BAB II
LANDASAN TEORI

II.1 Pengertian Smartphone

Dalam pengertian singkat, *smartphone* adalah sebuah *device* yang memungkinkan untuk melakukan komunikasi (seperti menelepon atau sms) juga di dalamnya terdapat fungsi PDA (*Personal Digital Assistant*) dan berkemampuan seperti layaknya komputer. *Smartphone* juga bisa diartikan sebagai alat komunikasi atau telepon seluler yang dilengkapi dengan *organizer digital*. *Smartphone* merupakan pengembangan dari telepon seluler yang kemudian ditambahkan fiitur dan fasilitas lainnya sehingga menjadi telepon yang cerdas.

Sebenarnya tidak ada definisi standar perusahaan mengenai *smartphone*. Umumnya suatu ponsel dikatakan sebagai *smartphone* bila dapat berjalan pada *software operating system* yang lengkap dan memiliki *interface* dan *platform* standar bagi pengembangan aplikasi. Sementara itu ada yang mengatakan *Smartphone* adalah ponsel sederhana dengan fitur canggih seperti kemampuan mengirim dan menerima email, menjelajah internet dan membeca *e-book, built in full keyboard* atau *external USB keyboard,* atau memiliki konektor VGA. [1]

II.2 Pengertian Android

Android adalah sistem operasi yang digunakan di *smartphone* dan juga *tablet* PC. Fungsinya sama seperti sistem operasi *Symbian* di Nokia, iOS di *Apple* dan *BlackBerry* OS. Android tidak terikat ke satu merek *handphone* saja, beberapa vendor terkenal yang sudah memakai Android antara lain Samsung, Sony

Ericsson, HTC, Nexus, Motorolla, dan lain-lain.

II.2.1 Sejarah Android

Android adalah sistem operasi untuk telepon seluler yang berbasiskan Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri sehingga dapat digunakan oleh bermacam peranti penggerak. Awalnya Google Inc. membeli Android Inc. pendatang baru yang membuat *software* (perangkat lunak) untuk telepon genggam. Kemudian untuk mengembangkan Android di bentuklah *Open Handset Alliance* yang merupakan gabungan dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi termasuk Google, HTC, Intel, Motorola, Qualcomm, TMobile, dan NVidia.

Pada saat perilisan perdana Android pada tanggal 5 november 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Terdapat dua jenis distributor sistem operasi Android. Pertama yang dapat dukungan penuh dari Google atau *Google Mail Service* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung dari Google atau dikenal sebagai *Open Handset Distribution* (DHD). [2]

II.2.2 Kelebihan Android

Setiap sistem operasi memiliki kelebihan dalam kinerja dan fitur-fiturnya, berikut adalah kelebihan sistem operasi android :

1. *User interface* menarik.
2. *Open Source*. Sumber bebas dan terbuka.
3. Multitasking. Bisa menjalankan berbagai aplikasi dalam satu waktu.
4. Kemudahan dalam notifikasi. Setiap ada SMS, Email, atau bahkan artikel terbaru dari RSS *Reader*, akan selalu ada notifikasi di *Home Screen* Ponsel Android.
5. Akses Mudah terhadap Ribuan Aplikasi Android lewat Google Android *App Market*.
6. Pilihan Ponsel yang beranekaragam. Android tersedia di ponsel dari berbagai produsen, mulai dari Sony Ericsson, Motorola, HTC sampai Samsung.
7. Bisa menginstal ROM yang dimodifikasi.
8. Widget.
9. Terintegrasi dengan google.
10. Aman dari virus. Karena menggunakan kernel dari linux.

II.3 *Point Of Sale*

Pengertian *Point Of Sale* atau yang biasa yang disingkat POS yaitu, merupakan kegiatan yang berorientasi pada penjualan serta sistem yang membantu proses transaksi. Setiap POS (*Point Of Sale)* terdiri dari *hardware* dan *software* dimana kedua komponen tersebut digunakan untuk setiap proses

transaksi. POS *software* merupakan komponen utama dari sistem POS (*Point Of Sale)* yang pada akhirnya menentukan jalannya proses, seperti apa yang harus dilakukan dan bagaimana harus melakukan. Sedangkan *hardware* POS dibutuhkan untuk menjalankan fungsinya, membantu proses pembayaran dan membuat tanda terima untuk pelanggan.

