2011

Indian Institute of
Information
Technology, Allahabad

Jivesh Singh Gahlawat
Sourabh Gupta
Deepak Bhoria

Under the supervision of
Dr. K. P. Singh

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY – ALLAHABAD
DEOGHAT, JHALWA
ALLAHABAD- 211012, (U.P.)
INDIA

[EMERGENCY GPS SERVICE]

A mini project on android application which reports accidents using GPS with exact location

Emergency GPS Service

by

Jivesh Singh Gahlawat

(IIT2009008)

iit2009008@iiita.ac.in

Sourabh Gupta

(IIT2009066)

iit2009066@iiita.ac.in

Deepak Bhoria

(IIT2009112)

iit2009112@iiita.ac.in

Report submitted to
Indian Institute of Information Technology – Allahabad

Table of Contents

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD
(Deemed University)

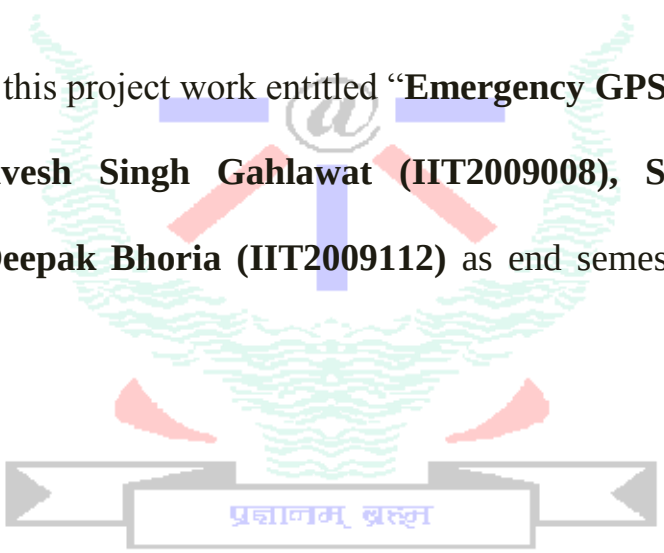
(A centre of excellence in IT, established by Ministry of HRD, Govt. of India)

Date: _____

CERTIFICATE OF ACCOMPLISHMENT

This is to certify that this project work entitled "**Emergency GPS Service**" which is submitted by **Jivesh Singh Gahlawat (IIT2009008), Sourabh Gupta (IIT2009066)** and **Deepak Bhoria (IIT2009112)** as end semester mini project report.

COUNTERSIGNED



_____

Dr. K. P. Singh
(Project Supervisor)

COMMITTEE, ON FINAL EVALUATION OF THE PROJECT:

_____

_____

_____

ACKNOWLEDGEMENT

As understanding of the study like this is never the outcome of the efforts of a single person, rather it bears the imprint of a number of persons who directly or indirectly helped us in completing the present study. We would be failing in our duty if we don't say a word of thanks to all those whose sincere advise make our this documentation of topic a real educative, effective and pleasurable one.

It is our privilege to study at Indian Institute of Information Technology, Allahabad where students and professors are always eager to learn new things and to make continuous improvements by providing innovative solutions. We are highly grateful to the honorable **Dr. M. D. Tiwari, Director IIIT-Allahabad,** for his ever helping attitude and encouraging us to excel in studies.

Regarding this thesis work, first and foremost, we would like to heartily thank our supervisor **Dr. K.P. Singh** for his able guidance. His fruitful suggestions, valuable comments and support were an immense help for me. In spite of his hectic schedule he took pains, with smile, in various discussions which enriched us with new enthusiasm and vigor.

ABSTRACT

In this project we are trying to build an Emergency GPS Messaging application for android mobile which will send the exact location of user to Police, Ambulance, Fire Brigade or other known to him in case of any emergency.

Sometimes user may not be aware of his location and mishap can occur. In such a situation user requires a method to send his accurate location and other details to get help. Our App calculates his correct position using GPS and sends an alert message to the people concerned with the exact location of accident.

It will locate the user with the help of GPS navigation. After exact location of user is known, any possible help can be provided by Police, Ambulance, Fire Brigade or others. So it can be of great help in case of emergency situations.

INTRODUCTION

Topic

Emergency GPS Service is an Android Mobile Application for report emergency quickly and accurately (in terms of place).

