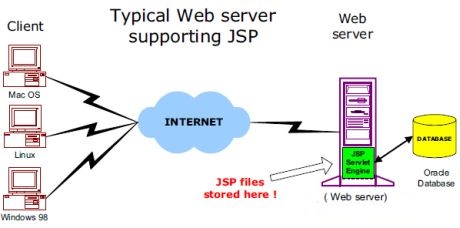
**METHODOLOGY**

* 1. **JSP Technology[7][12]**

The web server needs a JSP engine ie. container to process JSP pages. The JSP container is responsible for intercepting requests for JSP pages. A JSP container works with the Web server to provide the runtime environment and other services a JSP needs. It knows how to understand the special elements that are part of JSPs.

Following diagram shows the position of JSP container and JSP files in a Web Application.



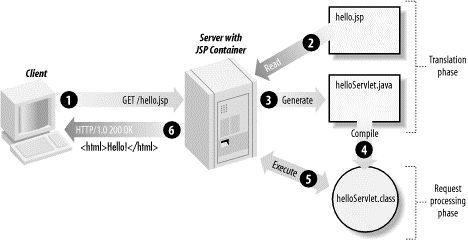
**Figure 3.1: Position of JSP container and JSP files**

* 1. **JSP Processing**

The following steps explain how the web server creates the web page using JSP:

* As with a normal page, the browser sends an HTTP request to the web server.
* The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of .html.
* The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior of the page.
* The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
* A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format, which the servlet engine passes to the web server inside an HTTP response.
* The web server forwards the HTTP response to your browser in terms of static HTML content.
* Finally web browser handles the dynamically generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be shown below in the following diagram:



**Figure 3.2: JSP Processing**

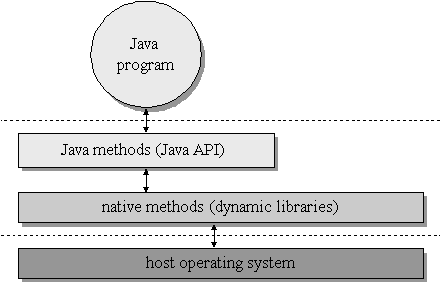
Typically, the JSP engine checks to see if a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with other scripting languages (such as PHP) and therefore faster.

So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

* 1. **Android [11]**

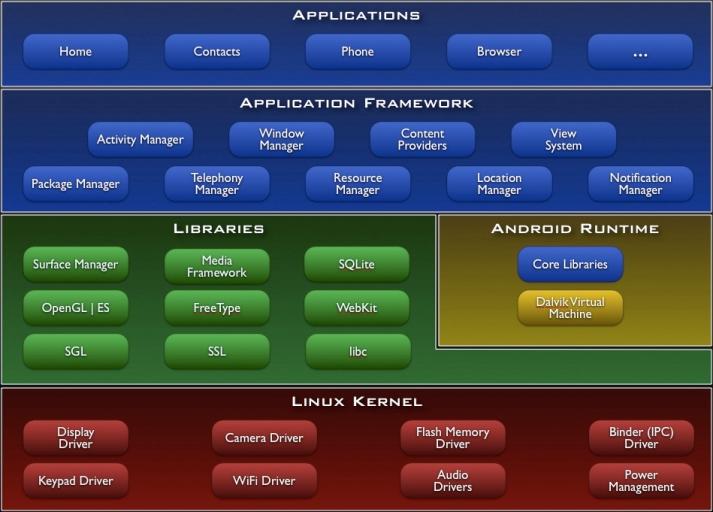
Android is a software stack for mobile device that includes an operating system, middleware and applications. Android is powered by Linux kernel, initially developed by Google and later the Open Handset Alliance. It allows developers to write managed code in java language, controlling the device via Google developed java library. Not like other famous rivals such as Microsoft window mobile or Symbian OS, android use developed java library because java is not just a programming language; it’s a complete dynamic platform offers powerful support for embedded devices that must maintain some form of dynamic behavior.

Moreover, java runtime environment can be integrated into almost any embedded device while java virtual machine includes interfaces that allow it to be readily integrated with RTOS and other native library. The RTOS supports multi-thread (scheduling), memory management, net working, and peripheral management for java VM.



