (Blackberry), S40 OS (Nokia), Windows OS (Windows Phone) etc., which are licensed and controlled by certain companies. As of May 2013, Android dominates the smartphone market accounting 74.4% of worldwide smartphone sales [Gartner].

Android is a full-fledged operating system and a complete software stack for mobile devices. Android APIs are a rich set of system services wrapped in an intuitive class files which provides easy access to several features like location, web, telephony, Wi-Fi, media, camera , and so on. All the tools, frameworks and software necessary to develop a mobile application are available for free .

3. ANDROID APPLICATION

Android app is a mobile software application developed for use on devices powered by Google's Android platform. An Android application can be written in several different programming languages. ‘NepGuide Mobile’ is written in the Java programming language. Although, this application is heavily coded on Java, it profoundly relies on a huge stack of native libraries written in C++. Application developers can easily tap into the huge stack of system services, tools and libraries to use in their application if required.

3.1 Android stack

Android is built on top of Linux. Linux is the base for all the stacks of programs in Android. There are so many reasons for choosing Linux as a base of the Android stack such as portability, security, networking, great memory and process management, and support for shared libraries.

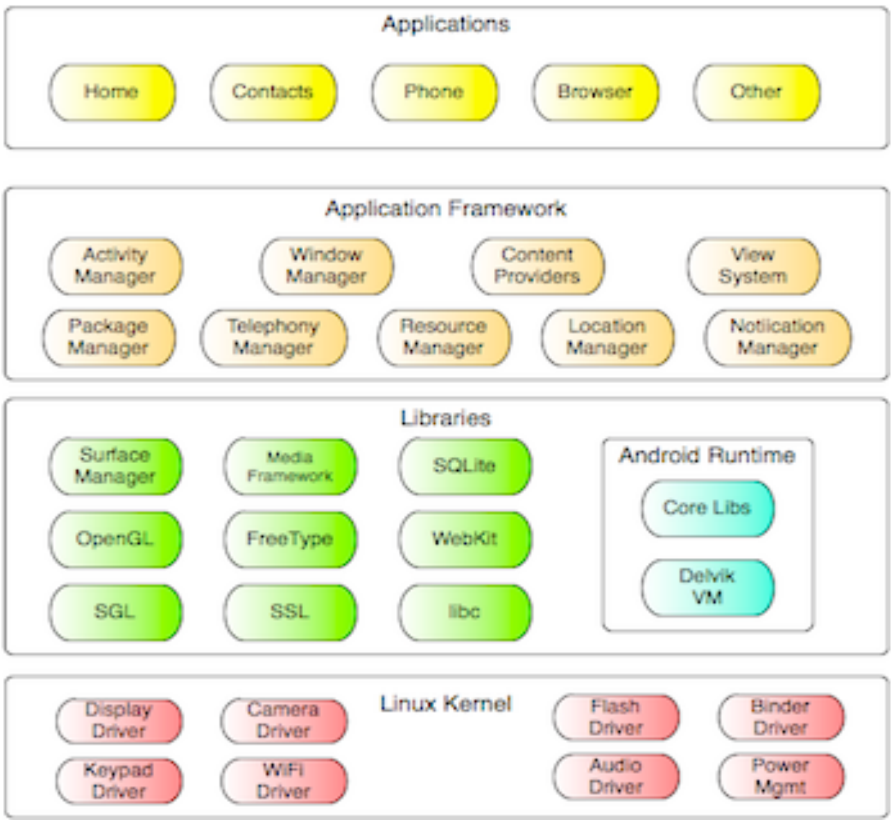


Figure 1. Android Stack

3.2. Main Building Blocks

The main building blocks are components that a developer uses to build an Android application. These components help break down the work into small conceptual units so that the application developer can work on them independently and put them together as a complete package.

There are five application components which are essential to build an Android application. These application components are very important for application developers to understand in detail because all the major actions(switching between screens/applications, database manipulation, triggering events, receiving notifications etc.) performed by an application are handled by them.

An **Activity** is an application component that provides a screen with which users can interact in order to perform certain tasks, such as dial the phone, take a photo, send an email, view a map and many more. One application can have several activities that a user flips back and forth on the device[Marko Gargenta]. Launching an Activity is the crucial part of the Android application development process. The Activity class is provided by an Android framework, which provides a wide range of facilities like displaying user interface, creating a new Linux process, and allocating memory for the UI objects. Typically, an Android application has one main activity which the user sees when the application is launched and the user can navigate to other activities as required. One activity can start/stop other activities to perform different actions in the application. When the user launches a new activity, the previous activity is stopped and the android system preserves the activity process in the stack. The previous activity can be resumed anytime by pressing the back button whenever the user is done with the current activity. Android has a very well-defined activity lifecycle. Android OS manages activities process by changing its state.

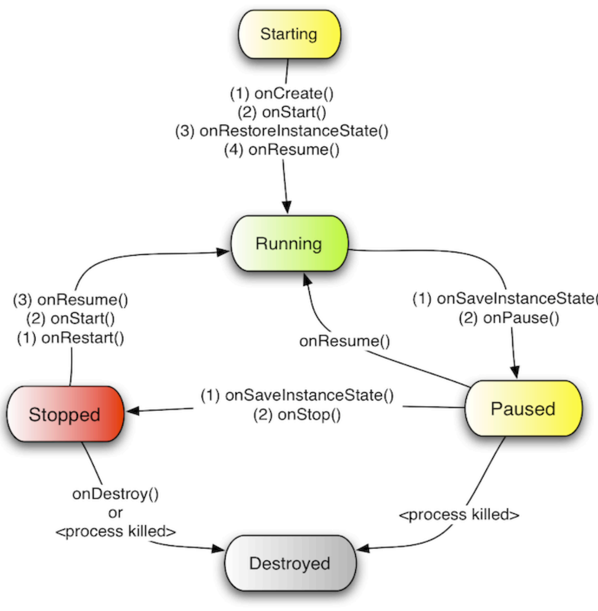


Figure 2. Android Activity Lifecycle

Intents represents actions or events that trigger an activity to start, service to start/stop, or broadcast in an application. Intents are asynchronous messages that are sent among the main building blocks. An activity sends one or several intents to another application to perform a given task, for example, open a webpage, play a media file, and so on. Applications capable of performing such tasks could compete to complete the task. If there are competing applications, Android asks the user to choose between applications, and the user can set any application as a default one.

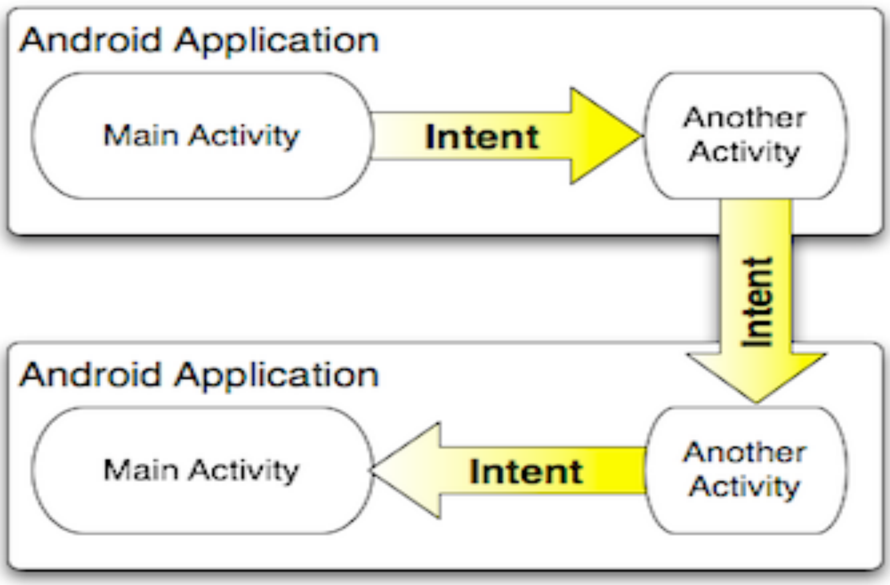


Figure 3. Android Intent to navigate from one Activity to another

A **Broadcast Receiver** is an intent-based public subscribe mechanism in Android. This application component allows users to register system events and receive notification when the registered event is triggered such as SMS notification, battery life and so on. The receiver is simply a stack of code in the application that becomes activated when a subscribed events is triggered. The system broadcasts events all the time and the broadcasted events can trigger any number of receivers. Broadcasts can be sent from one part of application to another or to a totally different application. Broadcast Receivers themselves do not have graphical representation, nor do they actively run in memory.