Problem Definition

Emergency GPS Service is an Android Mobile Application which will be using Google's Map API. Google Maps (formerly Google Local) is a web mapping service application and technology provided by Google, free (for non-commercial use), that powers many map-based services. It offers street maps and an urban business locator for numerous countries around the world.

Our application will use this feature to report the location of user to the concerned authority like ambulance, police or any other help.

Goal

The goal of our application is to report all the accidents and mishaps as soon as possible with the exact location of the place, which is not possible in case of dialing and telling as the place may be unknown to user.

Motivation

We were motivated to do such project because when we picked up newspaper any day, we would surely see any story about a fire accident or road accident and that there were several casualties just because the appropriate help could not reach in time.

Scope

The scope of our application is huge in the present scenario as we are making this app for android and we could see that this is the era of smartphones and android

being open source and the most feature full operating system for smartphones, and is also currently the largest selling smartphone os in market, makes our application of great use.

Proposed Functionalities

Below is a list of proposed functionalities in our app:

- Creating a connection over Wi-Fi and getting user's current location through Google Maps API.
- Sending an SMS to any number through application, including emergency numbers.
- Creating a database of emergency numbers of all countries using sqlite.
- Optimizing battery performance through good workflow.
- Using advanced features like autocomplete and menu scrolling in relative linear layout of our application.
- Spotting the exact user location on map on user side.
- A client side app which would be used for retrieving received user coordinates and showing location on map.

Advantages over usual emergency calling

- The application runs on android, which is the most widely used smartphone operating system (OS) in market.
- The exact location of the accident is known to the authority for quick action, even if user is not aware of his current location.

Negligible Drawbacks

- For location retrieval through GPS, the application needs thorough access to internet, which may not be available somewhere. But today is the 3G world and everyone has access to internet, so this may be neglected.

System Requirements and Software Used

The system and software requirements for developing Android applications using the Android SDK.

Supported Operating Systems

- Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu Linux, Lucid Lynx)
    - GNU C Library (glibc) 2.7 or later is required.
    - On Ubuntu Linux, version 8.04 or later is required.
    - 64-bit distributions must be capable of running 32-bit applications.

Supported Development Environments

Eclipse IDE

- Eclipse 3.4 (Ganymede) or greater
- Eclipse JDT plugin (included in most Eclipse IDE packages)
- If you need to install or update Eclipse, you can download it from http://www.eclipse.org/downloads/.

Several types of Eclipse packages are available for each platform. For developing Android applications, we recommend that you install one of these packages:

- Eclipse IDE for Java Developers
- Eclipse Classic (versions 3.5.1 and higher)
- Eclipse IDE for Java EE Developers
- JDK 5 or JDK 6 (JRE alone is not sufficient)
- Android Development Tools plugin (recommended)

Other development environments or IDEs

- JDK 5 or JDK 6 (JRE alone is not sufficient)
- Apache Ant 1.8 or later
- **Not** compatible with Gnu Compiler for Java (gcj)

Hardware requirements

The Android SDK requires disk storage for all of the components that you choose to install. The table below provides a rough idea of the disk-space requirements to expect, based on the components that you plan to use.

Component type	Approximate size	Comments
SDK Tools	35 MB	Required.
SDK Platform-tools	6 MB	Required.
Android platform (each)	150 MB	At least one platform is required.
SDK Add-on (each)	100 MB	Optional.
USB Driver for Windows	10 MB	Optional. For Windows only.
Samples (per platform)	10M	Optional.
Offline documentation	250 MB	Optional.

Note that the disk-space requirements above are *in addition to* those of the Eclipse IDE, JDK, or other prerequisite tools that you may need to install on your development computer.

