BHEL-Tracker

Technical document

1. Introduction
   1. Purpose of this document

The purpose of this document is to provide technical details of software components that this project is made of.

* 1. Document organisation

This document is organised as follows

* Section 1, Introduction, describes contents of this guide, used documentation during developing process.
* Section 2, Android OS, describes technical details about android platform.
* Section 3, Google play services, describes details about Google services used in this project.
* Section 4, Google maps API, describes implementation of Google maps and its features.
* Section 5, BHEL-Tracker, describes main classes, procedures and methods used in android application.
  1. Intended Audience

The intended audience is:

* Project customer
* Project supervisor
* Project team members
* Users
  1. References
* Configuration file
* Android Application
* Project Definition Document

1. Android OS

Android is a mobile operating system running on the Linux kernel. It allows developers to write managed code in Java programming language, controlling the device via Java libraries developed by Google.

The central feature of Android operating system is that it allows an application to reuse other application's components, provided those applications permit it. For this to work, the system must be able to start an application process when any parts of it are needed, and instantiate the Java classes for that part. Therefore, unlike the applications on most other operating systems, Android applications don't have a single entry point for everything in the application (no main() function, for example). Instead, they are composed of components that the system can instantiate and run as needed. There are four types of those components:

1. Activities -present a visual user interface for one focused endeavour

2. Services -run in the background for an indefinite period of time

3. Broadcast receivers -receive and react to broadcast announcements

4. Content providers -make a specific set of the application's data available to other applications

Activities are the only component that deals with user interface, and because of that, this application is based on activities. Some basic aspects of the Android activities are described in the following text.

An activity has essentially three states:

• It is active or running when it is in the foreground of the screen

• It is paused if it has lost focus, but is still visible to the user

• It is stopped if it is completely obscured by another activity

If an activity is paused or stopped, the system can drop it from memory either by asking it to finish (calling its finish() method), or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state. This means that the application state must be saved every time the application is paused, and restored every time the application is resumed.

As an activity transitions from state to state, it is notified of the change by calls to the following protected methods:

void onCreate(Bundle savedInstanceState);

void onStart();

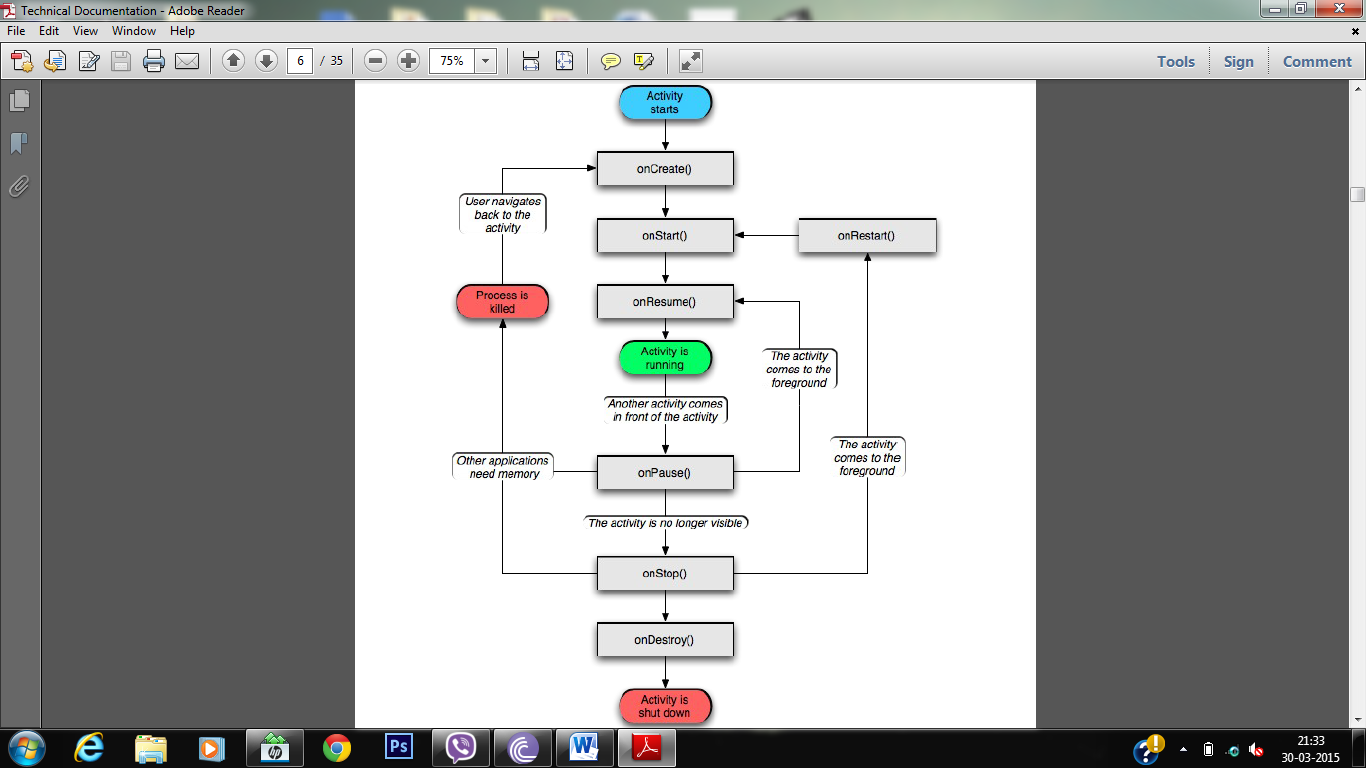
void onRestart();

void onResume();

void onPause();

void onStop();

These methods are hooks that can be overridden to do appropriate work when the activity changes states. These methods define a life-cycle of an activity. The following diagram illustrates life-cycle of an activity. The colored ovals are major states the activity can be in. The square rectangles represent the callback methods you can implement to perform operations when the activity transitions between states.



Besides the source code, Android applications usually contain some additional resources. Most of the resources are defined using XML files. For example, layout of the user interface, animations, constants can all be defined using XML files. Every Android application has a directory for resources (res/) and a directory for assets (assets/). Difference between resources and assets directory is that anything in the resources directory is easily accessible through the R class, which is compiled by Android. Whereas, anything in the assets directory maintains its raw file format, and is read as a stream of bytes using AssetManager class.

1. Google play services

Google play services library is freely provided by Google and how to use this in your app is also explained in developer.android.com

3.1 Google Technology

Google Play services provides you with easy access to Google services and are tightly integrated with the Android OS. Easy-to-use client libraries are provided for each service that let you implement the functionality you want easier and faster.

#### 3.2 Standard Authorization

All products in Google Play services share a common authorization API that leverages the existing Google accounts on the device. You and your users have a consistent and safe way to grant and receive OAuth2 access tokens to Google services.

#### 3.3 Automatic Updates

Devices running Android 2.3 or higher that have the Google Play Store app will automatically receive updates to Google Play services. Enhance your app with the most recent version of Google Play services without worrying about your users' Android version.

3.4 Features Used

* Places - Using the Google Places API for Android, you can build location-aware apps that respond contextually to the local businesses and other places near the device. Use the built-in place picker UI widget and API methods to find the device’s current place, autocomplete users’ queries, and more.
* Location settings - While the FusedLocationProviderApi combines multiple sensors to give you the optimal location, the accuracy of the location your app receives still depends greatly on the settings enabled on the device (GPS, wifi, airplane mode, and others). Using the new SettingsApi class, you can bring up a Location Settings dialog which displays a one-touch control for users to change their settings without leaving your app.

3.5 The benefits for app

Google Play services gives you the freedom to use the newest APIs for popular Google services without worrying about device support. Updates to Google Play services are distributed automatically by the Google Play Store and new versions of the client library are delivered through the Android SDK Manager. This makes it easy for you to focus on what's important: your users' experience.

1. Google Maps Android API v2

Allow your users to explore the world with rich maps provided by Google. Identify locations with **custom markers**, augment the map data with **image overlays**, embed **one or more maps** as fragments, and much more.

4.1 Add maps to app

With Google Maps Android API v2, you can embed maps into an activity as a fragment with a simple XML snippet. The new Maps offer exciting features such as 3D maps; indoor, satellite, terrain, and hybrid maps; vector-based tiles for efficient caching and drawing; animated transitions; and much more.

4.2 Customize the map

Add markers onto the map to indicate special points of interest for your users. You can define custom colors or icons for your map markers to match your app's look and feel. To further enhance the app, draw polylines and polygons to indicate paths or regions, or provide complete image overlays.

4.3 Control the user's view

Give your users a different view of the world with the ability to control the rotation, tilt, zoom, and pan properties of the "camera" perspective of the map.

4.4 Add Street View to app

Embed Street View into an activity and let your users explore the world through panoramic 360-degree views. Programmatically control the zoom and orientation (tilt and bearing) of the Street View camera, and animate the camera movements over a given duration.

1. BHEL-Tracker

Name of the product developed is BHEL-Tracker.

There are different types of source files in this application.

1. Manifest file

There is an **AndroidManifest.xml** file in the package. This file contains information like SDK version (versions of android supportable), permissions for the app (displayed at the time of installation), features of device used by the app, meta data like API key and application details like info for all the activities.

1. JAVA class files

There are approx. 12-13 java class files in the application.

When you launch the app, **login activity** starts. This activity uses two other java class files namely **login database adapter** and **database helper** for the smooth working of database and saving the info of users in it.

Depending on your choice to login or signup, it will pass on the control to other activities.

For signup, it will move to **signup activity**.

After signing up or logging in, depending on whether you are a carrier or a tracker, it will pass controls to **carriert** or **tracker** activity resp.

The **carriert activity** uses several other java class files, **location database adapter** and **database helperf** to store the locations. It also uses **feature database adapter** and **database helperf** to store unique features of the device.

The **tracker activity** uses **location database adapter** and **database helper** to fetch the locations. It also uses google maps API to show the traversed route on the map.

Both, the **carrier** and **tracker** activity has a menu with three options, **help, settings and logout**. The help option use another java class file **help** to display a set of instructions. The settings option will directly move you to location settings and a logout option will move you back to the very first page of application.

1. XML files under res section

Res stand for resources. We have several folders under res section.

The **drawable** folder contains the logo.

The **layout** folder contains the xml files which define the layouts which are used by java activities.

The **menu** folder contains the xml files which define the structure for menu.

The **values** folder contains dimensions and strings type xml files which contain the variables used at different points in the application.