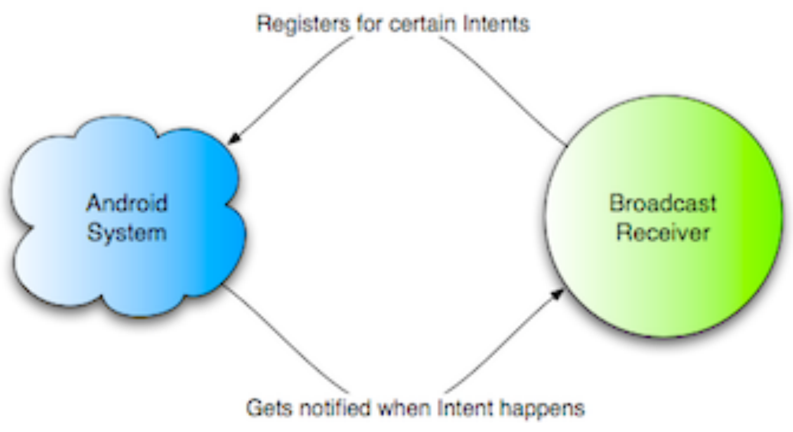

Registers for certain Intents
Android System
Broadcast Receiver
Gets notified when Intent happens

Figure 4. Android Broadcast Receiver

Services are application components that can perform long-running operations in the background. Service components run invisibly, updating the data sources and visible activities and triggering notifications. It is an application component that can start a service and continue to run in the background even when the user is switching through different mobile applications. Android OS provides and processes predefined system services that has to be declared in every Android application[Services].

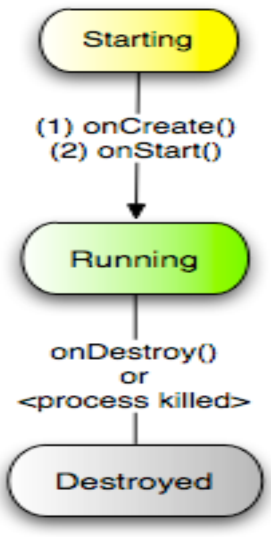


Figure 5. Android Service Lifecycle

The **Content Provider** is an application component that is used to manage and share application databases. Multiple applications can share the same data in so many different ways depending on the type of data. Multiple applications can tap into the same data source simultaneously. Content Providers are the preferred way of sharing data across application boundaries. Android itself includes native content providers that manage data such as audio, video, images, and personal contact information.

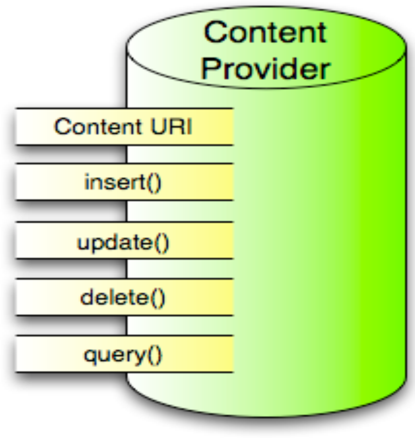


Figure 6. Android Content Provider

3.3 Native Libraries

Native libraries are a stack of software codes brought from other open source projects. Native libraries are basically C and C++ codes required to build an Android stack. There are a several libraries available in the package. Depending on the type of Android device like smartphones, home automation system, car dashboard, native libraries can be striped and modified as required. Some of the important native libraries include the following:

Media codec: Media codec provides different media codecs to supports different media formats.
SQLite: SQLite is the database engine, which provides a relational database management system.
WebKit: It is the browser engine for fast HTML rendering.
OpenGL: It is an API to render 2D or 3D computer graphics.

3.4 Programming Languages

A mobile app can be written in several different languages and platforms. However, 'NepGuide Mobile' was developed using two programming languages and a format to store and exchange structured data over a network connection known as JSON.

Java is general-purpose, structured, generic, class-based computer programming language. Android applications are written in the Java Programming language. An Android application is highly based on Java fundamentals. Java Incorporates with several powerful features and libraries of many powerful programming languages like C, C++. The reasons for picking Java as a native programming language for Android application are:

- It is easy to understand and learn
- It is platform-independent and secure
- It is object-oriented
- Java code is compiled and run by Virtual Machine

All the detailed information and proper documentation on Java can be found on http://www.oracle.com/technetwork/java/javase/overview/index.html

Extensible Markup Language (XML) is a markup language. It contains some of the very simple, scalable, and flexible text format that is both human-readable and machine-readable. It defines the set of rules to encode the document, and usability over the Internet. XML is a commonly used data format on the Internet. XML is easy to parse and manipulate programmatically. Android resources preprocess the XML into the compressed binary format and stores it on the device. Most of the User Interface layout, screen elements are declared in XML files. More information on XML can be found on http://www.w3.org/XML/ .

JSON (JavaScript Object Notation) is a lightweight text-data interchange format. JSON uses JavaScript syntax for describing data objects, but JSON is still language and platform independent [JSON Tutorial]. JSON parsers and JSON libraries exist for many different programming languages. It is easy for an application developer to read and write, and for Android devices to parse and generate. JSON is derived from the JavaScript scripting language to represent simple data structure and associative arrays which are commonly addressed as JSON objects. More detail information on JSON can be found on http://www.json.org/ .

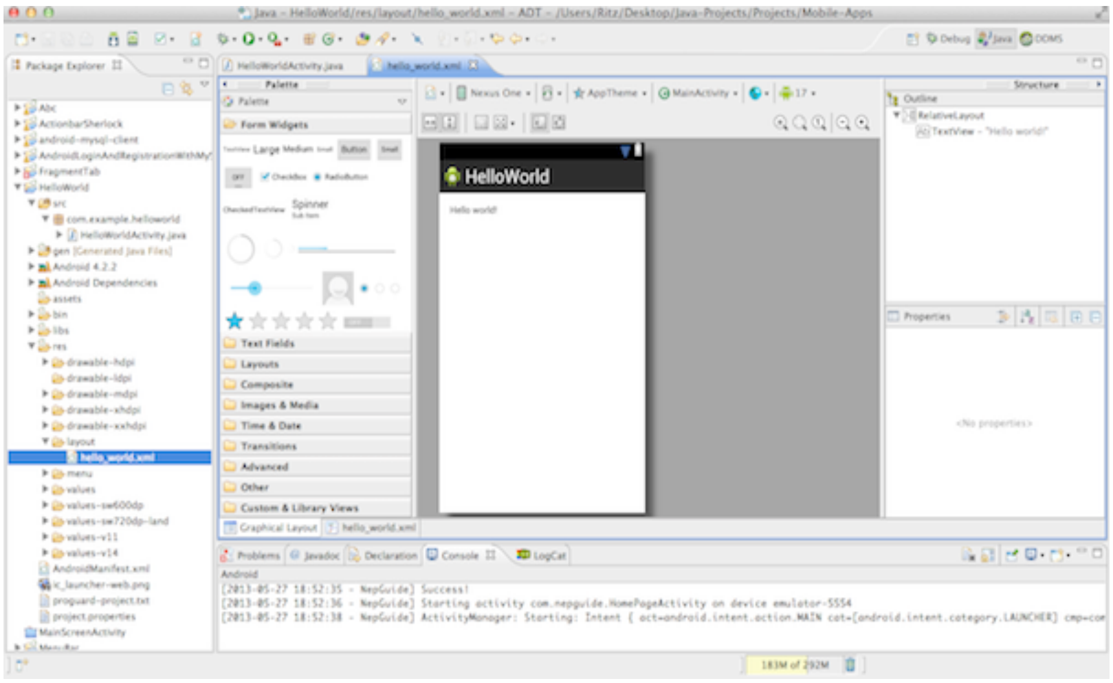
4. ENVIRONMENT SETUP

Building an environment to develop a mobile app for Android devices is rather easy. It only requires installation of Eclipse, Android SDK and Android emulator to initiate the development process -although more software and developer tools can be installed later during the process. Eclipse is considered to be the best Java development tool available, the Eclipse IDE for java developer provides superior Java editing with validation, compilation and cross-referencing. Android SDK is a software development kit that enables a developer to create applications for Android platforms. Android SDK includes application development tools, sample projects with source codes and required libraries to built Android application. The Android emulator is a virtual mobile device running on the computer. The software emulates an Android device, running the Android OS, for debugging applications without needing a variety of devices and OS versions.

'NepGuide mobile' was developed in a Macintosh system, running Mac OSX Lion as the operating system. Different versions of software are available for different operating system, depending on the operating system, the right version of the software has to be installed. All the required software has versions compatible to Mac OSX Lion. For Mac, an Android Development Tools(ADT) bundle can be downloaded from http://developer.android.com/sdk/index.html, which includes all the software programs needed to begin the application development process. If needed, more software and developer tools can be installed later during the process.