PROJECT DEVELOPMENTS

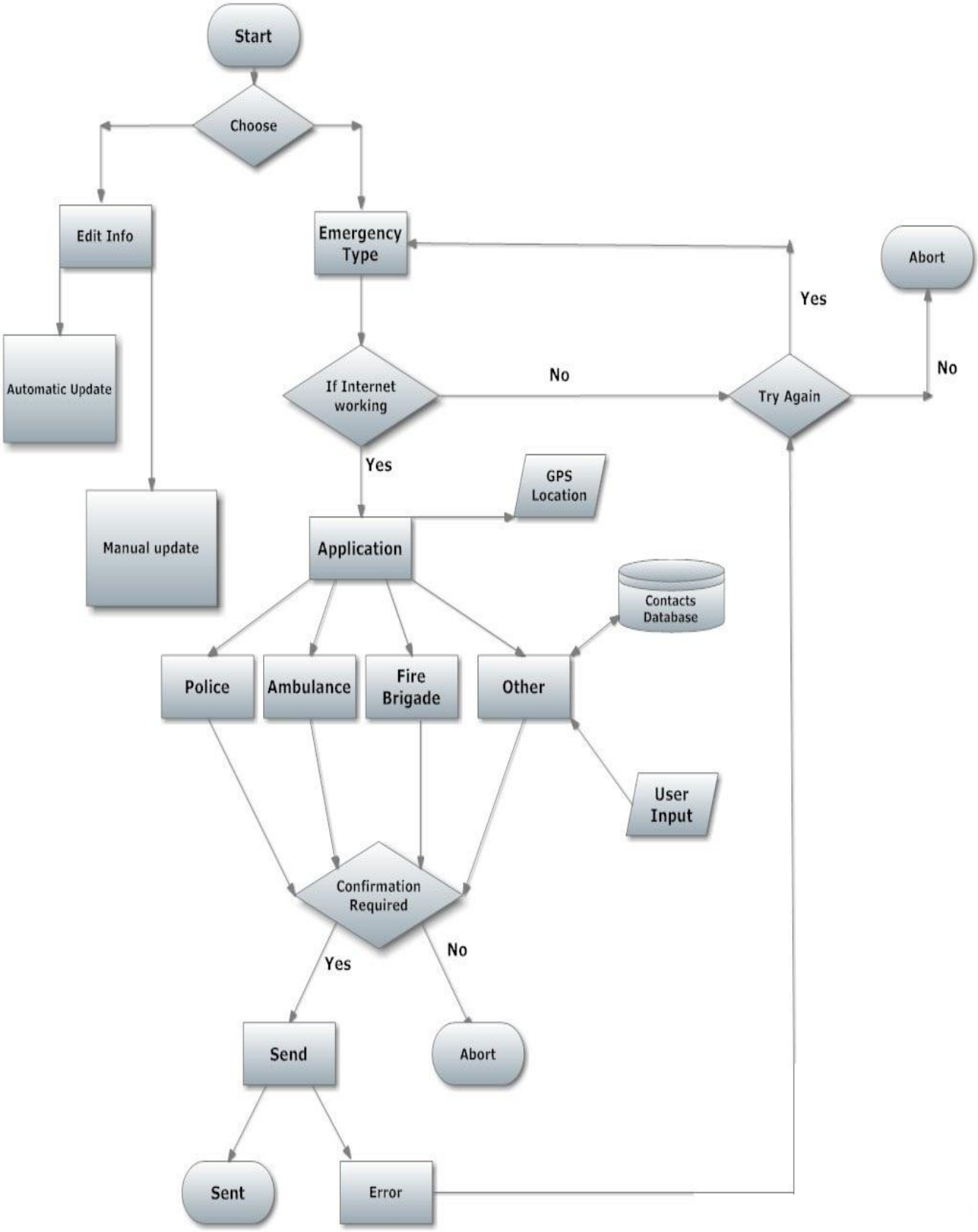
(In chronological order)

- ✓ Installation of Android SDK with needed packages, eclipse plugins, Google API and Android Virtual Device.
- ✓ Learn our way through Android development by making simple applications.
- ✓ Familiarization with Google APIs' like Location Manager and their usage.
- ✓ Discussed and made complete layout of the GPS Emergency Service Application including work flow diagram.
- ✓ Designed the complete Graphical User Interface.
- ✓ Created a database backend using SQLite.
- ✓ Retrieve location of user using Google APIs'.
- ✓ Send SMS message to the destination.
- ✓ Store contacts in a list which could be searched and scrolled.
- ✓ Made a database of already existing emergency numbers of various countries.
- ✓ Locate the user using latitude and longitude coordinated which is more accurate than addresses.
- ✓ Our application saves battery life as we have optimized and restricted the need of internet to only when the location is retrieved by calling retrieve activity in the end.