**Figure 3.3: Android Environment**

* 1. **Android Architecture**

****

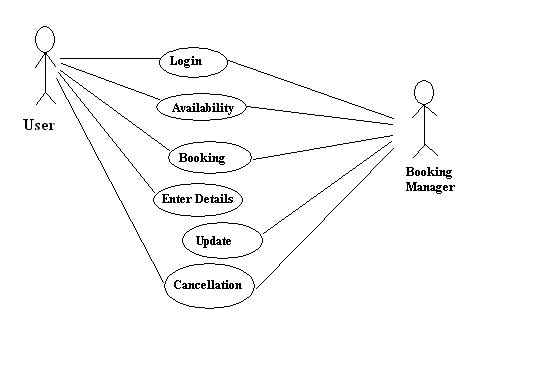
**Figure** **3.4: Android architecture**

* **Application layer**: Android provides a lot of applications which come with its release including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language. Moreover, the number of developers who interested in developing Android’s application is increasing and provides a huge market of application to chose.
* **An application framework layer:** Enabling reuse and replacement of components. Not like Window mobile which restricts developer from system API, developers have full access to the same framework APIs used by the core’s applications. The strength points of Android is that the application architecture is designed for reusing of components which means any application can publish its capabilities and any other application may the make use of those capabilities base on the definition of Intent which will be described later.
* **Views:** It is a flexible definition. It can be a list, a grid, text box, button or even an embedded web browser. View in Android is very different from the definition of “view” in Symbian OS or Window mobile which often means the container in which graphic components are organized and displayed to user.
* **Content provider:** store and retrieve data and make it accessible to all applications. They're the only way to share data across applications; there's no common storage area that all Android packages can access.
* **Resource manager:** Resources are external files (that is, non-code files such as image, icon, and string for internationalization) that are used by developer’s code and compiled into their application at build time. Android supports a number of different kinds of resource files, including XML, PNG, and JPEG files. The XML files have very different formats depending on what they describe. Resources are externalized from source code, and XML files are compiled into a binary, fast loading format for efficiency reasons. Strings likewise are compressed into a more efficient storage form. All resource has its own id and will be added to special interface file R.java automatically.
* **Libraries:** They are all written in C/C++ internally, but you’ll be calling them through Java interfaces. These capabilities are exposed to developers through the Android application framework. Some of core libraries are:

1. System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded linux based devices.
2. Media Libraries - based on PacketVideo's OpenCORE; 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
3. Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
4. LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view.
5. SGL -the underlying 2D graphics engine.
6. SQLite - a powerful and lightweight relational database engine available to all applications

* **Android Runtime:** A set of core libraries provides most the functionality available in the core library of java programming language. Android runtime includes the Dalvik Virtual Machine. Dalvik runs dex files, which are coverted at compile time from standard class and jar files. Dex files are more compact and efficient than class files, an important consideration for the limited memory and battery powered devices that Android targets. The core Java libraries are also part of the Android runtime. They are written in Java, as is everything above this layer. Here, Android provides a substantial set of the Java 5 Standard Edition packages, including Collections, I/O, and so forth.
* **Dalvik Virtual Machine** which is optimized for mobile devices. Dalvik is a major piece of Google’s Android, runs Java platform applications which have been converted into a compact Dalvik Excutable format suitable for systems that are constrained in terms of memory and processor speed. Unlike most virtual machines true java virtual machines which are stack machines, Dalvik VM is register based architecture. Generally, however, stack based machines must use instructions to load data on the stack and manipulate that data and thus require more instructions than register machines to implement the same high level code. However, the instructions in a register machine must encode the source and destination registers and therefore tend to be larger. This difference is primarily of importance to VM interpreters for whom opcode dispatch tends to be expensive and other factors are relevant for JIT compilation. Being optimized for low memory requirements, Dalvik VM use less space, has no JIT compiler and uses its own byte code, not java byte code.
* **Linux Kernel:** Starting at the bottom is the Linux kernel. Android uses it for its device drivers, memory management, process management and networking. However we will never be programming to this layer directly. Android relies on Linux kernel version 2.6 for core system services. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. One of the unique and powerful qualities of Android is that all applications have a level playing field which means that the applications Google writes have to go through the same public API that we use. We can even tell Android to make our application replace the standard applications.
  1. **Use Case Diagram**

The team worked upon the use case diagram and on designing. Fig. below is the use case diagram.



**Figure 3.5: Use Case Diagram**

The actors are:

**Booking Manager:**

Booking Manager or the operator is the member of the staff or is the incharge of the parking lot organisation. He is responsible for the booking and looks after the management of the parking slots. Parking lot management involves booking, unbooking and managing the availablity of the parking slots.

**User:**

User is the registered user who uses the android application on his/her smart phone. The user can look for the available parking slots, book one emolty booking slot, cancel the booked slot by him/her or even extend the time period for which he/she might be using the parking slots.

There are different actions that are performed by the actors:

1. Login: Actor’s login for authentication and indentity verification is done through this action.
2. Availability: Actor checks for availability of parking slots.
3. Booking: Actor chooses a particular empty slot and books it for on spot occupation or for latter user.
4. Enter Details: Actor has to enter his/her personal details during registration. Personal details are required to maintain database about the personal information regarding actors which can be latter used to identify each them.
5. Update: Actors can update their filled information.
6. Cancellation:Actors can cancel their booked slots as a consequense of which the cancelled slot is returned back to the free pool of parking slots.