4.1 Eclipse + ADT Plug-in

Eclipse is an open source collection of programming tools originally created by IBM for Java. Nowadays, most developers in the Java community favor Eclipse as their Integrated Development Environment (IDE) of choice. Eclipse lives at http://eclipse.org [Marko Gargenta]. Eclipse is multi-language software development environment, which has tools integrated workspaces and extensible plug-in system. The ADT bundle has a version of the Eclipse IDE with a built-in ADT (Android Developer Tool) to streamline Android app development.

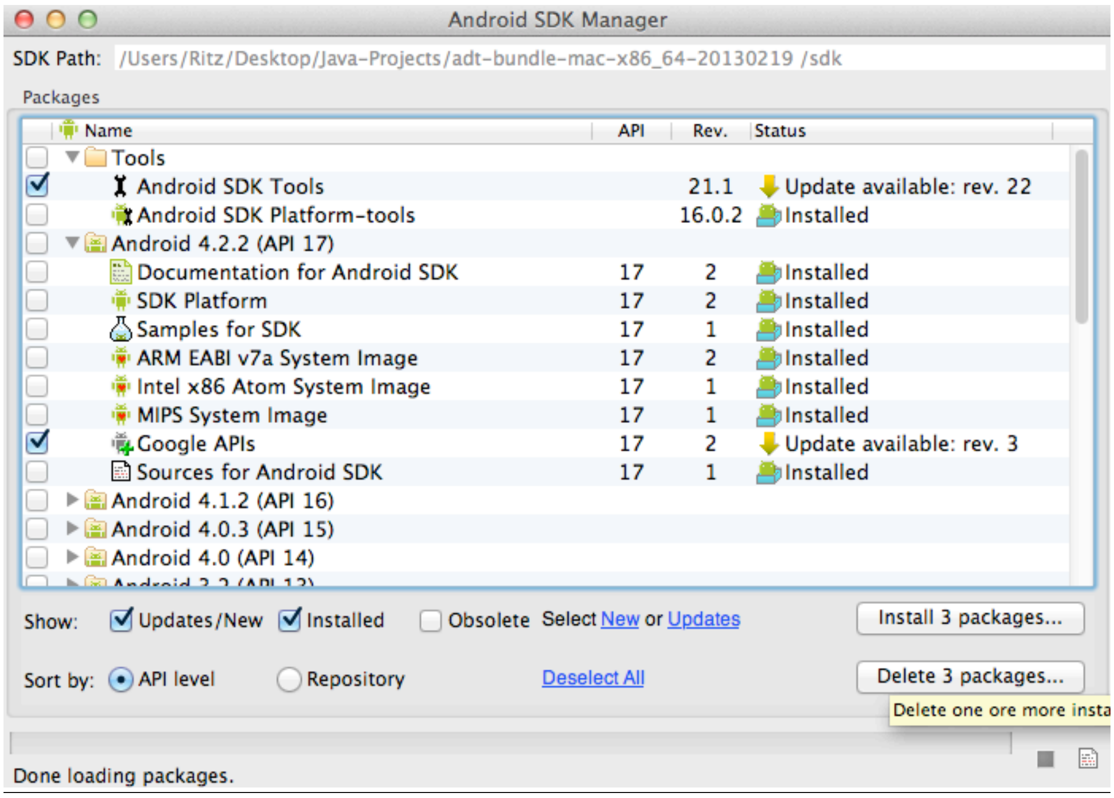


```
Android
[2013-05-27 18:52:35 - NepGuide] Success!
[2013-05-27 18:52:36 - NepGuide] Starting activity com.nepguide.HomePageActivity on device emulator-5554
[2013-05-27 18:52:38 - NepGuide] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] cmp=com
```

Figure 7. Eclipse IDE

4.2 Android System Development Kit (SDK)

The Android SDK provides all the API libraries and developer tools necessary to build, test, and debug apps for Android.[Get the Android SDK]. The ADT bundle has an IDE already loaded with SDK. By default, only the latest version of Android, API 17, is installed and as the development continues, other versions of Android have to be installed in order to support a wide range of Android mobile devices. Not all of the Android devices use the latest version of Android, so it is important for an app developer to set the API range of an app because some of the class and libraries are depreciated from a certain API level onward.



Android SDK Manager

SDK Path: /Users/Ritz/Desktop/Java-Projects/adt-bundle-mac-x86_64-20130219 /sdk

Packages

	Name	API	Rev.	Status
	Tools			
✓	Android SDK Tools		21.1	Update available: rev. 22
	Android SDK Platform-tools		16.0.2	Installed
	Android 4.2.2 (API 17)			
	Documentation for Android SDK	17	2	Installed
	SDK Platform	17	2	Installed
	Samples for SDK	17	1	Installed
	ARM EABI v7a System Image	17	2	Installed
	Intel x86 Atom System Image	17	1	Installed
	MIPS System Image	17	1	Installed
✓	Google APIs	17	2	Update available: rev. 3
	Sources for Android SDK	17	1	Installed
	Android 4.1.2 (API 16)			
	Android 4.0.3 (API 15)			
	Android 4.0 (API 14)			

Show: [✓ Updates/New] [✓ Installed] Obsolete Select New or Updates

Install 3 packages...

Sort by: [✓ API level] Repository Deselect All

Delete 3 packages...

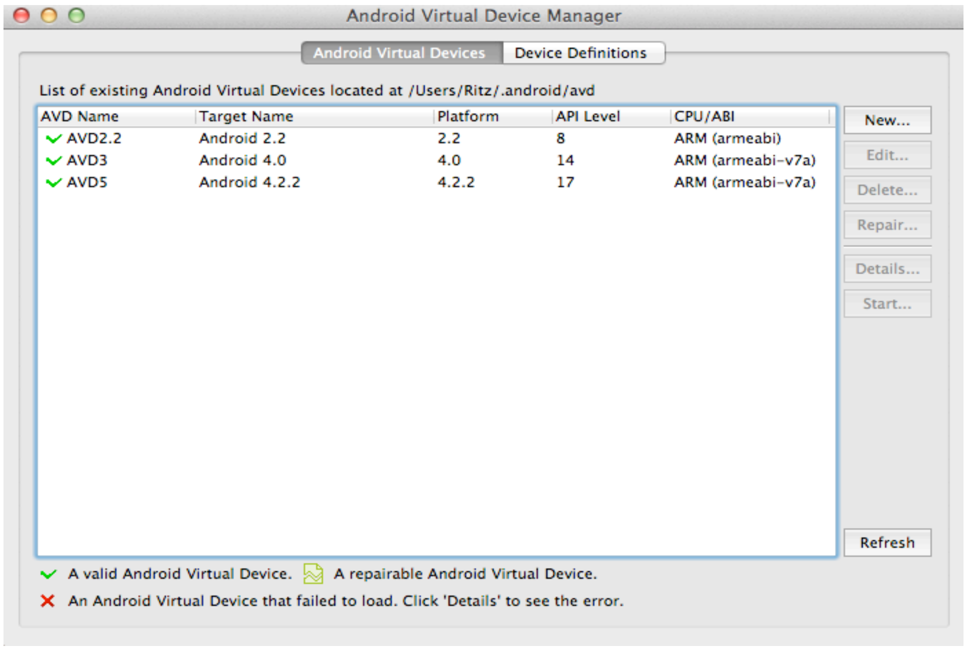
Delete one ore more insta

Done loading packages.

Figure 8. Android SDK Manager

4.3 Android Emulator

An Android emulator is a virtual Android device running on the computer. The Android emulator mimics all of the hardware and software features of a typical mobile device, except that it cannot place actual phone calls. The emulator allows an application developer to test an Android application on different API levels without using a physical device[Using the emulator]. An Android Virtual Device (AVD) is a device configuration that is run within the Android emulator. It works with the emulator to provide a virtual device-specific environment in which to install and run Android apps. The AVD Manager provides a graphical user interface in which a developer can model different configurations of Android devices, which are required by the Android emulator.