PLAN OF ACTION

July	Introduction about android device and their components
August	Learning about Android API's and SDK, Java, and Eclipse IDE.
September	Developed and tested sample Android Applications to brush up android skill: 1. Flashlight Application 2. Task manager Application 3. Made GUI for our App.
October	Developed the 'Emergency GPS' application . We divided our work into three different phases: Phase I - Making a database of emergency contacts in sqlite. Phase II – Making location finder for our app using google api's. Phase III – Sending SMS to destination and retrieving location at receiver end using client side app.
November	Testing and Debugging includes testing of validations and error exceptions.

.



WORKFLOW DIAGRAM

LITERATURE REVIEW

Introduction to android and application development

What is Android?

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. It is developed by the Open Handset Alliance led by Google. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Google purchased the initial developer of the software, Android Inc., in 2005. The unveiling of the Android distribution on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 84 hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software license. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android.

Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Android uses the Dalvik virtual machine with just-in-time compilation to run compiled Java code. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. Developers write primarily in a customized version of Java. There are currently more than 250,000 apps available for Android. Apps can be downloaded from third-party sites or through online stores such as Android Market, the app store run by Google.

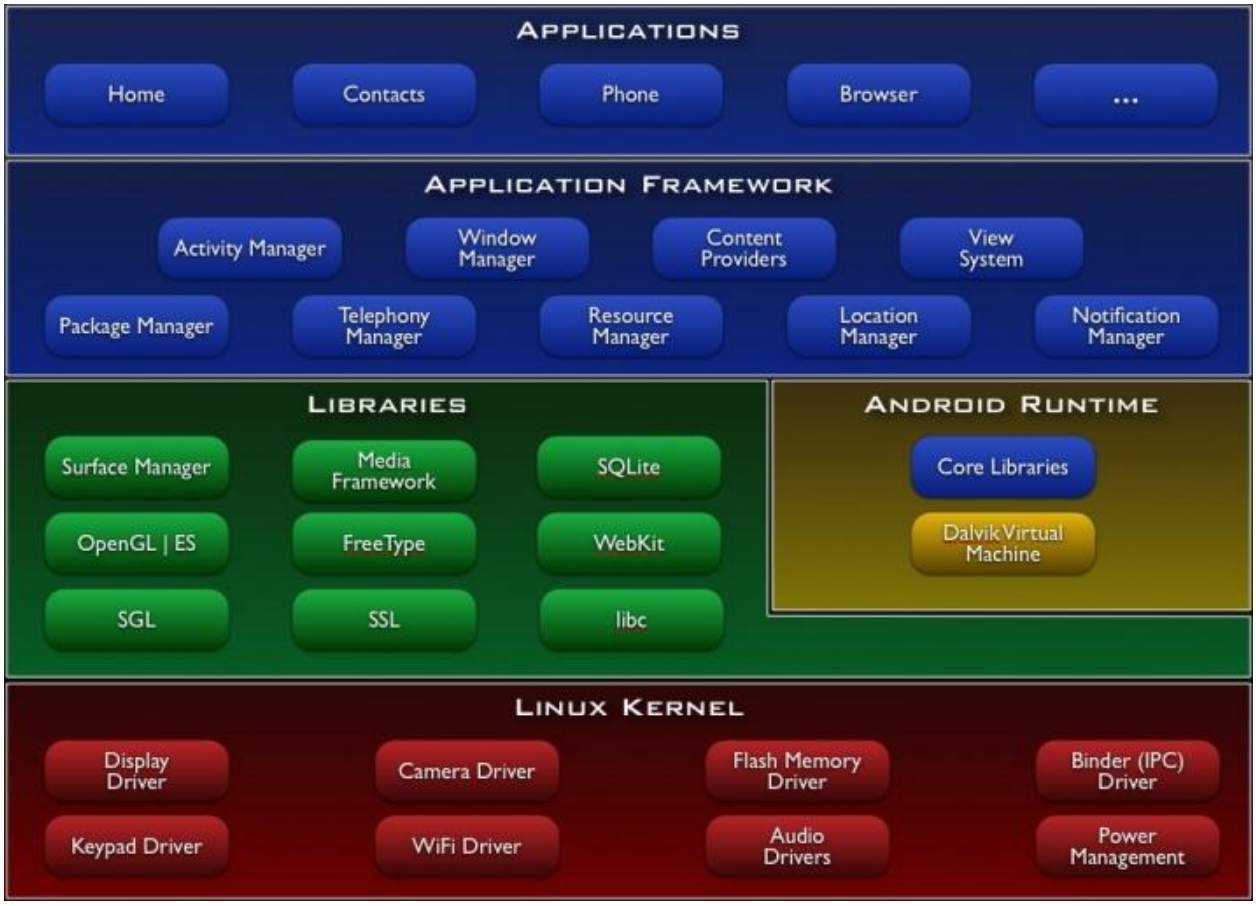
Features

Application framework: enabling reuse and replacement of components
Dalvik virtual machine: optimized for mobile devices
Integrated browser: based on the open source WebKit engine

