**1.3.2 Touch, gesture and Multitouch**

The touchscreen is one of the most popular and almost a must for the smartphones and other hand-held device. It provides the most convenient and quick way for a user to interact with the device. If utilized well, it can transcend a need for detailed instructions. Multitouch provides a way to track more than one finger down at the same time. This is often used to zoom or rotate a view. Some touch events are available transparently to the developer without the need to implement their detailed behaviors. Custom gestures can be defined as needed. It is important to try to maintain a consistent usage of touch events as compared to other applications.

**1.3.3 Multi-process and App Widgets**

Android does not limit the processor to a single application at a time. Instead, Android allows multiple threads within the same application and they are all managed by the system. This has the benefit that background tasks can be run while a user engages the device in a foreground process. For example, the same application can allow the user to play the game at the same time the application can also be listening to the new users joining the game hub and trigger an action to the user device if necessary, at the same time playing music. App Widgets are mini applications that can be embedded in other applications (such as the Home screen). They can process events, such as start a music stream or update weather, while other applications are running. Multiprocessing has the benefit of a rich user experience. Such capabilities expose the device to the applications which might be consuming too much power so as a developer it is your responsibility to make sure your application does not drain the battery.

**1.3.4 Access to Hardware**

Android provides API libraries which enable developers to access device hardware, should their application need to do so. These libraries help developers to avoid the need to create specific implementations of your software for different devices, so you can create Android applications that work as expected on any device that supports the Android software stack. The hardware access includes APIs for location-based hardware (such as GPS), the camera, audio, network connections, Wi-Fi, Bluetooth, accelerometers, the touchscreen, and power management.

**1.3.5 Maps and Location Based Services**

Android comes with Google native map support which allows developers to create different map-based applications. Android lets you create activities that integrate with interactive Google Maps as a part of your user interface, with full access to maps that you can control programmatically.

Android provides location-based services which manage GPS and Google's GSM cell-based location technology which is used to determine the current position of the Android device. Such services are important if your application provides services whose contents may differ from one location to another. These also ensure the performance of your application regardless of the device manufacturer. Putting together maps with locations in Android you will be able to forward and reverse geocoding using Android API, which allows you find map coordinates for an address and other position related activities.

**1.3.6 Background Service**

Android supports applications and services designed to run invisibly in the background using its multi-process support discussed earlier. Smart phones are meant to be multifunctional device but the size of their screen limits the ability displaying multiple applications at the same time as we all do desktops and laptops. So while user is interacting with the screen, Android allows applications to run on the background and only pass notification to the user when required. Background services make it possible to create invisible application components that perform automatic processing without direct user action. Background execution allows your applications be event-driven and to support regular updates without blocking user, which is perfect for monitoring ongoing events, or any other dynamic events.

**1.3.7 Data Storage**

Android provides different ways for storing data, as efficient data storage and retrieval are essential for a mobile device due to its limited capacity. Android provides a lightweight relational database for each application using SQLite. By default each application database is sandboxed and its content is available only to the application that created it. Android provides a mechanism of sharing data from the database through Content Providers which manage sharing of application databases.

Not every application needs to use database even though they might want to store some simple data such as user settings preferences, is such cases Android provides Shared Preference, that is a light weight alternative of storing data in a key/value form which allows you to store simple data such as user name, password and other application state preferences. Android also allows you to store your data in static file dynamically.

**1.3.8 Inter-application Communication**

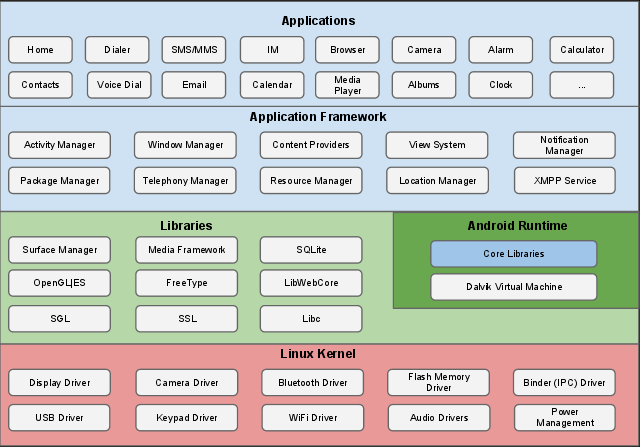
Android includes three techniques for transmitting information from your applications for use elsewhere: Notifications, Intents, and Content Providers. Notifications are used to notify user about certain actions. Intents provide a mechanism for message-passing within and between applications. Using Intents you can broadcast a desired action such as querying for nearby Bluetooth device. Android also allows you to pass message within your application between components through Message Handlers.

**1.3.9 Enhancing Home Screen**

Android provides ways for developers to interact their applications in a way that it allows users to interact with the application from the home page. Widgets, Live Folders, and LiveWallpaper let you create dynamic application components that provide a window into your applications or offer useful and timely information directly on the home screen. By allowing users to interact with the application directly from the home screen, they get instant access to interesting information without needing to open an application, and you get a dynamic shortcut into your application.

## 1.4 Android Software Stack

The Android software stack is composed of elements shown in Figure 1.1.

  
Figure 1.1: Android software stack

Linux Kernel provides core services such as Network, hardware drivers, power, process and memory management. Linux Kernel also provides an abstraction layer between device hardware and the rest of the stack.

Libraries running on top of Linux Kernel handle core libraries SSL and WebKit for integrated web browser and internet security, SQLite for native database support, surface manager for display management, media library for playback of video and audio.

Android runtime is what makes Android device Android rather than Linux mobile implementation. The runtime is responsible for providing Android applications with the Android libraries which you use to develop Android applications. Android runtime includes Core libraries and Dalvik virtual machine. Each Android application runs in a separate process, with its own instance of the Dalvik virtual machine (VM). Based on the Java VM, the Dalvik design has been optimized for mobile devices. The Dalvik VM consumes small amount of memory, and multiple instances of the Dalvik VM can run concurrently on the same device.

Application framework provides the classes used to create Android applications and generic abstract for hardware access. The framework is also responsible for managing user interface and application resources.

Application layer holds all applications, native apps and third party apps through the same API libraries. Application layer runs within Android runtime using services provided by the Application framework layer.