Dalam hal pemilihan *hardware* ini, sebaiknya mencocokan dengan lingkungan kerja, seperti yang akan digunakan oleh penulis pada laporan skripsi ini adalah *barcode scanner,* yang merupakan bagian terpenting untuk mempercepat proses pemasukkan barang dan proses pelayanan penjualan.

Sistem POS *(Point OF Sale)* berdasarkan komputer terdiri dari platform komputer, berbagai perangkat periferal khusus, dan aplikasi perangkat lunak P0S yang mengikat semua bersama-sama. Berikut adalah komponen POS *(Point Of Sale)* :

1. *Software*

Software membuat perputaran, atau dalam hal ini, mengontrol semua perilaku POS *(Point OF Sale)*. Perangkat lunak ini dapat dibagi menjadi empat kategori :

a. *The operating system (0S)*
b. *The P0S application software*
c. *The credit card authorization software*
d. *The accounting software.*

2. *Hardware*

Sebagian pengguna komputer pasti akrab dengan *hardware* komputer seperti *keyboard* dan *mouse*. Sistem POS *(Point OF Sale)* berbasis komputer hanya

membutuhkan sistem operasi komputer yang standar (*The Small Business Depot*; 2005)

II.4. Android SDK

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di *release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi netral, android *member* anda kesempatan unutk membuat aplikasi yang kita butuhkan yang merupakan aplikasi bawaan *Hadphone/Smartphone* (Ardzi Firman Ihtiyar Rohanianto, 2014; hal 8) [3].

Beberapa fitur-fitur android yang paling penting adalah :

1. *Framework* : aplikasi yang mendukung pengganti komponen dan reusable. b. *Dalvik Virtual Machine* dioptimalkan untuk perangkat *mobile*
2. Integrated *Browser* verdasarkan engine *open source* WebKit.
3. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi opengl ES 1,0 (Opsional Ekselerasi *hardware*)
4. SQLite untuk penyimpanan data.
5. Media *Support* yang mendukung audio, video, dan gambar (MPEG4, H.264,

MP3, AAC, AMR, JPG, PING, GIF), GSM *Telephony* (tergantung H*ardware*)

6. *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*)
7. Kamera, GPS, Kompas, dan *Accelerometer* (tergantung *hardware*)
8. Lingkungan *Development* yang lengkap dan termasuk pernagkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse.

II.5 Aptana Studio

Aptana Studio adalah pengembangan *open source* IDE (*Integrated Development Environment*) yang mengkhususkan diri dalam membangun aplikasi web. Aptana Studio telah ada sejak tahun 2008. Bahasa pemrograman yang didukungan adalah untuk HTML, CSS, *JavaScript*, Ruby, Rails, PHP, Python, dan banyak lainnya. Sejak Versi 3.0.4, tim pengembangan Aptana Studio telah terintegrasi HTML5 dan CSS3 spesifikasi terbaru. Hal ini memungkinkan kemampuan browser paling modern untuk dimanfaatkan untuk pengembangan Aptana Studio. Versi terbaru telah diunduh lebih dari 6 juta kali.

Aptana Studio dengan tugas-tugas lain, seperti FTP dan *Git integration*, *JavaScript libraries*, dan *JavaScript debugging*.. Selanjutnya, *web server Aptana Jaxer* disertakan, yang mengkhususkan diri dalam bekerja dengan aplikasi AJAX dan *website*. Aptana Studio dibangun di atas *platform Java Eclipse*, karena itu adalah sebuah perangkat lunak lintas *platform* dan bekerja pada sistem operasi umum, seperti Linux, Mac OS X, dan *Windows*. [4]

II.6 Pemodelan UML

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek.

Sejarah UML sendiri terbagi dalam dua *fase* sebelum dan sesudah munculnya UML. Dalam *fase* sebelum, UML sebenarnya sudah mulai diperkenalkan sejak tahun 1990an namun notasi yang dikembangkan oleh para ahli analisis dan desain berbeda-beda, sehingga dapat dikatakan belum memiliki *standarisasi.*

Fase kedua; dilandasi dengan pemikiran untuk mempersatukan metode tersebut dan dimotori oleh *Object Management Group* (OMG) maka pengembangan UML dimulai pada akhir tahun 1994 ketika Grady Booch dengan metode OOD (*Object-Oriented Design*), Jim Rumbaugh dengan metode OMT (*Object Modelling Technique*) mereka ini bekerja pada *Rasional Software Corporation* dan Ivar Jacobson dengan metode OOSE (*Object-Oriented Software Engineering*) yang bekerja pada perusahaan *Objectory Rasional*. [5]

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek, yaitu:

1. *Use Case* Diagram untuk memodelkan proses bisnis.
2. *Conceptual* Diagram untuk memodelkan konsep-konsep yang ada di dalam aplikasi.