Optimized graphics: powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
SQLite: for structured data storage
Media support: for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
GSM Telephony: (hardware dependent)
Bluetooth, EDGE, 3G, and WiFi: (hardware dependent)
Camera, GPS, compass, and accelerometer: (hardware dependent)
Rich development environment: including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

Android Architecture

The following diagram shows the major components of the Android operating system. Each section is described in more detail below.



Applications

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

Views: A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.

Content Providers: Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.

Resource Manager: A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.

Notification Manager: A Notification Manager that enables all applications to display custom alerts in the status bar.

Activity Manager: An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack.

Libraries:

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

System C library: A BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.
Media Libraries: based on Packet Video's Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.
Surface Manager: manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
LibWebCore: a modern web browser engine which powers both the Android browser and an embeddable web view.
SGL: the underlying 2D graphics engine.
3D libraries: an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer.
FreeType: bitmap and vector font rendering.
SQLite: a powerful and lightweight relational database engine available to all applications.

Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Application Fundamentals

Android applications are written in the Java programming language. The Android SDK tools compile the code—along with any data and resource files—into an Android package, an archive file with an .apk suffix. All the code in a single .apk file is considered to be one application and is the file that Android-powered devices use to install the application.

Once installed on a device, each Android application lives in its own security sandbox:

- The Android operating system is a multi-user Linux system in which each application is a different user.
- By default, the system assigns each application a unique Linux user ID (the ID is used only by the system and is unknown to the application). The system sets permissions for all the files in an application so that only the user ID assigned to that application can access them.
- Each process has its own virtual machine (VM), so an application's code runs in isolation from other applications.
- By default, every application runs in its own Linux process. Android starts the process when any of the application's components need to be executed, then shuts down the process when it's no longer needed or when the system must recover memory for other applications.

Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role—each one is a unique building block that helps define your application's overall behavior.

There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of application components:

Activities

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email application, each one is independent of the others. As such, a different application can start any one of these activities (if the email application allows it). For example, a camera application can start the activity in the email application that composes new mail, in order for the user to share a picture.

Services

A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it.

Content providers

A content provider manages a shared set of application data. You can store the data in the file system, an SQLite database, on the web, or any other persistent storage location your application can access. Through the content provider, other applications can query or even modify the data (if the content provider allows it). For example, the Android system provides a content provider that manages the user's contact information. As such, any application with the proper permissions can query part of the content provider (such as ContactsContract.Data) to read and write information about a particular person.

Content providers are also useful for reading and writing data that is private to your application and not shared. For example, the NotePad sample application uses a content provider to save notes.

Broadcast receivers

A broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system—for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Applications can also initiate broadcasts—for example, to let other applications know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work. For instance, it might initiate a service to perform some work based on the event.

A unique aspect of the Android system design is that any application can start another application's component. For example, if you want the user to capture a photo with the device camera, there's probably another application that does that and your application can use it, instead of developing an activity to capture a photo yourself. You don't need to incorporate or even link to the code from the camera application. Instead, you can simply start the activity in the camera application that captures a photo. When complete, the photo is even returned to your application so you can use it. To the user, it seems as if the camera is actually a part of your application.

Because the system runs each application in a separate process with file permissions that restrict access to other applications, your application cannot directly activate a component from another application. The Android system, however, can. So, to activate a component in another application, you must deliver a message to the system that specifies your intent to start a particular component. The system then activates the component for you.

The Manifest File

Before the Android system can start an application component, the system must know that the component exists by reading the application's AndroidManifest.xml file (the "manifest" file). Your application must declare all its components in this file, which must be at the root of the application project directory.