Android Virtual Device Manager

Android Virtual Devices | Device Definitions

List of existing Android Virtual Devices located at /Users/Ritz/.android/avd

AVD Name	Target Name	Platform	API Level	CPU/ABI
✓ AVD2.2	Android 2.2	2.2	8	ARM (armeabi)
✓ AVD3	Android 4.0	4.0	14	ARM (armeabi-v7a)
✓ AVD5	Android 4.2.2	4.2.2	17	ARM (armeabi-v7a)

✓ A valid Android Virtual Device. A repairable Android Virtual Device.
✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

Figure 9. AVD Manager

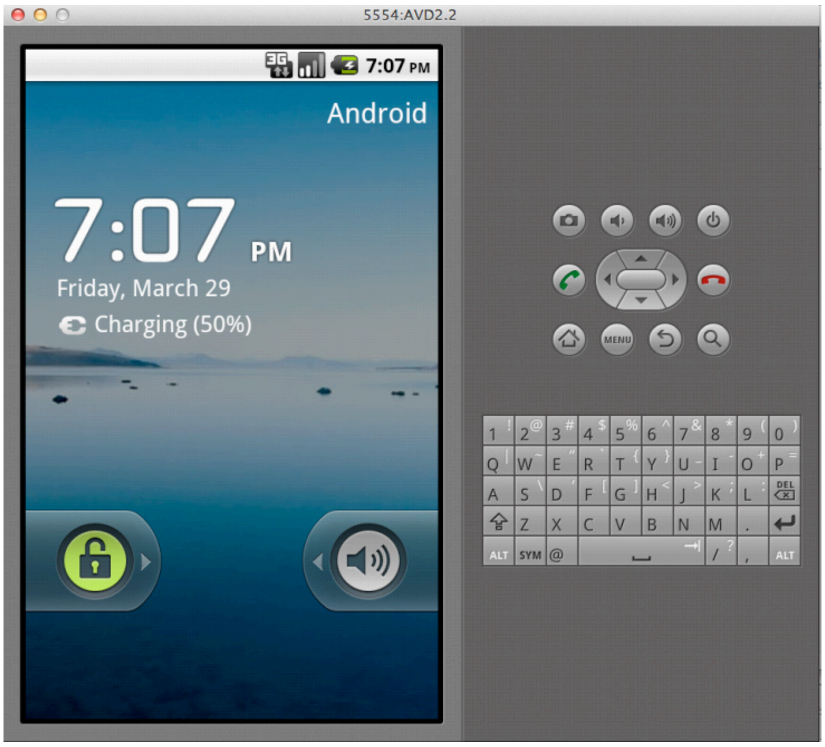


Figure 10. Android Emulator

5. APPLICATION FUNDAMENTALS

An Android application is compiled and packaged in a single file that contains all of the application's code and resources. Every piece of Java code written for an Android application is referred to as Dalvik executable code, and anything other than code are resources such as icon, animation, text, and XML files. Dalvik is an Android implementation of the Java Virtual Machine. It is the software that runs the apps on Android devices. Dalvik is thus an integral part of Android, which is typically used on mobile devices such as mobile phones and tablet computers. An Android application is commonly written in Java and compiled to Java byte code. The Java byte code is then converted from Java Virtual Machine-compatible .class files to Dalvik-compatible .dex (Dalvik Executable) files before installation on a device. The compact Dalvik Executable format is designed to be suitable for systems that are constrained in terms of memory and processor speed. Android applications are compiled and packaged into a .apk file by Android SDK. The compiled package contains all of the files necessary to run the application on a device or emulator such as compiled .dex files, a binary version of the AndroidManifest.xml file, and other resource files for an application [Building and Running].

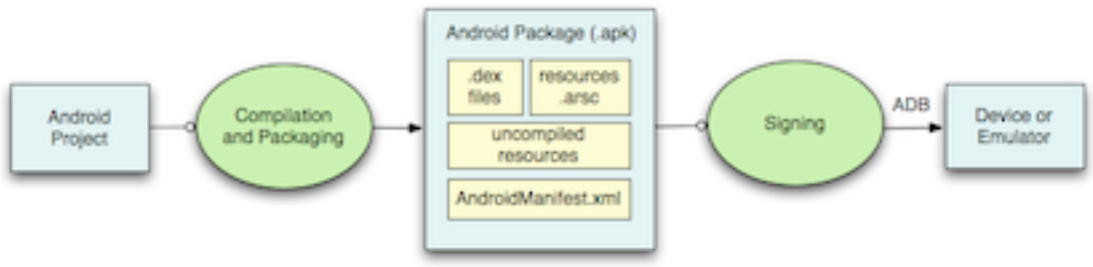


Figure 11. Android Compilation and Build Process

Other than the Java code files, Android contains lots of important program and resource files in the application. R.java and AndroidManifest.xml are two very important files in an Android application.

AndroidManifest.xml is the foundation of any Android application and is located at the root of application directory. All the activities, services , permissions are declared in this file. The AndroidManifest.xml file presents essential information about the application to the Android system, information the system must have before it can run any of the application's code[The AndroidManifest.xml File].

R.Java is a auto generated file which glues Java codes with resource files. The R.java file contains all the resource’s IDs assigned to resources (layout, style, icons, animation etc.) either by an app developer or through an Android SDK.

6. APPLICTION DESIGN

NepGuide Pvt. Ltd provides wide range of online services. ‘NepGuide Mobile’ offers services of NepGuide Pvt Ltd. through different activities(screens). ‘NepGuide Mobile’ contains many activities, each serving different purposes of the application. Events(search, edit, display) are declared in activities, each event when triggered, invokes series of actions and the results are displayed in activities. The process of retrieving contents from the server and displaying them on the mobile device can be completed in three different activities, which are different in construction from one another. ‘NepGuide Mobile’ offers many services to the users.

As agreed with NepGuide Pvt. Ltd., this thesis covers one of the very important services provided by NepGuide Pvt. Ltd. This application fetches general information (Name, Address, Contacts, Services, Image/Video, map) about businesses from the company’s database and displays them on mobile devices. The requirements of this project are met in three different activities which are as follows:

HomePageActivity: It is a homepage of the ‘NepGuide Mobile’ which contains menu, menu items, tab, search field, spinner and search button.

ListDisplayActivity: It is a activity that displays the search result.

CompanyDetailActivity: It is a page that displays the full profile of one specific company.

The ‘NepGuide Mobile’ UML diagram(Figure 12) is the detailed blueprint on how the application works.

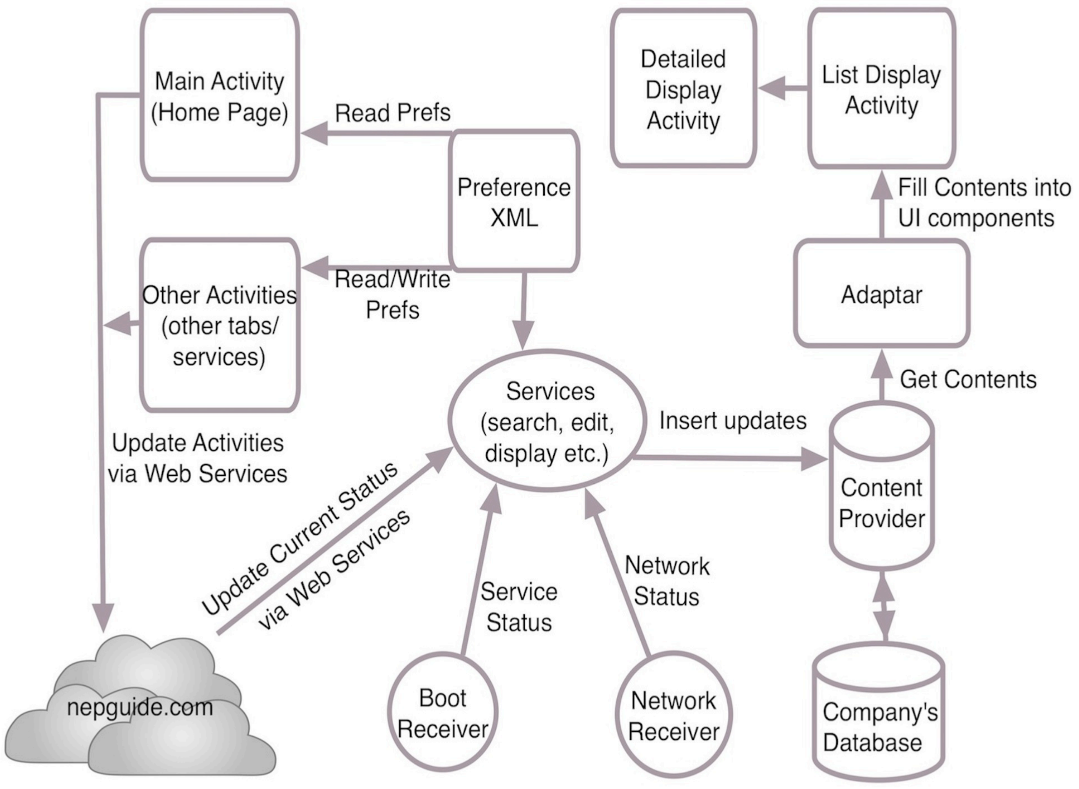

Main Activity (Home Page)
Read Prefs
Preference XML
Read/Write Prefs
Other Activities (other tabs/ services)
Detailed Display Activity
List Display Activity
Fill Contents into UI components
Adaptar
Get Contents
Services (search, edit, display etc.)
Insert updates
Content Provider
Company's Database
Update Activities via Web Services
Update Current Status via Web Services
Service Status
Network Status
Boot Receiver
Network Receiver
nepguide.com

Figure 12. 'NepGuide Mobile' UML Diagram

The prime focus of this project is one of the tabs(companies) in HomePageActivity although all other tabs, menu, menu items and activities can be designed and developed in the same fashion.

6.1 Design Process

The 'NepGuide Mobile' design process started after the environment setup was successfully achieved. The design process started with the launcher icon followed by tabs, menu, menu items and so on.

Launcher icon

All the Android apps have a launcher icon. The launcher icon is the visual representation of the app on the home screen of Android devices. The launcher icon is clicked so that the application can run and start the homepage of the application. The launcher icon can be designed using image-editing software or created using the default tools and options provided by Android SDK. The launcher icon must have a .png extension and an ic_launcher prefix.

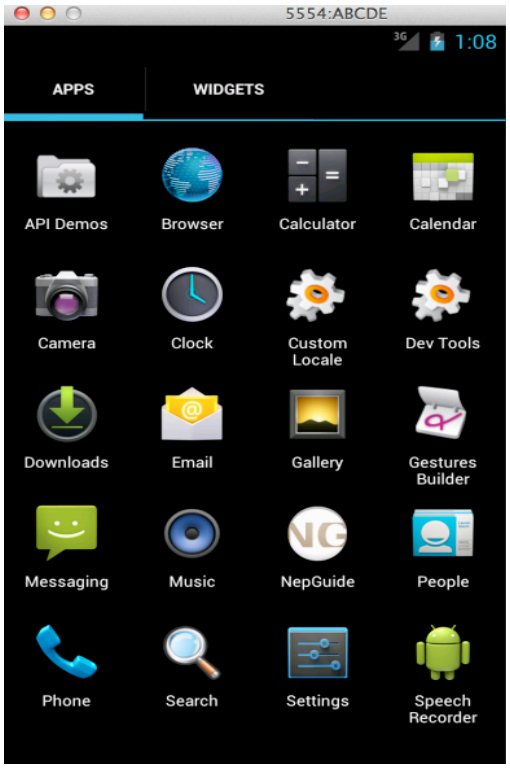


Figure 13. Applications icons on Android Device with NepGuide launcher icon

Tab

'NepGuide Mobile' has three tabs designed and created on the homepage of the application. Tabs in Android are created using ActionBar API. The ActionBar APIs were first added in Android 3.0 (API level 11) but they are also available in the Support Library for compatibility with Android 2.1 (API level 7) and above. The code below corresponds to the tab creation on 'NepGuide Mobile' (Appendix 1.0 HomePageActivity.java).

```
public void onCreate(Bundle savedInstanceState) {
```

```
StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
 super.onCreate(savedInstanceState);

 final ActionBar actionBar = getActionBar();
 actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
 Tab FragmentYP = actionBar.newTab();
 FragmentYP.setText("Companies");
 FragmentYP.setTabListener(new TabListener<FragmentYP>(this,
"FragmentCompanies", FragmentYP.class));
 actionBar.addTab(FragmentYP);

Tab FragmentWP = actionBar.newTab();
FragmentWP.setText("People");
FragmentWP.setTabListener(new TabListener<FragmentWP>(this, "Fragment
People", FragmentWP.class));
actionBar.addTab(FragmentWP);

Tab FragmentMap  = actionBar.newTab();
FragmentMap .setText("Important Numbers");
FragmentMap .setTabListener(new TabListener<FragmentImpNumbers>(this,
"FragmentImpNumbers",FragmentImpNumbers.class));
actionBar.addTab(FragmentMap );

if (savedInstanceState != null) {
int savedIndex = savedInstanceState.getInt("SAVED_INDEX");
getActionBar().setSelectedNavigationItem(savedIndex);
}
```

6.2 Design Method

The ‘Companies’ tab of ‘NepGuide Mobile’ offers two types of search methods to the users. The users can search for business details either by name or by category. Once the user fills all the required fields, the application fetches query relevant information on the company’s database and displays the search result on the ListDisplayActivity screen. The relevant search data from from the company’s database are parsed by JSONParser in the Android System. The code below explains how the data are fetched from the server using JSON(Appendix 1.1 FragmentCompany.java)

JSON Object is used to retrieve all the categories from the company’s database to the drop down menu(spinner)

```
JSONObject jsonResponse =
jParser.getJSONFromUrl("http://nepguide.com/api/index.php/getcat/");
catList = jsonResponse.getJSONArray("data");
```

URL link to company’s php files based on the search type

```
url = "http:nepguide.com/api/index.php/search/yellow/name/"+keyword+"/"+city+"/";
url = "http://nepguide.com/api/index.php/search/yellow/cat/"+category+"/"+city+"/";
```

Parsing the information received from the URL and assigning as variable in Android system.

```
JSONObject json = jParser.getJSONFromUrl(url);

final String name=json_data.getString("Name");
final String services=json_data.getString("Services");
final String compid=json_data.getString("CompId");
final String info=json_data.getString("OtherInfo");
final String contact=json_data.getString("Contact");
final String address=json_data.getString("Address");
allItems.put(compid, name+"\n"+address+"\n"+contact+"\n"+services+"\n"+info);
```

The search results are pointed to ListDispalyActivity screen to display as search results.

```
Intent i = new Intent();
i.putExtra("items", allItems);
i.setClass(getActivity(), ListDisplayActivity.class);
```

From the search result screen(ListDisplayActivity), the user can click on one of the search results to see the detailed information of the respective business. The code below corresponds to redirection form search result activity to specific business profile(Appendix 1.1 ListDisplayActivity.java).

```
Intent i = new Intent();
i.putExtra("items", compid);
i.setClass(ListDisplayActivity.this, CompanyDetailActivity.class);
```

7. APPLICATION DEVELOPMENT

An Android application consist of many components, which are picked and used as per the requirement of the application. Many components are implemented logically to enrich the user experience in ‘NepGuide Mobile’. Some of the components used in ‘NepGuide Mobile’ are layout, menu, fragments, text fields, spinner and buttons.

7.1 User Interface

The ‘NepGuide Mobile’s user interface is designed in such a way user friendly way. Different activities hold different UI components to serve the intended purpose. The home screen (HomePageActivity) consists of three fragments (tabs), an action bar (menu bar) and few menu items. The visual structure of an application can be accessed in several layout views namely linear layout, relative layout and list view. User interaction with the application can be accessed by using a variety of control and input methods. The control method includes buttons and spinner whereas the input method includes mainly text fields. A user also can navigate through different activities using menu items, which are common on different fragments within this application. In situations where an error message or notification has to be displayed, ‘NepGuide Mobile’ manifests these messages through the AlertDialog component.

The NepGuide Mobile’s ‘companies’ fragment is used to search for companies in a certain location(district or city) either by name or category. The result of the search query is displayed in list view in the ListDisplay Activity, furthermore, if one of the lists in ListDisplayActivity is clicked, the respective company’s full profile is displayed in CompanyDetailActivity.

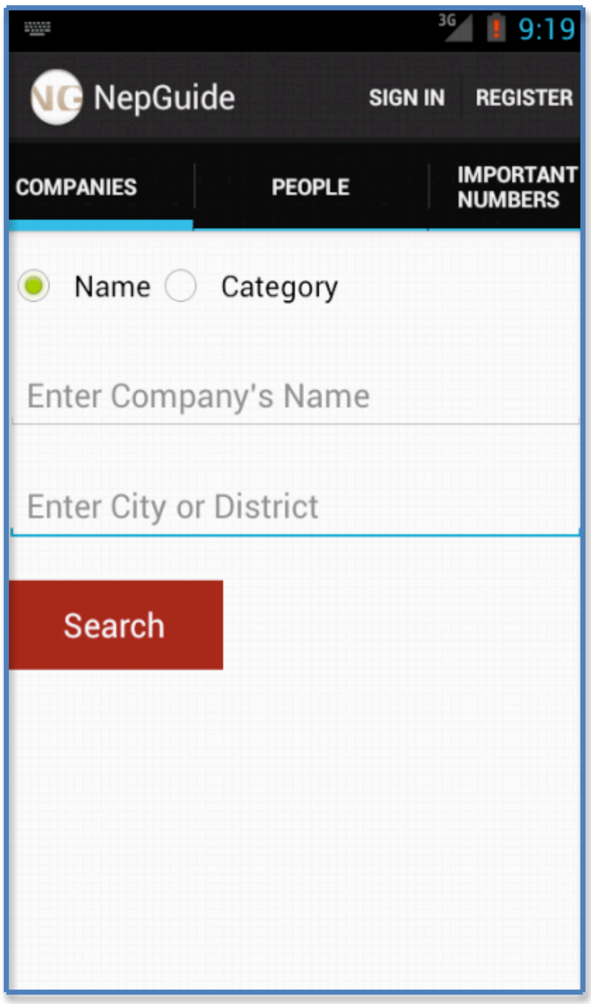


Figure 14. NepGuide Mobile UI

7.2 Database

NepGuide Pvt. Ltd. has its own MySQL database where companies are systematically registered based on their category and location. There are three PHP files in the server which contain all the SQL queries and PHP variables required to fetch data from the database server. These PHP variables are parsed by JSON to JSON objects . The PHP files used in this application are provided by NepGuide Pvt. Ltd.