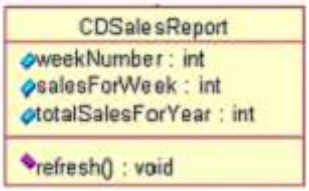
3. *Sequence* Diagram untuk memodelkan pengiriman pesan *(message*) antar objek.
4. *Collaboration* Diagram untuk memodelkan interaksi antar objek.
5. *State* Diagram untuk memodelkan perilaku *objects* di dalam sistem.
6. *Activity* Diagram untuk memodelkan perilaku *Use Cases* dan objek di dalam *system.*
7. *Class* Diagram untuk memodelkan struktur kelas.
8. *Object* Diagram untuk memodelkan struktur objek.
9. *Component* Diagram untuk memodelkan komponen *object.*
10. *Deployment* Diagram untuk memodelkan distribusi aplikasi.

Untuk menggambarkan analisa dan *desain* diagram, UML memiliki seperangkat notasi yang akan digunakan ke dalam tiga kategori diatas yaitu struktur diagram, *behaviour* diagram dan *interaction* diagram. Berikut beberapa notasi dalam UML diantaranya :

1. *Actor*; menentukan peran yang dimainkan oleh *user* atau sistem lain yang berinteraksi dengan subjek. *Actor* adalah segala sesuatu yang berinteraksi langsung dengan sistem aplikasi komputer, seperti orang, benda atau lainnya. Tugas *actor* adalah memberikan informasi kepada sistem dan dapat memerintahkan sistem untuk melakukan suatu tugas.

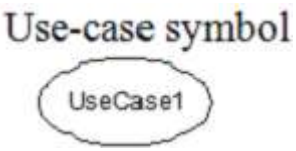
Gambar II.1 Actor
Sumber : Haviluddin, 2011

2. *Class* diagram; Notasi utama dan yang paling mendasar pada diagram UML adalah notasi untuk mempresentasikan suatu *class* beserta dengan atribut dan operasinya. *Class* adalah pembentuk utama dari sistem berorientasi objek.



Gambar II.2 Class Diagram
Sumber : Haviluddin, 2011

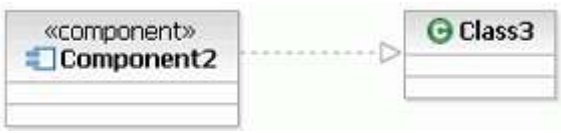
3. *Use Case* dan *use case specification*; *Use case* adalah deskripsi fungsi dari sebuah sistem perspektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem disebut skenario. *Use case* merupakan awal yang sangat baik untuk setiap *fase* pengembangan berbasis objek, *design, testing*, dan dokumentasi yang menggambarkan kebutuhan sistem dari sudut pandang di luar sistem. Perlu diingat bahwa *use case* hanya menetapkan apa yang seharusnya dikerjakan oleh sistem.



Gambar II.3 Use Case
Sumber : Haviluddin,2011

4. *Realization; Realization* menunjukkan hubungan bahwa elemen yang ada di bagian tanpa panah akan merealisasikan apa yang dinyatakan oleh elemen

yang ada di bagian dengan panah.

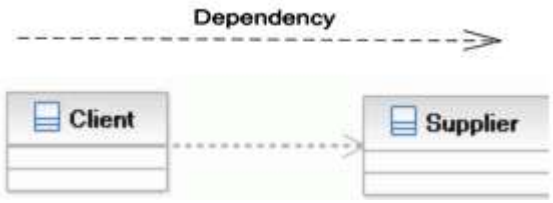


Gambar II.4 Realization
Sumber : Haviluddin, 2011

5. *Interaction; Interaction* digunakan untuk menunjukkan baik aliran pesan atau informasi antar *obyek* maupun hubungan antar *obyek.*

Gambar II.5 Interaction
Sumber : Haviluddin, 2011

6. *Dependency; Dependency* merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Terdapat 2 *stereotype* dari *dependency*, yaitu *include* dan *extend*. *Include* menunjukkan bahwa suatu bagian dari elemen (yang ada digaris tanpa panah) memicu eksekusi bagian dari elemen lain (yang ada di garis dengan panah). *Extend* menunjukkan bahwa suatu bagian dari elemen di garis tanpa panah bisa disisipkan ke dalam elemen yang ada di garis dengan panah.