The manifest does a number of things in addition to declaring the application's components, such as:

- Identify any user permissions the application requires, such as Internet access or read-access to the user's contacts.
- Declare the minimum API Level required by the application, based on which APIs the application uses.
- Declare hardware and software features used or required by the application, such as a camera, bluetooth services, or a multitouch screen.
- API libraries the application needs to be linked against (other than the Android framework APIs), such as the Google Maps library.

What is GPS?

The Global Positioning System (GPS) is a space-based global navigation satellite system (GNSS) that provides location and time information in all weather, anywhere on or near the Earth, where there is an unobstructed line of sight to four or more GPS satellites. It is maintained by the United States government and is freely accessible by anyone with a GPS receiver with some technical limitations which are only removed for military users.

The GPS project was developed in 1973 to overcome the limitations of previous navigation systems, integrating ideas from several predecessors, including a number of classified engineering design studies from the 1960s. GPS was created and realized by the U.S. Department of Defense (USDOD) and was originally run with 24 satellites. It became fully operational in 1994.

In addition to GPS, other systems are in use or under development. The Russian GLObal NAvigation Satellite System (GLONASS) was in use by only the Russian military, until it was made fully available to civilians in 2007. There are also the planned Chinese Compass navigation system and the European Union's Galileo positioning system.

Eclipse

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework) Groovy and Scheme. It can also be used to develop packages for the software Mathematica. The IDE is often called Eclipse ADT (Ada Development Toolkit) for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

The initial codebase originated from VisualAge. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its abilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Released under the terms of the Eclipse Public License, Eclipse is free and open source software. It was one of the first IDEs to run under GNU Classpath and it runs without issues under IcedTea.

Sms Messaging Service

It sends an SMS message with user current location. We use the SmsManager class. We call the getDefault() static method to obtain an SmsManager object. The sendTextMessage() method sends the SMS message with a PendingIntent. When an SMS is sent successfully, it will display a "SMS sent" message. When it is successfully delivered, it will display a "SMS delivered" message.

GUI

GUI in android is completely XML based, so in order to be efficient in graphical layout and event handling, one must have a thorough knowledge of XML. We have made several GUI files showing various activities that are present in our application.

***Latitude and Logitude Locator* Service**

We programmed an applicaton with our main GPS Emergency Service to easily get location of user on google map by finding the ***Latitude and Logitude*** on Google Map.

It is for conversion of latitude and longitude in degrees, minutes, seconds to decimal degrees. Display the point on the map and upload to Google Maps

Sqlite Database

SQLite Database is a freeware, public domain, open source visual tool used to create, design and edit database files compatible with SQLite. It is meant to be used for users and developers that want to create databases, edit and search data using a familiar spreadsheet-like interface, without the need to learn complicated SQL commands.

Andoid Smart phones use SQLite Database. The local library implements natively within C/C++ and supply Sqlite. SQLite Database has methods to create, delete, execute SQL commands, and perform other common database management tasks.

We have stored various emergency numbers of various countries, so the user may use it in times of need without actually knowing in which part of world he/she is and without having any information about his/her current location, he can still use this application with ease.

Android Contact List

Android Contact List is used to fetch all the phone number saved in Android Phone Directory.

For this we require these things.

1) Permissions

Add a permission to read contacts data to your application manifest.

<uses-permission android:name="android.permission.READ_CONTACTS"/>

2) Calling the Contact Picker

Within your Activity, create an Intent that asks the system to find an Activity that can perform a PICK action from the items in the Contacts URI.

Intent intent = new Intent (Intent.ACTION_PICK,
ContactsContract.Contacts.CONTENT_URI);

3) Listening for the Result

You can get the URI of the selected contact by calling getData() on the *data* Intent parameter. To get the name of the selected contact you need to use that URI to create a new query and extract the name from the returned cursor.

Database of various countries

We have created a backend database using SQLite which contains names of various countries along with all of their emergency phone numbers, so that user is always updated with the correct data irrespective of his current location, whether he is in any country, he doesn't has to know that country's emergency numbers and still be able to report with perfect location to the right authorities.

Template

Application uses any one of template which is stored in database. We give authority to user to edit pre-exist template from sqlite database which saves a lot of time and create flexibility in application.

HOW OUR APPLICATION IS BETTER THAN CURRENTLY EXISTING TECHNOLOGIES

In a google map app tells the current user location, but has no feature of sending this location**.** It is slow and battery ineffective, and there is no database containing emergency numbers, so this can't be used in emergency.

Our app has this automatic updating feature that sets its current country location with its emergency numbers from already provided database, which can also be updated or edited by user. We save a lot of battery by just using the coordinates and using the GPS in the end, which is much efficient. We can send sms from automatic templates and also user defined templates. We can contact as many persons/numbers at one single time like police, fire brigade etc. for informing, this makes our app a lot more better than conventional calling, which on the other hand also needs the user to know his location and also remember the numbers. Therefore in totality our app is a lot more advanced, simple to use, quick and efficient feature of reporting emergencies.

This also has a bonus feature that you can send your location to your friend if he wants to find you. So it is also useful to find people using it besides reporting emergencies.

If you get lost and don't know your path then also this application gives you the function to send your location to anyone like a forest officer, to come and take you from forest!

We have also made a client side application in which if we put coordinates, it will show us the sender's location on map.

CONCLUSION

Emergency GPS Messaging Service is used to send SMS to as many concerned authorities/persons as required with the type of emergency and the exact location coordinates of the place of emergency.

This SMS consists following information about the user:

1) User GPS Location with accuracy of 10 to 15 meters in land area and about 30 meters in mountain areas.
2) Text message (Template) which is already in the database of mobile or customized by user, is also saved in database after sending message.
3) User chooses multiple phone numbers to send emergency message by just clicking on checkbox or take it from application's contact list or by just typing it.
4) GPS messaging is efficient in our android application which saves a lot of battery power because when you start android application with GPS activity, it takes a lot of memory and sometime hang the smartphone's operating system. So to reduce this problem, we make connection with GPS in our last activity when we send message within fraction of seconds.
5) In our project, we have used evolutionary prototyping model of software development in which we divide whole process in five parts, namely Sqlite Database, SMS Service, Map API, GUI and Contact List Menu.
6) We have also provided instructions for usage of the application if at any point of time know any information.
7) Google Maps satellite images are not updated in real time, they are several months old. But still our application functions well as we our using the coordinates of place and not its English Address.

- *Assets*

  Assets are things in an android application which are included for adding more functionality to the application like database, or any other file which is to be used in application. For saving our contacts on the database, we used SQLite as the backend in which we saved all the emergency contact numbers of all countries. To use this database we faced many problems like there was no available database as such, so we have to make all by ourselves from the start, this consumed much of our software development time.

  We also had to make another database to be used as current working database in which we feed the current user information like his present country. This database is updated accordingly as user location is changed.

- *Problem in retrieving data*

  While querying the database, we had to, let's say select from japan, the we were using the query, "Select number from "Japan" where emergency type = 'fire'", we were not getting correct result, after lots of mind boggling we finally found that the query has to be like, "Select number from "'Japan'" where emergency type = 'fire'". Notice the extra pair of inverted comma used in "'Japan'". This happened because in the database, Japan is stored as 'Japan' and not Japan.

- *String tokenizer*

  The nos. in country database are stored as 123-456-789. But this form was unacceptable for sending SMS. We spent a lot of time in figuring out where the real problem is. After we recognized the problem, using toast notifications (these are special flashing notifications on screen), we searched for a method which could retrieve the required number from database as a string and finally remove all the unwanted tokens like hyphens and at last we found about the function available I java API. stringTokenizer("1233-456-789"). This method could remove the tokens and finally give the string as 123456789. This number was later parsed into integer using parseInt method but again we had severe problems as the data range of integers was not enough to hold some large numbers, so finally

changer our parsing method to parseDouble which would parse it in type Double that can handle required data size.

- *Checklist*

  While creating menu entries we placed checkboxes with texts but while we were using them, they were interpreted as separate entities and did not relate them. After some time we devised a way out. We used customary checkboxes names instead of buttons so that they can be used for touching functions

REFERENCES

[1] Google Android Developer [Online], http://developer.android.com/guide/index.html, August, 2011

[2] Sai Geetha, Solution Designer, Bangalore, Karnataka, India, http://saigeethamn.blogspot.com/, [Online], September, 2011

[3] Google's official blog, [Online], http://android-developers.blogspot.com/, 2008

[4] Sayed Hashimi, Satya Komatineni, Dave MacLean, Apress, "Pro Android 2", 2010

[5] Tony Hillerson, O'Reilly Media, Creativetechs, "Developing Android Applications with Java, Part 1", January, 2010

Remarks by Board members