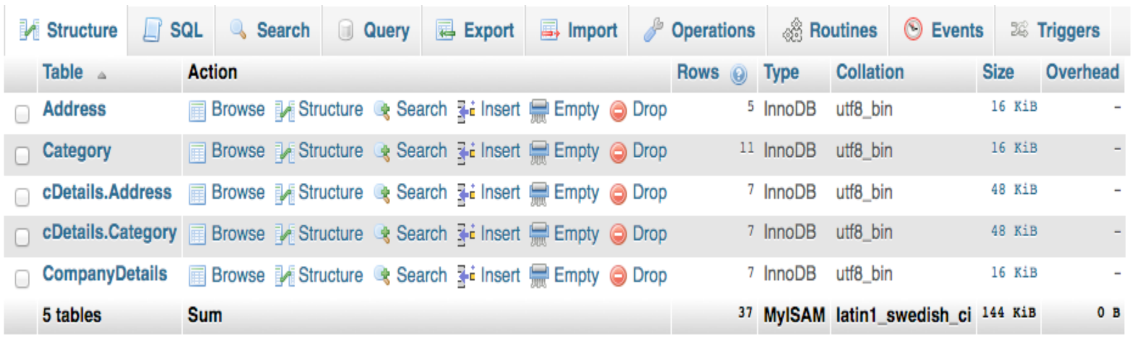


Table	Action	Rows	Type	Collation	Size	Overhead
Address	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8_bin	16 KiB	-
Category	Browse Structure Search Insert Empty Drop	11	InnoDB	utf8_bin	16 KiB	-
cDetails.Address	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8_bin	48 KiB	-
cDetails.Category	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8_bin	48 KiB	-
CompanyDetails	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8_bin	16 KiB	-
5 tables	Sum	37	MyISAM	latin1_swedish_ci	144 KiB	0 B

Figure 15. Company’s Database

8. APPLICATION TEST

After writing all the source code and fixing all the major bugs(some of the bug fixing in this project required some professional help), ‘NepGuide Mobile’ was ready to be tested. During the development process, each and every step of code writing process and output of the code was tested on Android virtual device. After testing the application successfully on the virtual device, the application was then tested on the Android mobile device.

The aim of the thesis was to achieve the companies tab working as intended. When the application is launched it first starts the Home Screen Activity, which has all the tabs and menu item to access different activities. On the companies tab, the user has the option to search the companies either by name or category in some particular city or district of Nepal. In the categorical search, the user can choose the category for the list of categories provided in the spinner.

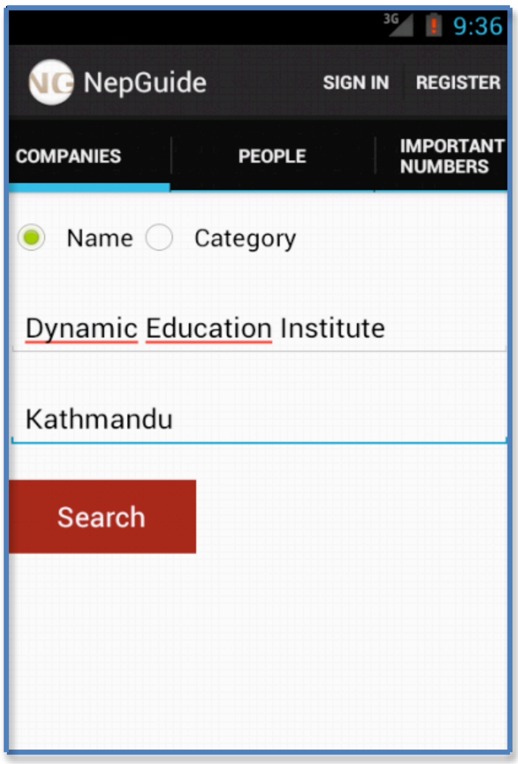


Figure 16. Search Company by Name

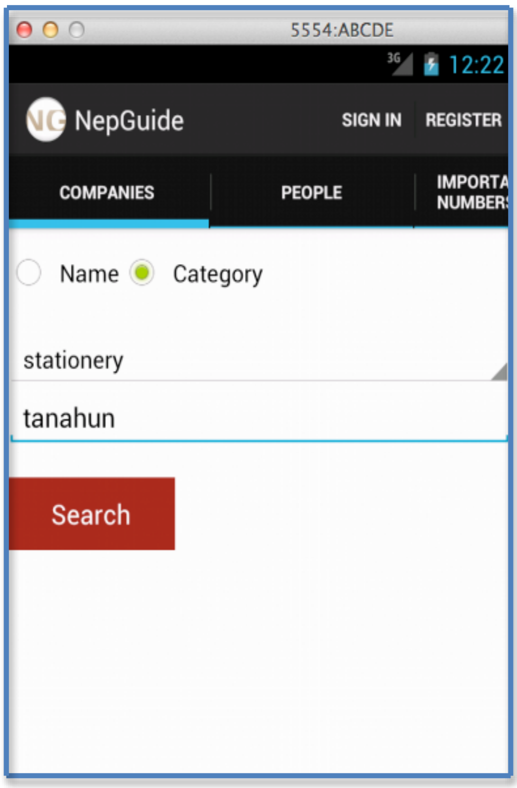
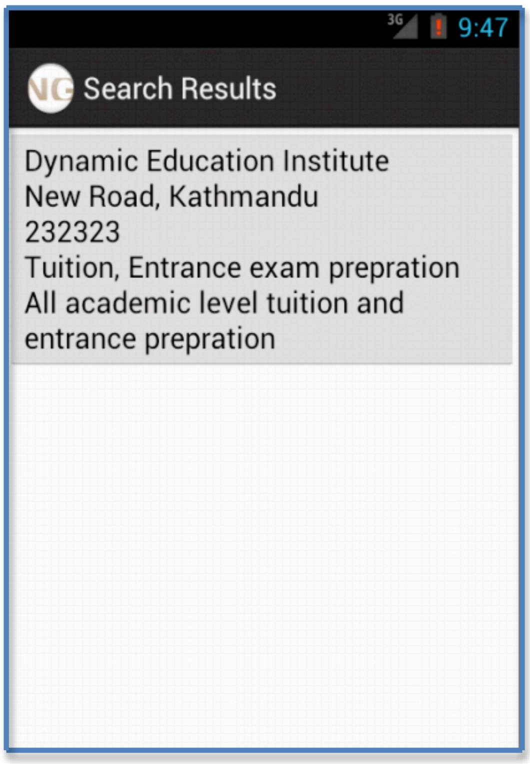


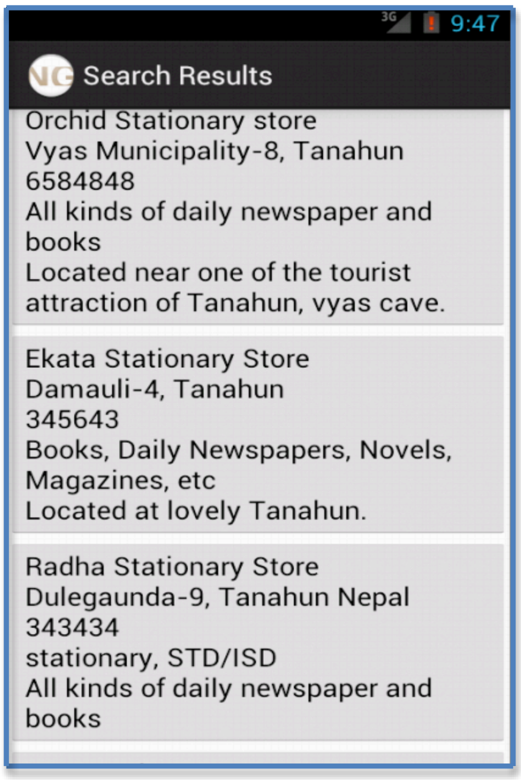
Figure 17. Search Companies by Category

Once the search button is clicked with right search input, the search results are displayed as a list in ListDisplayActivity. If there are many search results, the user has the option to scroll through the list of results and choose the desired company.



Dynamic Education Institute
New Road, Kathmandu
232323
Tuition, Entrance exam prepration
All academic level tuition and entrance prepration

Figure 18. Search Result by Name



Orchid Stationary store
Vyas Municipality-8, Tanahun
6584848
All kinds of daily newspaper and books
Located near one of the tourist attraction of Tanahun, vyas cave.

Ekata Stationary Store
Damauli-4, Tanahun
345643
Books, Daily Newspapers, Novels, Magazines, etc
Located at lovely Tanahun.

Radha Stationary Store
Dulegaunda-9, Tanahun Nepal
343434
stationary, STD/ISD
All kinds of daily newspaper and books

Figure 19. Search Results by Category

A user can click(tap) on one of the item from list of search result to view the full profile of the respective company in DetaileDisplayActivity.

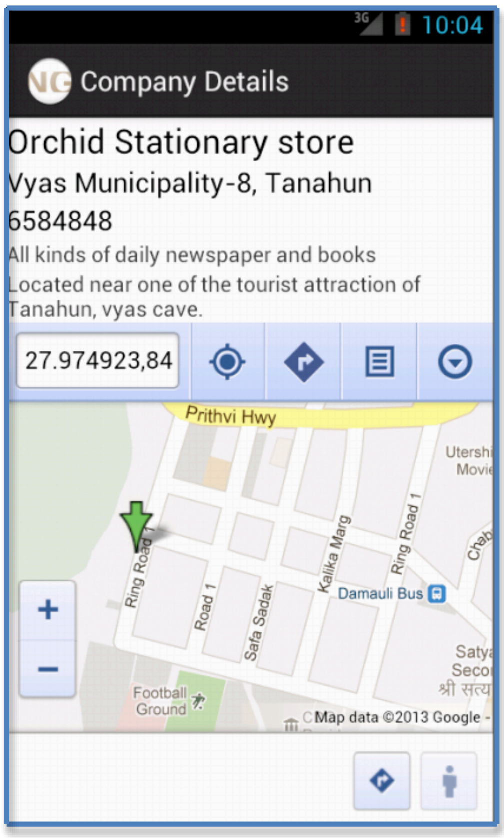


Figure 20 . Company's Full Profile

The application test result meets all the requirements of this project. NepGuide Pvt. Ltd. was satisfied with the method of filtering categorized information from the server and the process of logically displaying search results on Android devices.

9 CONCLUSION

The goal of this thesis was to develop Android mobile application for NepGuide Pvt. Ltd., which could interact with the company's backend server. These goals have been successfully achieved on completion of this project. The project was tested on both Android emulator and Android mobile device. The application ran smoothly and the UI components responded as expected.

This thesis projects some of the fundamentals on Android database application, but more services and features can be added. One of the key online services NepGuide Pvt. Ltd. offers has been developed and implemented successfully through this application. Although this application is developed for NepGuide Pvt. Ltd, the guidelines provided on this thesis could be beneficial to students who are new to Android application development. This thesis provides general information and guidelines for application developers on Android Operating System, developer tools and application development process. Since Android is an open source OS, an abundance of resources, developer tools, guidelines and documentation can be found in Android open source community forums for free. http://developer.android.com is one of the very important website that provides all the information, guidelines, tools, and resources necessary to develop an Android app.

REFERENCES

Gargenta, M. 2011, Learning Android. Retrieved May 20, 2013

Gartner. Retrieved May 20, 2013 from
http://www.gartner.com/newsroom/id/2482816

Services. Retrieved April 25, 2013 from
http://developer.android.com/guide/components/services.html.

JSON Tutorial. Retrieved May 4, 2013 from
http://www.w3schools.com/json/

Using the emulator. Retrieved May 10, 2013 from
http://developer.android.com/tools/devices/emulator.html.

Get the Android SDK
http://developer.android.com/sdk/index.html

Building and Running. Retrieved May 15, 2013 from
http://developer.android.com/tools/building/index.html.

The AndroidManifest.xml File. Retrieved May 21, 2013
http://developer.android.com/guide/topics/manifest/manifest-intro.html.

APPENDICES

Source code download link: http://nepguide.com/apps/sourcecode

(All the files available except database and PHP files)

Application download link: http://nepguide.com/apps/download

APPENDIX 1.0 HomePageActivity.java

```
package com.nepguide;

import android.app.ActionBar;
import android.app.ActionBar.Tab;
import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.os.Bundle;
import android.os.StrictMode;
import android.view.Menu;
import android.view.MenuItem;

public class HomePageActivity extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
StrictMode.setThreadPolicy(policy);
super.onCreate(savedInstanceState);

final ActionBar actionBar = getActionBar();
actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);

Tab FragmentYP = actionBar.newTab();
FragmentYP.setText("Companies");

FragmentYP.setTabListener(new TabListener<FragmentCompany>(this, "Companies",
FragmentCompany.class));
actionBar.addTab(FragmentYP);

Tab FragmentWP = actionBar.newTab();
FragmentWP.setText("People");
FragmentWP.setTabListener(new TabListener<FragmentPeople>(this, "People",
FragmentPeople.class));
actionBar.addTab(FragmentWP);

Tab FragmentMap  = actionBar.newTab();
FragmentMap .setText("Important Numbers");
FragmentMap .setTabListener(new TabListener<FragmentImpNumbers>(this,
"Important Numbers",FragmentImpNumbers.class));
actionBar.addTab(FragmentMap );

if (savedInstanceState != null) {
int savedIndex = savedInstanceState.getInt("SAVED_INDEX");
getActionBar().setSelectedNavigationItem(savedIndex);
}
}

@Override
protected void onSaveInstanceState(Bundle outState) {
//TODO Auto-generated method stub
super.onSaveInstanceState(outState);
outState.putInt("SAVED_INDEX", getActionBar().getSelectedNavigationIndex());
}

public static class TabListener<T extends Fragment>  implements
ActionBar.TabListener{

private final Activity myActivity;
private final String myTag;
private final Class<T> myClass;

public TabListener(Activity activity, String tag, Class<T> cls) {
myActivity = activity;
myTag = tag;
myClass = cls;
}

@Override
public void onTabSelected(Tab tab, FragmentTransaction ft) {

Fragment myFragment =
myActivity.getFragmentManager().findFragmentByTag(myTag);

//Check if the fragment is already initialized
if (myFragment == null) {
//If not, instantiate and add it to the activity
myFragment = Fragment.instantiate(myActivity, myClass.getName());
ft.add(android.R.id.content, myFragment, myTag);
}
else {
//If it exists, simply attach it in order to show it
ft.attach(myFragment);
}

}

@Override
public void onTabUnselected(Tab tab, FragmentTransaction ft) {

Fragment myFragment =
myActivity.getFragmentManager().findFragmentByTag(myTag);
```

```
if (myFragment != null) {
//Detach the fragment, because another one is being attached
ft.detach(myFragment);
}
}

@Override
public void onTabReselected(Tab tab, FragmentTransaction ft) {
//TODO Auto-generated method stub

}

}

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

@Override
public boolean onOptionsItemSelected(MenuItem item){
super.onOptionsItemSelected(item);
switch(item.getItemId()){
case R.id.SignIn:
startActivity(new Intent(this, SignInActivity.class));;
break;

case R.id.Register:
startActivity(new Intent(this, RegisterActivity.class));;
break;
//default:return super.onOptionsItemSelected(item);
}
return true;

}
}
```

APPENDIX 1.1 FragmentCompany.java

```
package com.nepguide;


import java.util.ArrayList;
import java.util.HashMap;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.json.JSONStringer;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.app.Fragment;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup;
import android.webkit.WebView.FindListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;

public class FragmentCompany extends Fragment{

public View onCreateView(LayoutInflater inflater, ViewGroup container,Bundle
savedInstanceState) {
final View view = inflater.inflate(R.layout.fragment_company, container, false);
Spinner catSpinner = (Spinner) view.findViewById(R.id.ypcatspinner);
JSONParser jParser = new JSONParser();
JSONArray catList = null;
try{
JSONObject jsonResponse =
jParser.getJSONFromUrl("http://nepguide.com/api/index.php/getcat/");
catList = jsonResponse.getJSONArray("data");
```

```
}
catch(JSONException e){
alertbox("Internet connection not found");
e.printStackTrace();
}
if(catList != null){
ArrayAdapter<String> catA = new ArrayAdapter<String>(getActivity(),
android.R.layout.simple_spinner_item);
catA.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
catA.add("Select a Category");
JSONObject out_data;
for (int i = 0; i < catList.length(); ++i)
{
String CatName = null;
//String CatId = null;
try {
out_data  = catList.getJSONObject(i);
//Log.v("FragmentCompany",out_data.toString());
//CatId = out_data.getString("id");
CatName = out_data.getString("name");
} catch (JSONException e) {
Log.v("FragmentCompany","Category parsing failed");
e.printStackTrace();
}
if(CatName != null){
catA.add(CatName);
}
}
catSpinner.setAdapter(catA);
catSpinner.setVisibility(View.GONE);
}
RadioGroup radioGroup = (RadioGroup) view.findViewById(R.id.ypsearchtype);
radioGroup.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
public void onCheckedChanged(RadioGroup group, int checkedId) {
```

```
View p = (View)group.getParent();
RadioButton btn = (RadioButton)p.findViewById(checkedId);
Spinner SpinnerCat = (Spinner) p.findViewById(R.id.ypcatspinner);
EditText EditTextKeyword = (EditText) p.findViewById(R.id.ypkeyword);
String selType = btn.getText().toString();
if(selType.equals("Category")){
EditTextKeyword.setVisibility(View.GONE);
SpinnerCat.setVisibility(View.VISIBLE);
}else{
EditTextKeyword.setVisibility(View.VISIBLE);
SpinnerCat.setVisibility(View.GONE);
}
}
});
Button YpSearchBttn = (Button) view.findViewById(R.id.ypsearch);
YpSearchBttn.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
JSONParser jParser = new JSONParser();
View p = (View)v.getParent();
RadioButton ypname = (RadioButton) p.findViewById(R.id.ypname);
EditText cityText = (EditText) p.findViewById(R.id.ypaddress);
String city = cityText.getText().toString();
String url = "";
if(ypname.isChecked()){
EditText key = (EditText) p.findViewById(R.id.ypkeyword);
String keyword = key.getText().toString();
if(keyword.length() == 0  || city.length() == 0){
alertbox("All fields are required");
}
else{
url = "http:nepguide.com/api/index.php/search/yellow/name/"+keyword+"/"+city+"/";
}
}else{
Spinner cat = (Spinner) p.findViewById(R.id.ypcatspinner);
```

```
String category = cat.getSelectedItem().toString();
if(category.equals("Select one...") || city.length() == 0){
alertbox("Please choose category and input address");
}
else{
url = "http://nepguide.com/api/index.php/search/yellow/cat/"+category+"/"+city+"/";
}
}
if(url != ""){
Log.v("FragmentCompany","Fetching url: "+url);
JSONObject json = jParser.getJSONFromUrl(url);
final String TAG_DATA = "data";
final String TAG_ERROR = "error";
JSONArray error;
String sError;
ArrayList<HashMap<String, String>> items;
try {
error = json.getJSONArray(TAG_ERROR);
}
catch (JSONException e) {
error = null;
}
if(error == null){
try {
sError = json.getString(TAG_ERROR);
}
catch (JSONException g) {
sError = "Not valid JSON";
}
if(sError.length()==0){
HashMap<String, String> allItems = new HashMap<String, String>();
try{
JSONArray data = json.getJSONArray(TAG_DATA);
JSONObject json_data;
for(int i=0; i < data.length() ; i++) {
```

```
json_data = data.getJSONObject(i);

final String name=json_data.getString("Name");
final String services=json_data.getString("Services");
final String compid=json_data.getString("CompId");
final String info=json_data.getString("OtherInfo");
final String contact=json_data.getString("Contact");
final String address=json_data.getString("Address");
allItems.put(compid, name+"\n"+address+"\n"+contact+"\n"+services+"\n"+info);
}
}
catch(JSONException f) {
Log.v("FragmentCompany","Data cannot be parsed");
items = null;
f.printStackTrace();
}
if(allItems.size() > 0){
Intent i = new Intent();
i.putExtra("items", allItems);
i.setClass(getActivity(), ListDisplayActivity.class);
getActivity().startActivity(i);
}
else{
alertbox("Sorry! No result found");
}
}else{
alertbox(sError);
}
}
else{
alertbox(error.toString());
}
}
}
});
```

```
return view;
}
protected void alertbox(String mymessage){
AlertDialog.Builder show = new AlertDialog.Builder(getActivity());
show.setMessage(mymessage);
show.setTitle("Alert");
AlertDialog dialog = show.create();
dialog.show();
}
}
```

APPENDIX 1.2 ListDisplayActivity.java

```
package com.nepguide;
import java.util.HashMap;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TableRow.LayoutParams;

public class ListDisplayActivity extends Activity {
@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.list_display);
HashMap<String, String> items = (HashMap<String,
String>)getIntent().getSerializableExtra("items");
LinearLayout l = (LinearLayout) findViewById(R.id.ListDisplayLayout);
LayoutParams params = new
LayoutParams(LayoutParams.MATCH_PARENT,LayoutParams.WRAP_CONTENT);
for(String key : items.keySet()) {
final Button item = new Button(this);
item.setText(items.get(key));
item.setTag(Integer.parseInt(key));
Log.v("FragmentYP",items.get(key));
item.setLayoutParams(params);
item.setGravity(Gravity.LEFT);
item.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
View p = (View)v.getParent();
String compid = ((Button)v).getTag().toString();
//item.setBackgroundColor(Color.RED);
//Log.v("FragmentYP","Compid: "+compid);
Intent i = new Intent();
i.putExtra("items", compid);
i.setClass(ListDisplayActivity.this, CompanyDetailActivity.class);
ListDisplayActivity.this.startActivity(i);
}
});
l.addView(item);
}
}
}
```

APPENDIX 1.3 CompanyDetailActivity.java

```
package com.nepguide;
```

```
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.webkit.WebView;
import android.widget.TextView;

public class CompanyDetailActivity extends Activity {

@Override
public void onCreate(Bundle savedInstanceState)
{
super.onCreate(savedInstanceState);
setContentView(R.layout.company_details);
String items = (String)getIntent().getSerializableExtra("items");
//Log.v("FragmentYP",items);
JSONParser jParser = new JSONParser();
JSONArray comp = null;
try{
JSONObject jsonResponse =
jParser.getJSONFromUrl("http://binaydevkota.com/nepguide/api/index.php/getcomp/"+i
tems);
JSONObject error = null;
try{
error = jsonResponse.getJSONObject("error");
alertbox(error.toString());
}
catch(JSONException e1){
e1.printStackTrace();
}
if(error == null){
comp = jsonResponse.getJSONArray("data");
JSONObject company = (JSONObject) comp.get(0);
TextView compname = (TextView) findViewById(R.id.compname);
compname.setText(company.getString("Name").toString());
TextView compaddress = (TextView) findViewById(R.id.compaddress);
compaddress.setText(company.getString("Address").toString());
TextView compcontact = (TextView) findViewById(R.id.compcontact);
compcontact.setText(company.getString("Contact").toString());
TextView compservices = (TextView) findViewById(R.id.compservices);
compservices.setText(company.getString("Services").toString());
TextView compotherinfo = (TextView) findViewById(R.id.compotherinfo);
compotherinfo.setText(company.getString("OtherInfo").toString());
WebView compmap = (WebView) findViewById(R.id.compmap);
compmap.getSettings().setJavaScriptEnabled(true);
compmap.loadUrl("https://maps.google.com/maps?z=13&q="+company.getString("Latit
ude").toString()+","+company.getString("Longitude").toString());
}
}
catch(JSONException e){
alertbox("Cound not reach to the internet for company details");
e.printStackTrace();
}
if(comp != null){

}
}
protected void alertbox(String mymessage){
AlertDialog.Builder show = new AlertDialog.Builder(CompanyDetailActivity.this);
show.setMessage(mymessage);
show.setTitle("Alert");
AlertDialog dialog = show.create();
dialog.show();
}
}
```

APPENDIX 1.4 JSONParser.java

```
package com.nepguide;
import java.io.BufferedReader;
import java.io.Console;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URI;
import java.net.URISyntaxException;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONException;
import org.json.JSONObject;
import android.app.AlertDialog;
import android.util.Log;

public class JSONParser {
```

```
static InputStream is = null;
static JSONObject jObj = null;
static String json = "";
String TAG = "FragmentYP";
//constructor
public JSONParser() {
}
public JSONObject getJSONFromUrl(String url) {
//Making HTTP request
URI uri = null;
try {
uri = new URI(url.replace(" ", "%20"));
} catch (URISyntaxException e1) {
//TODO Auto-generated catch block
e1.printStackTrace();
}
try {
//defaultHttpClient
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpGet httpPost = new HttpGet(uri);
HttpResponse httpResponse = httpClient.execute(httpPost);
HttpEntity httpEntity = httpResponse.getEntity();
is = httpEntity.getContent();
} catch (UnsupportedEncodingException e) {
e.printStackTrace();
} catch (ClientProtocolException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
try {
BufferedReader reader = new BufferedReader(new InputStreamReader(
is, "utf-8"), 8);
StringBuilder sb = new StringBuilder();
String line = null;
while ((line = reader.readLine()) != null) {
sb.append(line + "\n");
}
is.close();
json = sb.toString();
//Log.v(TAG, json);
} catch (Exception e) {
Log.e("Buffer Error", "Error converting result " + e.toString());
}
//try parse the string to a JSON object
try {
jObj = new JSONObject(json);
} catch (JSONException e) {
Log.e("JSON Parser", "Error parsing data " + e.toString());
}
//return JSON String
return jObj;
}
}
```