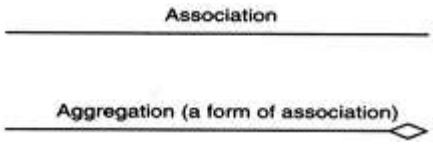


Gambar II.6 Dependency
Sumber : Haviluddin, 2011

7. *Note; Note* digunakan untuk memberikan keterangan atau komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. Note ini bisa disertakan ke semua elemen notasi yang lain.

Gambar II.7 Note
Sumber : Haviluddin, 2011

8. *Association; Association* menggambarkan navigasi antar *class (navigation*), berapa banyak obyek lain yang bisa berhubungan dengan satu obyek (*multiplicity* antar *class*) dan apakah suatu *class* menjadi bagian dari *class* lainnya (*aggregation*).

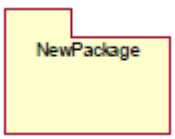


Gambar II.8 Association
Sumber : Haviluddin, 2011

9. *Generalization; Generalization* menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik.

Gambar II.9 Generalization
Sumber : Haviluddin,, 2011

10. Package; package adalah mekanisme pengelompokkan yang dgunakan untuk menandakan pengelompokkan elemen-elemen model.



Gambar II.10 Package
Sumber : Haviluddin, 2011

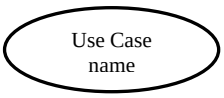
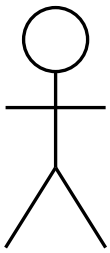
11. *Interface; Interface* merupakan kumpulan operasi berupa implementasi dari suatu class. Atau dengan kata lain implementasi operasi dalam *interface* dijabarkan oleh operasi di dalam *class.*

Gambar II.11 Interface
Sumber : Haviluddin, 2011

Berikut adalah simbol dan komponen-komponen diagram pada UML :

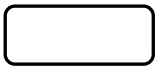
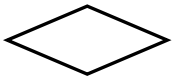
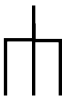
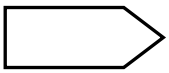
1. *Use Case* Diagram

Tabel II.1 Komponen *Use Case* Diagram

Nama Komponen	Keterangan	Simbol
Use Case	*Use case* digambarkan sebagai lingkaran elips dengan nama *use case* dituliskan didalam elips tersebut.	
Actor	*Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan *input* atau memberikan *output*, maka aplikasi tersebut juga bisa dianggap sebagai *actor*.	
Association	Asosiasi digunakan untuk menghubungkan *actor* dengan *use case*. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara *Actor* dengan *Use Case*.	

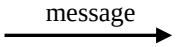
2. *Activity* Diagram

Tabel II.2 Komponen *Activity* Diagram

Simbol	Keterangan
	Titik awal
	Titik akhir
	*Activity*
	Pilihan untuk mengambil keputusan
	Fork; Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Rake*; Menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (*Flow Final*)

3. *Sequence* Diagram

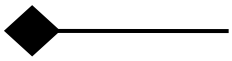
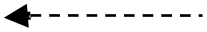
Tabel II.3 Komponen *Sequence* Diagram

Nama Komponen	Keterangan	Simbol
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus *vertikal* yang ditarik dari sebuah *object* .	
Activation	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
Message	*Message*, digambarkan dengan anak panah horizontal antara *Activation* *Message* mengindikasikan komunikasi antara *object -object* .	

4. Class Diagram

Tabel II.4 Komponen *Class* Diagram

Nama Komponen	Keterangan	Simbol
Class	*Class* adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian. Bagian atas adalah bagian nama dari class. Bagian tengah mendefinisikan *property*/*atribut class*. Bagian akhir mendefinisikan *method-method* dari sebuah *class*.	Nama *Class* +atribut +atribut +atribut +*method* +*method*
Association	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa	

	melambangkan *tipe-tipe relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*. (Contoh: *One-to-one*, *one-to-many*, *many-to-many*).	1..n owned by 1
Composition	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/*solid*.	
Dependency	Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.	
Aggregation	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi.