BAB 2
TINJAUAN PUSTAKA

2.1 Ruang Lingkup Objek Penelitian

Objek penelitian yang sedang diteliti adalah Kopi panggang yang bertempat di jalan Ir.H.Djuanda No.391 Bandung.

2.1.1 Profil Kopi Panggang

Kopi panggang didirikan 10 Maret 2013 di Bandung. Kopi panggang berkembang menjadi salah satu coffee shop atau kedai kopi yang terpopuler khususnya di kalangan anak muda untuk ngopi, ngemil dan nongkrong. Dengan segmentasi market anak muda, mahasiswa, serta professional muda dengan kebutuhan bersosialisasi yang relatif tinggi Dengan konsep penyajian kopi fresh bean to the cup, biji kopi di giling setelah konsumen memesan menu agar menghasilkan kualitas kopi yang mantap dan nikmat. Dan menggunakkan bahan baku utama biji kopi arabika pilihan dan di olah melalui prose roasing denga profil roasting khas Kopi panggang.

2.1.2 Logo Kopi Panggang

Berikut ini merupakan logo dari Kopi Panggang pada Gambar 2. 1 Logo Kopi Panggang:



Gambar 2. 1 Logo Kopi Panggang

2.1.3 Visi dan Misi

Berikut ini adalah visi dan misi Kopi Panggang yaitu :

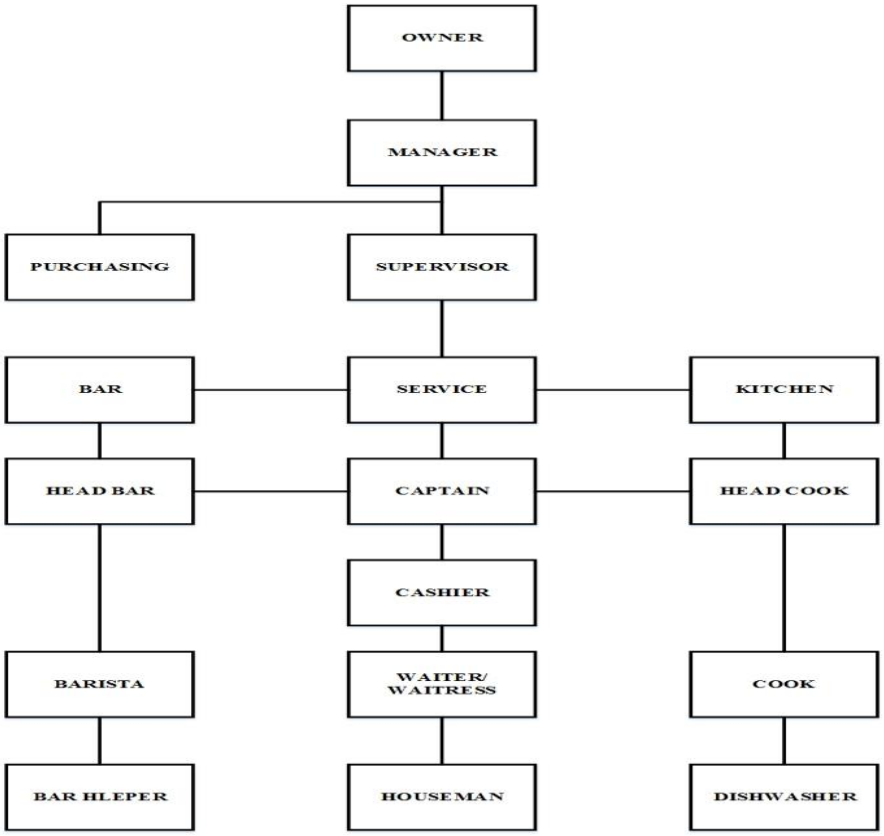
1. Visi

   Menjadi kedai kopi generasi muda nomor satu di Indonesia yang menyajikan produk-produk berkuliatas serta pelayanan yang terbaik.

2. Misi
   a. Pengembangan Mutu Produk dan Pelayanan
   b. Peningkatan Kualitas Sumber Daya Manusia
   c. Perbaikan Yang berkesinambungan

2.1.4 Struktur Organisasi

Struktur organisasi di tunjukkan Gambar 2. 2 Struktur Organisasi di bawah ini:



Gambar 2. 2 Struktur Organisasi

2.1.5 Deskripsi Tugas

Deskripsi tugas akan dijelaskan di bawah ini:

1. Owner adalah pemilik perusahaan.
2. Manager adalah orang yang bertanggung jawab mengatur sebuah perusahaan.
3. Supervisor adalah orang yang bertanggung jawab menjalankan dan mengatur operasional perusahaan.
4. Purchasing adalah orang yang bertanggung jawab atas pembelian keperluan perusahaan.
5. Head Of Department (Service, Kitchen, dan Bar) adalah orang yang bertanggung jawab menjalankan tugas dari atasan (supervisor) dan mengatur bawahan setiap departemen masing-masing.
6. Cashier adalah orang yang bertanggung jawab atas segala transaksi café.
7. Waiter/waitress adalah orang yang bertanggung jawab atas pelayanan tamu café.
8. Cook adalah orang yang bertanggung jawab atas pembuatan produk makanan
9. Barista adalah orang yang bertanggung jawab atas pembuatan produk minuman.
10. Houseman adalah orang yang bertanggung jawab atas kebersihan area atau lingkungan café.
11. Dishwasher adalah orang yang bertanggung jawab atas kebersihan area atau lingkungan dapur.
12. Bar Helper adalah orang yang bertanggung jawab atas kebersihan area atau lingkungan bar.

2.2 Landasan Teori

2.2.1 Restoran

Menurut Marsum W.A (2005) definisi restoran adalah temapat arau bangunan yang terorganisasikan secara komersial yang menyelenggarakan pelayanan dengan baik kepada semuatamu, baik berupa kegiatan makan maupun minuman.Secara Umum restoran temapta yang di kunjungi orang untuk mencari macam makanan dan miuman.Restoran Biasanya juga menyuguhkan keunikan tersendiri sebagai daya tariknya , baik melalui menu masakan, hiburan maupun tampilan fisik bangunan.

2.2.2 Café

Menurut Marsum (2005) Kafe adalah tempat untuk makan dan minum sajian cepat saji dan menyuguhkan suasana santai atau tidak resmi, selain iitu juga merupakan suatu tipe dari restoran yang biasanya menyediakan tempat duduk di dalam dan di luar restoran. Kebanyakan kafe tidak menyajikan makanan berat namun lebih berfokus pada menu makanan ringan seprti kue, roti, sup dan minuman.Kafe pertama kali muncul di daerah barat.

2.2.3 Data Survey

Data survei yang digunakan pada penelitian ini menggunakan wawancara. Wawancara dilakukan dengan narasumber dengan bapak Irvan selaku Supervisor Pada Kopi Panggang yang terdapat beberapa pertanyaan namun yang di ambil hanya 5 pertanyaan mengenai proses pemesanan dan pembayaran pada kopi panggang.Detail wawancara dapat dilihat pada lampiran C.

2.2.4 Konsep Rekayasa Perangkat Lunak

Perangkat lunak merupakan program-program komputer dan dokumentasi yang berkaitan dengan program tersebut. Sedangkan definisi dari rekayasa perangkat lunak sendiri yaitu disiplin ilmu rekayasa atau teknik yang berkaitan

dengan semua aspek dalam membuat perangkat lunak. Perangkat lunak memiliki beberapa karakteristik diantaranya [3]:

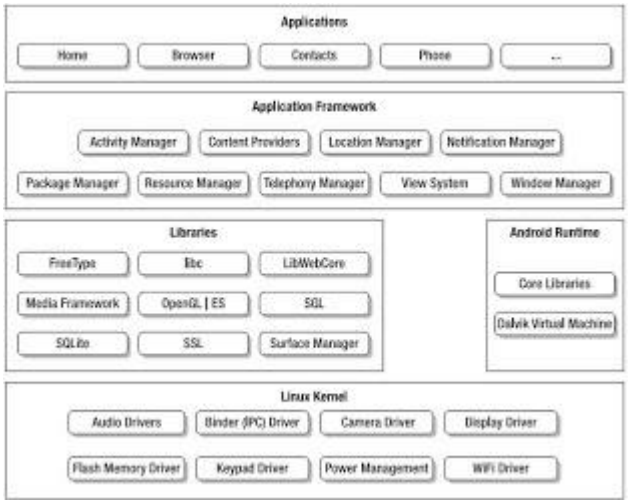
1. *Maintanability* (dapat dirawat) yaitu perangkat lunak harus dapat memenuhi perubahan kebutuhan.
2. *Dependability* yaitu perangkat lunak harus dapat dipercaya.
3. *Efisiensi* yaitu perangkat lunak harus efisien dalam penggunaan *resource*.
4. *Usability* yaitu perangkat lunak harus digunakan sesuai dengan apa yang direncanakan.

2.2.5 Android

Android merupakan sistem operasi *mobile* yang tumbuh di tengah sistem operasi lainnya yang berkembang dewasa ini. Sistem operasi lainnya seperti *Windows Mobile, i-Phone OS, Symbian* dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat hardware yang ada. Akan tetapi, sistem operasi yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga. Oleh karena itu, adanya keterbatasan distribusi aplikasi pihak ketiga untuk *platform* mereka.Android menawarkan sebuah lingkungan yang berbeda untuk pengembang. Setiap aplikasi memiliki tingkatan yang sama. Android tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. *Application Programming Interface* (API) yang disediakan menawarkan akses ke *hardware*, maupun data-data ponsel sekalipun, atau data sistem sendiri. Bahkan pengguna dapat menghapus aplikasi inti dan menggantikannya dengan aplikasi pihak ketiga. [4]

2.2.5.1 Arsitektur Android

Arsitektur Android dapat digambarkan seperti pada Gambar 2.3 Arsitektur Android dan secara garis besar Arsitektur Android dapat dijelaskan sebagai berikut:



Gambar 2. 3 Arsitektur Android [5]

1. *Application* dan *Widget*

Application dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di *layer* terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Hampir semua aplikasi ditulis menggunakan bahasa pemrograman *Java* [6].

2. *Application Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambah status notifikasi, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga bisa kita simpulkan *Application Frameworks* ini adalah layer dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content provider* yang berupa sms dan panggilan telepon [6].

Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

a. *Views*
b. *Content Provider*
c. *Resource Manager*
d. *Notification Manager*
e. *Activity Manager*

3. *Libraries*

Libraries ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya. Berjalan di atas *Kernel*, *layer* ini meliputi berbagai *library* C/C++ inti seperti Libc SSL, serta:

a. *Libraries* media untuk pemutaran media audio dan video.
b. *Libraries* untuk manajemen tampilan.
c. *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D.
d. *Libraries* SQLite untuk dukungan *database.*
e. *Libraries* SSL dan WebKit terintegrasi dengan *web browser* dan *security.*
f. *Libraries* LiveWebcore mencakup *modern web browser* dengan *engine embedded web view*.
g. *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's.

4. Android *Run Time*

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

a. *Core Libraries*

   Aplikasi Android dibangun dalam bahasa *Java*, sementara Dalvik sebagai *virtual* mesinnya bukan *Virtual Machine Java*, sehingga diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa *Java*/C yang ditangani oleh *Core Libraries*.

b. *Dalvik Virtual Machine*

Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux Kernel untuk melakukan *threading* dan manajemen tingkat rendah.

5. *Linux Kernel*

Linux Kernel adalah *layer* dimana inti dari sistem operasi Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya. *Linux Kernel* yang digunakan Android adalah *Linux Kernel release* 2.6.

Keunikan dari nama sistem operasi (OS) Android adalah dengan menggunakan nama makanan hidangan penutup (*Desser*). Selain itu juga nama-nama OS Android memiliki huruf awal berurutan sesuai abjad. Adapun beberapa nama dan versi Android yang sudah diluncurkan adalah sebagai berikut [4]:

a. Android versi Beta

Pertama kali dirilis pada 5 November 2007, kemudian 12 November 2007 *Software Development Kit* (SDK) dirilis oleh Google.

b. Android versi 1.0 Astro

Pertama kali dirilis pada 23 September 2008. Sebenarnya Android versi pertama ini akan dinamai dengan nama "Astro" tapi karena alas an hak cipta dan *trademark* nama "Astro" tidak jadi disematkan pada versi pertama dari OS Android ini. HTC Dream adalah ponsel pertama yang menggunakan OS ini.

c. Android versi 1.1 Bender

Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam *alarm*, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

d. Android versi 1.5 Cupcake

Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (Software Development Kit) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video

dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

e. Android versi 1.6 Donut

Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus; kamera, *camcorder* dan galeri yang dintegrasikan; CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-speech engine*; kemampuan dial kontak; *teknologi text to change speech* (tidak tersedia pada semua ponsel; pengadaan resolusi VWGA.

f. Android versi 2.0/2.1 Éclair

Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (Eclair), perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1. Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi *mobile* terbaik (killer apps – aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah *handset* Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds, dan WeatherBug. Sistem operasi Android dalam situs Internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh MySpace dan Facebook.

g. Android versi 2.2 Froyo: Frozen Youghurt

Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi *Hotspot portabel*, dan kemampuan *auto update* dalam aplikasi Android Market.

h. Android versi 2.3 Gingerbread

Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek *audio* baru (*reverb*, *equalization*, *headphone virtualization*, dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

i. Android versi 3.0/3.1 Honeycomb

Pertama kali diperkenalkan pada 22 Februari 2011 dan Motorola Xoom adalah yang pertama kali menggunakannya. Android versi ini merupakan OS yang didesain khusus untuk mengoptimalkan penggunaan pada tablet PC.

j. Android versi 4.0 ICS (Ice Cream Sandwich)

Pertama kali dirilis pada 19 Oktober 2011. *Smartphone* ini yang pertama kali menggunakan OS Android ini adalah Samsung Galaxy Nexus. Secara teori semua perangkat seluler yang menggunakan versi Android sebelumnya, Gingerbread, dapat diupdate ke Android Ice Cream Sandwich.

k. Android versi 4.1 Jelly Bean

Android Jelly Bean diluncurkan pada cara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Tak ketinggalan *Google Now* juga menjadi bagian yang diperbaharui. *Google Now* memberikan informasi

yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android ini muncul pertama kali dalam produk *tablet* Asus, yakni Google Nexus.

2.2.5.2 Android *Life Cycle*

Apikasi Android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan *alarm*, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen Android yang berbeda seperti ditunjukkan pada tabel 2.1, klasifikasi tersebut berdasarkan kelas-kelas dasar *Java* yang digunakan [4].

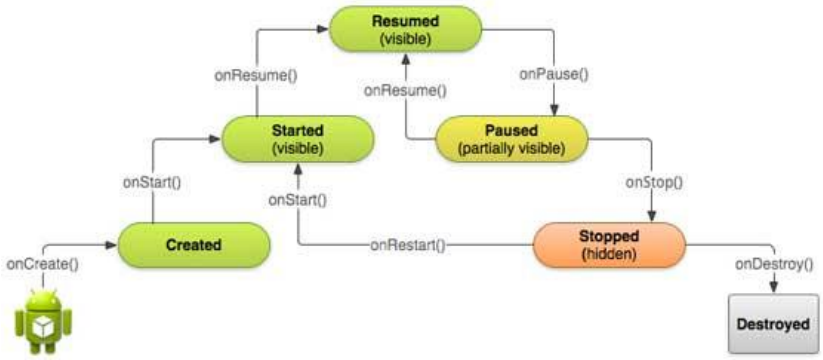
Tabel 2. 1 Komponen Aplikasi Android

Functionality	Java Base Class	Example
Focused thing a user can do	Activity	Edit a note, play a game
Background process	Service	Play music, update weather icon
Receive messages	BroadcastReceiver	Trigger alarm upon event
Store and retrive data	ContentProvider	Open a phone contact

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen *Service, BroadcastReceiver, ContentProvider* [4].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa dimana aplikasi yang dijalankan pada fungsi main(), sistem Android menjalankan kode dalam *method Activity* dengan menerapkan metode *callback* tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi Android memiliki siklus hidup yang berbeda dengan aplikasi *dekstop* atau web. Hal ini dikarenakan aplikasi *mobile* memiliki

tingkat *interupsi* proses yang lumayan tinggi seperti ketika *handling* panggilan masuk aplikasi diharuskan menghentikan proses sementara. Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna [7].



Gambar 2. 4 Siklus Hidup Android [7]

2.2.5.3 Fitur

Android memiliki beberapa fitur utama yang sering digunakan dalam proses pembangunan aplikasi diantaranya adalah:

1. *Multi*-proses dan *App Widget*

Sistem operasi Android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi Android dapat mengatur aplikasi dan *thread* yang berjalan secara *multitasking.* Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di layer depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan *game,* proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan [7].

App Widget adalah mini aplikasi yang dapat *embedded* dalam aplikasi seperti *home screen*. *App widget* dapat menjalankan proses *request* seperti musik *streaming* atau mendeteksi suhu ruangan secara *background* [7].

Multi-proses dapat memberikan manfaat berupa *user experience* yang lebih banyak, namun pengguna fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [7].

2. *Touch Gestures* dan *Multi-touch*

Touchscreen adalah *user interface* intuitif yang digunakan banyak *smartphone* di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. *Multi-touch* adalah kemampuan yang digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [7].

3. Hard dan Soft Keyboard

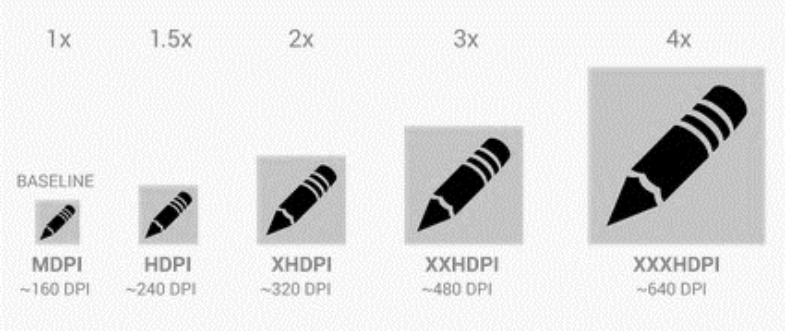
Salah satu fitur pada perangkat *smartphone* adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoprasian Android. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang diuat oleh sistem operasi seperti *keyboard* virtual, dan tombol navigasi aplikasi [7].

2.2.5.4 Prinsip Desain

Android memiliki beberapa prinsip desain yang dapat menjadi acuan dalam membuat desain aplikasi Android diantaranya adalah:

1. *Multiple Assets*

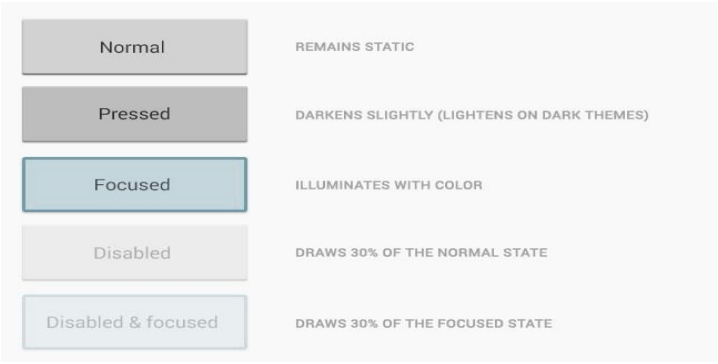
Android mendukung jutaan *smartphone*, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi *fragmentasi* pada Android. Seperti ilustrasi pada Gambar 2.3 Klasifikasi Ukuran Ikon, Android menciptakan beberapa klasifikasi ukuran *icon* yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada *device* berukuran *smartphone* sedangkan untuk XHDPI, XXHDP dan XXXHDPI digunakan pada *device* berukuran *tablet* [8].



Gambar 2. 5 Klasifikasi Ukuran Ikon

2. *Touch Feedback*

Touch Feedback dalam Android digunakan sebagai respon setiap objek yang ditekan pengguna seperti yang diilustrasikan pada Gambar 2.4. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [8].



Gambar 2. 6 Touch Feedback

3. *Pattern Gesture*

Touch gesture berguna untuk meningkatkan *experience* pengguna dalam menggunakan aplikasi. Terdapat beberapa *gesture* yang didukung oleh Android diantaranya adalah [9]:

a. *Touch*

Touch adalah *gesture default* sebagai *event* untuk item tertentu yang paling sering digunakan dalam perangkat Android, hampir seluruh kontrol utama aplikasi ini menggunakan *touch*.

b. *Long Press*

Biasanya digunakan untuk seleksi data, dengan *gesture* ini dimungkinkan untuk memilih satu atau lebih item dalam sebuah tampilan dan menjalankan suatu fungsionalitas tertentu.

c. *Swipe or Drag*

Swipe adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. *Swipe* dapat dilakukan dari dan ke arah mana saja.

d. *Double Touch*

Pada *smartphone* dan tablet Android, melakukan dua kali tapping secara berturut-turut pada satu objek, fungsinya berbeda dengan double klik mouse komputer. Pada Android, teknik ini biasanya dipakai untuk melakukan *zoom in* atau memperbesar dan *zoom out* atau memperkecil sebuah objek gambar.

e. *Pinch Open*

Teknik lain yang biasa digunakan adalah dengan menggunakan dua jari, dimana kedua jari tersebut menyentuh dua titik pada layar yang terpisah dimana ujung dari dua jari terseut tidak bersentuhan, kemudian kedua jari tersebut sambil tetap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu.

f. *Pinch Close*

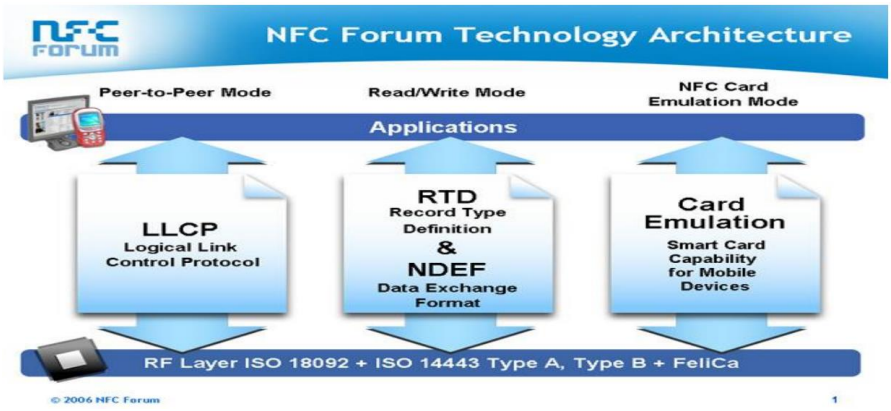
Teknik *Pinch Close* adalah kebalikan dari *Pinch Open* dimana *spread* dilakukan dengan berawal dua jari bersentuhan pada ujungnya dan ditempelkan sebuah titik yang sama pada layar, kemudian kedua jari tersebut bergerak memisahkan atau menjauhi satu sama lain. Gerakan ini untuk menutup layar.

2.2.6 NFC

Near Field Communication (NFC) merupakan teknologi komunikasi baru dengan menggunakan induksi magnet berbasis teknologi *Radio Frequency Identification* (RFID). NFC mulai dikembangkan pada tahun 2002 oleh NXP dan Sony. Mereka berhasil membangun Forum NFC pertama bersama dengan Nokia. Sejak saat itu Forum NFC memiliki lebih dari 150 anggota dan peneliti yang terlibat

dari beberapa perusahaan dan organisasi. Pada forum ini juga NFC dipromosikan dan secara resmi mengeluarkan sertifikasi untuk perangkatperangkat yang mengadopsi teknologi NFC. NFC bisa juga disebut sebagai generasi kedua dari teknologi RFID.

NFC beroperasi pada frekuensi 13,56 MHz dengan kecepatan transmisi pengiriman mencapai 424 kbit/s. Jarak transmisi NFC sekitar 4-10 cm. Perbedaan antara NFC dan teknologi komunikasi *contactless* lainnya yaitu perangkat NFC dapat bersifat aktif – aktif (*peer to peer*) dan aktif – pasif. Oleh karena itu NFC selalu melibatkan inisiator (*reader*) dan target. Inisiator aktif menghasilkan medan RF (*Radio Frequency*) yang dapat memberikan kekuatan ke target yang pasif (tidak memiliki sumber daya). Hal ini memungkinkan target NFC untuk memiliki bentuk yang sangat sederhana seperti stiker, gantungan kunci, atau kartu yang tidak memerlukan energy khusus [10].



Gambar 2. 7 Arsitektur NFC [10]

Banyak manfaat kegiatan-kegiatan yang diperoleh dengan memanfaatkan tekonolgi NFC. Seiiring perkembangan jaman, inovasi-inovasi terhadap pemanfaatan NFC ini juga ikut berkembang. Perkembangan inovasi-inovasi tersebut terbagi ke dalam 3 kategori:

1. *Service Initiation*

Pada kategori ini NFC berguna untuk melakukan layanan sebagai media untuk mendapatkan informasi tertentu atau dapat juga untuk membuka layanan yang

mengharuskan autentikasi data terlebih dahulu. Hal ini melibatkan NFV *Tag* sebgai media penyimpanan maupun penyampaian sementara. Pada *Tag* tersebit dapat berisi informasi tejs, alamat sebuah website (*url*), nomor telepon atau data lain dengan ukuran yang kecil.

Salah satu contoh penggunaannya yaitu pada poster pintar (*smart poster*). Dengan mendekatkan ponsel kepada NF *Tag* pada poster, maka pengguna akan langsung mendapatkan informasi lebih detail pada yang belum tertera pada poster tersebut tanpa harus menginputkan apapun pada ponselnya.

2. *Peer-to-peer*

Pada kategori ini NFC berperan dalam mengkomunikasikan antara dua perangkat aktif dengan ukuran data yang relative kecil. Namun untuk kebutuhan data yagn berukuran besar, NFC dapat digunakan sebagai sambungan nirkael seperti *Bluetooth* dan WiFi.

Contoh dari penggunaannya yaitu pada proses pencetakan foto langsung dari ponsel atau kamera. Pengguna tinggal memilih foto yang akan dicetak, kemudian mendekatkan ponselnya pada printer yang sdah terintregrasi dengan teknologi NFC untuk mebuat sambungan *Bluetooth* untuk mengirimkan gambar dan mencetaknya.

3. *Payment*

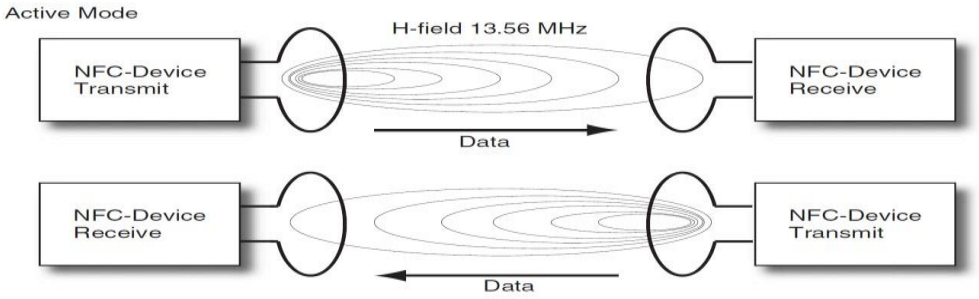
Pada kategori ini NFC berperan dalam proses pembayaran suatu barang yang sudah terintegrasi sebelumnya dengan kartu kredit, bank dan operator jaringan lainnya. Dengan memanfaatkan teknologi NFC dalam pembayaran ini dapat mempermudah dan mempercepat proses transaksi pembayaran karena tidak menggunakan uang tunai maupun metode pembayaran konvensional lainnya. Contoh dari penggunaanya yaitu pada proses transaksi pembayaran di restoran, pusat perbelanjaan dan area parkir.

Komunikasi NFC dapat dilakukan dalam 2 mode yaitu mode aktif dan mode pasif.

1. Mode Aktif

Pada mode aktif, NFC yang berperan sebagai initiator mengaktifkan transmiter yang dimiliki sehingga arus dengan frekuensi tinggi yang mengalir pada antena

menimbulkan medan magnetik H yang menyebar ke sekitarnya seperti terlihat pada Gambar 2.8:



Gambar 2. 8 Mode Aktif [10]

Sebagian akan bergerak dan tertangkap oleh antena interface NFC yang terletak pada lokasi berdekatan. Kemudian tegangan U dihasilkan di antena yang dapat dideteksi dan diterima oleh receiver pada NFC target.

Cara kerja tersebut mirip dengan konsep RFID Reader dan transponder. Namun, terdapat perbedaan dimana pada transponder RFID reader harus memberikan suplai energi kepada transponder untuk mengoperasikan microchip untuk komunikasi data. Sementara pada NFC, perangkat elektronik yang memiliki fitur NFC yang memberikan suplai energi.

Pada mode aktif, pengiriman data dilakukan dari NFC target kepada NFC initiator dengan NFC target mengaktifkan transmitter dan NFC initiator berpindah mengaktifkan receiver. Sehingga pada mode ini kedua interface NFC bergantian menghasilkan medan magnetik.

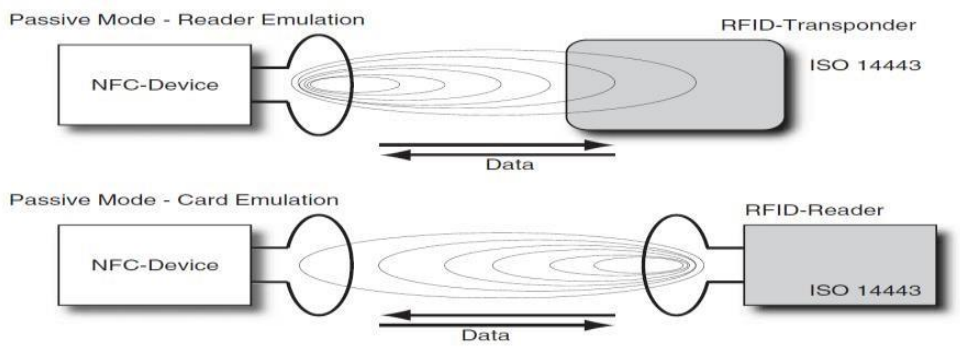
2. Mode Pasif

Pada mode pasif, NFC inisiator juga menghasilkan medan magnetik untuk memulai komunikasi dengan NFC target. Namun setelah paket data sampai ke target, NFC inisiator tidak mengganti mode menjadi penerima seperti NFC mode aktif, melainkan tetap mengaktifkan transmitter. Sehingga trasmitter hanya diaktifkan pada NFC inisiator.

Kemudian data pada NFC target akan “ditumpangkan” pada gelombang tersebut dengan menghasilkan yang disebut sebagai *load modulation*. Hal ini

memberikan keuntungan dalam pengoperasiannya pada kondisi daya yang dimiliki oleh NFC target sedang lemah.

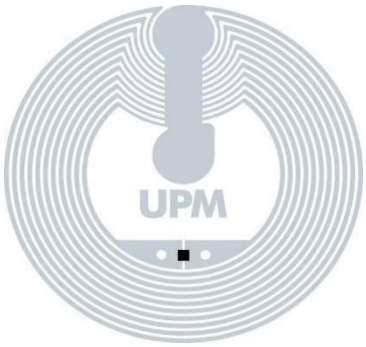
Hal ini juga memungkinkan *interface* NFC berfungsi seperti RFID *Reader* untuk berkomunikasi dengan transponder pasif dengan memberikan suplai daya pada transponder dan menggunakan *load modulation* data ditransimisikan pada interface NFC. Misalnya, *smartphone* dengan fitur NFC dapat digunakan untuk membaca dan menulis pada transponder seperti *smart label* atau *e-ticket*. Aktivitas *interface* NFC yang menyerupai RFID *Reader* disebut juga '*reader-emulation mode*'.



Gambar 2. 9 Mode Pasif [10]

2.2.6.1 NFC *Tag*

NFC *Tag* merupakan media pendukung teknologi NFC. NFC Tag biasanya bersifat pasif, sehingga membutuhkan inisiator (NFC bersifat aktif) untuk proses komunikasi. NFC *Tag* dapat berisi data dengan ukuran kecil. NFC *Tag* saat ini memiliki memory berkisar antara 96 dan 4.096 byte. NFC *Tag* berisfat *read-only* tetapi memungkinkan juga bersifat *re-writeable* [11].



Gambar 2. 10 NFC Tag [11]

Produsen NFC telah menentukan standar dari NFC *Tag* untuk dapat dioperasikan dengan perangkat NFC. Spesifikasi yang ditentukan terbagi menjadi 4 tipe. Keempat tipe *Tag* NFC tersebut didasarkan pada produk dan ketersediaan komersil. Berikut adalah 4 tipe dari NFC *Tag*:

1. Tipe 1 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 96 bytes dan dapat di-*upgrade* hingga 2 Kbyte.
2. Tipe 2 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 48 bytes dan dapat di upgrade hingga 2 Kbyte.
3. Tipe 3 : didasari pada standar industri Jepang atau *Japanese industrial standard* (JIS) X 6319-4, atau disebut Felica. *Tag* diatur pada pembuatannya untuk mendukung baik dalam *read* dan *re-writeable*, ataupun *read-only*. Memori yang tersedia dapat diubah, secara teori batas memorinya adalah 1 Mbyte per *service*.
4. Tipe 4 : kompatibel sepenuhnya dengan ISO/IEC 14443 standard *series*, memiliki memori 32 KB dan lebih cepat daripada tipe *tag* lainnya, memiliki proteksi terhadap data *collision*, dapat menggunakan komunikasi NFC-A maupun NFC-B. Tipe 4 dapat ditulis ulang *(rewriteabl*e) atau *read-only*. Tetapi pengaturan tersebut dilakukan saat tipe 4 dalam tahap pembuatan. Saat sudah selesai dibuat, pengaturannya tidak dapat diubah [11].

2.2.6.2 NFC *Reader*

NFC reader adalah perangkat yang menerima atau mengirimkan perintah dari NFC. NFC reader akan sangat berguna bagi pedagang yang ingin menyediakan layanan pembayaran menggunakan NFC [12].



Gambar 2. 11 Reader NFC (Near Field Communication) [12]

2.2.6.3 Tabel URI (Uniform Resource Identifier)

Untuk mempersingkat URI, byte pertama dari data record menggambarkan protokol dari URI. Tabel tersebut harus digunakan untuk proses encoding dan decoding URI, meskipun aplikasi mungkin menggunakan nilai 0x00 untuk menunjukkan tidak adanya awalan saat encoding, terlepas dari adanya kode singkatan yang sesuai. Untuk penjelasan dari protokol yang berbeda, silahkan lihat tabel URI berikut:

Tabel 2. 2 Format URI Forum NFC

Value	Protocol
0x00	URI *Field*
0x01	*http://www.*

0x02	*https://www.*
0x03	*http://*
0x04	*https://*
0x05	*Tel:*
0x06	*mailto:*
0x08	*ftp://ftp.*
0x09	*ftps://*
0x0A	*sftp://*
0x0B	*smb://*
0x0C	*nfs://*
0x0D	*ftp://*
0x0E	*dav://*
0x0F	*news:*
0x10	*telnet://*
0x11	*imap:*
0x12	*rtsp://*
0x13	*urn:*
0x14	*pop:*
0x15	*sip:*
0x16	*sips:*
0x17	*tftp:*
0x18	*btspp://*
0x19	*btl2cap://*
0x1A	*btgoep://*
0x1B	*tcpobex://*
0x1C	*irdaobex://*
0x1D	*file://*
0x1E	*urn:epc:id:*
0x1F	*urn:epc:tag:*

0x20	*urn:epc:pat:*
0x21	*urn:epc:raw:*
0x22	*urn:epc:*
0x23	*urn:nfc:*

Sebagai contoh, jika isi baris dari tabel URI adalah 0x02, dan datanya berisi "nfcforum.org", maka yang dihasilkan adalah"https://www.nfcforum.org". Jika isi baris dari tabel URI adalah nol (0x00), maka awalan akan dilakukan. Semua kolom yang ditandai RFU akan diperlakukan seolah-olah bernilai nol (tidak ada awalan). Sebuah sistem yang kompatibel tidak harus menghasilkan nilai-nilai yang ditandai dengan RFU.

1. URI *Field*

Kolom URI ini menyediakan URI sesuai RFC 3987 [RFC 3987] (yang sebenarnya merupakan IRI, atau internasionalisasi Resource Identifier, tetapi sejak dahulu sudah ditetapkan dengan kata URI). IRI ini bisa menjadi URL atau URN seperti yang dijelaskan sebelumnya. Pengkodean yang digunakan harus UTF-8, kecuali skema URI menentukan beberapa encoding tertentu.

Panjang IRI dapat dihitung dengan mengambil panjang payload(muatan), dan mengurangkan 1 untuk protokol kode singkatan byte. Ini merupakan panjang dalam satuan byte, Ini merupakan panjang dalam satuan byte, tidak dalam karakter (seperti UTF-8 karakter yang dapat menempati lebih dari satu byte).

URI didefinisikan hanya dalam 7-bit ruang US-ASCII. Oleh karena itu, aplikasi yang kompatibel harus mengubah UTF-8 IRI string ke bentuk 7-bit USASCII string dengan mengubah poin kode di atas 127 ke pengkodean yang tepat. Kode ini telah didefinisikan dalam RFC 3987 [RFC 3987] dan IDN [RFC 3492] dokumen. Untuk skema yang berbeda, pengkodean juga mungkin berbeda. Misalnya, jika URI (setelah awalan dari jenis bidang URI) mengandung string berikut: "Http: //www.hääyö.com/", itu diubah, sesuai standar IDN [RFC 3492] aturan, menjadi "http://www.xn--hy-viaa5g.com" sebelum melakukan proses

selanjutnya. Kebanyakan aplikasi modern yang sudah mendukung skema internasionalisasi Resource Identifier (IRI) yg baru. Hal ini direkomendasikan untuk implementasi yang mendukung IRI dimana tampilan dari form URI dapat dibaca manusia dengan mudah.

Untuk memperjelas, URI mungkin mengandung karakter UTF-8. Namun, Internet tidak bisa menangani kaarakter tersebut, dan oleh karena itu URI perlu diubah sebelum digunakan. Untuk sebagian besar perangkat, konversi ini dilakukan oleh aplikasi.

Setiap nilai karakter dalam URI antara (dan termasuk) 0 dan 31 akan dicatat sebagai kesalahan (error), dan *record* URI akan dibuang. Setiap data yang tidak sesuai dengan urutan UTF-8 akan dianggap kesalahan, dan seluruh catatan URI akan dibuang. 4 Handling Guideline / 4 Pedoman Penanganan URI RTD tidak mendefinisikan aksi khusus jika perangkat sesuai untuk melakukan proses tersebut. Ini merupakan implementasi dari aksi yang akan dilakukan. Berikut contoh tentang cara menggunakan URI RTD dalam aplikasi:

a. Menyimpan URL Sederhana

   Untuk menempatkan URL http://www.nfc.com pada tag menggunakan protokol NDEF, tambahkan urutan byte berikut. Total panjang: 12 bytes.

Tabel 2. 3 Menyimpan Data URL

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), ME=1, MB=1
1	0x01	Length of the Record Type (1 byte)
Offset	**Content**	**Explanation**

2	0x08	Length of the payload (8 bytes)
3	0x55	The URI record type (“U”)
4	0x01	URI identifier (“http://www.”)
5	0x6e 0x66 0x63 0x2e 0x63 0x6f 0x6d	The string “nfc.com” in UTF-8.

b. Menyimpan No Telepon

Untuk menyimpan nomor telepon (misalnya, untuk membuat ponsel melakukan panggilan ke nomor ini), menggunakan urutan.

Contoh nomor: + 358-9-1234567. Total panjang adalah 17 byte.

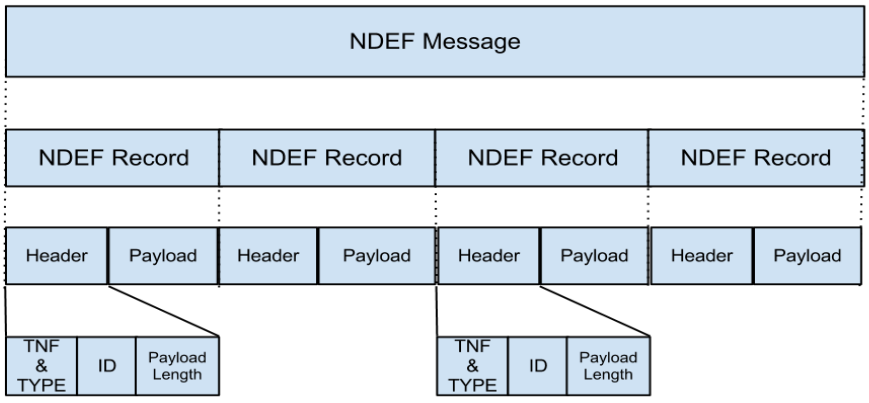
Tabel 2. 4 Menyimpan Data Nomor Telepon

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), ME=1, MB=1
1	0x01	Length of the Record Type (1 byte)
Offset	**Content**	**Explanation**
2	0x0D	Length of the payload (13 bytes)
3	0x55	The URI record Name (“U”)

4	0x05	URI identifier (“tel:”)
5	0x2b 0x33 0x35 0x38 0x39 0x31 0x32 0x33 0x34 0x35 0x36 0x37	The string “+35891234567” in UTF-8.

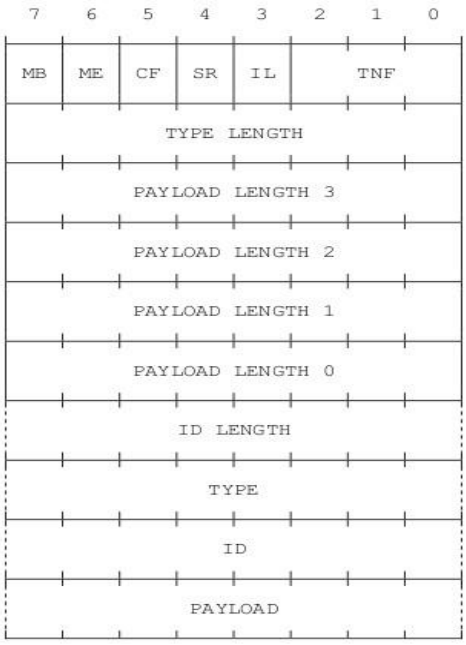
2.2.6.4 NFC *Data Exchange Format* (NDEF)

NFC Data Exchange Format (NDEF) adalah format pesan standar pada proses pertukaran informasi antara perangkat NFC ke perangkat NFC dan perangkat NFC ke *Tag* NFC. Di dalam NDEF terdapat NDEF *Message* yang terdiri dari beberapa *record* seperti pada gambar 2.13 [11].



Gambar 2. 12 Struktur NDEF Message [11]

NDEF *Record* merupakan format umum dari panjang variabel *record.* Berikut adalah struktur NDEF *Record.*



Gambar 2. 13 Struktur NDEF Record [11]

Berikut adalah keterangan dari struktur NDEF *Record*:

1. MB (*Message Begin*) : Area 1 bit. Ketika diset maka akan menunjukan awal NDEF *Message*.
2. ME (*Message End*) : Area 1 bit. Ketika diset maka akan menunjukan akhir NDEF *Message*.
3. CF (*Chunk Flag*) : Area 1 bit. Untuk mengidentifikasi apakah potongan *record* pertama atau potongan tengah *record*.
4. SR (Short Record) : Area 1 bit. Jika diset maka PAYOAD_LENGTH merupaan oktet tunggal. Short Record ini dimaksudakn untuk mengkapsulasi *Payloadi*, dengan kisaran ukuran 0 sampai 255
5. IL (*Identification Length*) : Area 1 bit. Jika diset maka ID_LENGTH akan muncul di bagian *header* sebagai oktet tunggal. Jika ID_LENGTH bernilai 0. ID_LENGTH

dihilangkan dari *header record* dan area ID dihilangkan dari *record.*

6. TNF (*Type Name Format*) : Area bernilai 3 bit. Untuk mengidentifikasi struktur dari *value* TYPE.

Tabel 2. 5 Value Type Name Format

Type Name Format	***Value***
Empty	0x00
NFC Forum well-known type [NFC RTD]	0x01
Media-type as defined in RFC 2046 [RFC 2046]	0x02
Absolute URI as defined in RFC 3986 [RFC 3986]	0x03
NFC Forum external type [NFC RTD]	0x04
Unkown	0x05
Unchanged	0x06
Reserved	0x07

Keterangan tabel 2.5 *Value Type Name Format*:

1. 0x00 (Empty)

   Menunjukan bahwa tidak ada jenis atau payload yang terkait dengan *record*. Ketika digunakan TYPE_LENGTH, ID_LENGTH dan bidang PAYLOAD_LENGTH harus bernilai nol dan TYPE_ID dan PAYLOAD dihilangkan dari *record*. Nilai TNF ini dapat digunakan setiap kali *record* kosong diperlukan, misalnya untuk mengakhiri NDEF *message* ketika tidak ada PAYLOAD yang diinputkan oleh pengguna aplikasi.

2. 0x01 (NFC Forum wel-known type [NFC RTD])

   Menunjukan bahwa TYPE berisi nilai yang mengikuti *Type Name Format* RTD sesuai spesifikasi pada NFC Forum.

3. 0x02 (Absolute URI as defined in RFC 3986 [RFC 3986])

Menunjukan bahwa TYPE berisi nilai yang sesuai dengan yang dibuat oleh BNF dan didefinisikan dalam RFC 2046.

4. 0x03 (Absolute URI as defined in RFC 3986 [RFC 3986])
   Menunjukan bahwa TYPE berisi nilai yang mutlak sesuai dengan URI BNF dan didefinisikan dalam RFC 3986.
5. 0x04 (NFC Forum external type [NFC RTD])
   Menunjukkan bahwa bidang TYPE berisi nilai yang mengikuti *Type Name Format* RTD yang didefinisikan dalam NFC Forum RTD spesifikasi [NFC RTD] untuk nama jenis eksternal.
6. 0x05 (Unkown)
   Menunjukan jika tipe PAYLOAD tidak diketahui. Hal ini serupa dengan "application/octet-stream" jenis media yang didefinisikan oleh MIME RFC 2046. Ketika digunakan TYPE_LENGTH harus bernilai nol. Maka dengan demikian TYPE dihilangkan dari NDEF *record*.
7. 0x06 (Unchanged)
   Hanya untuk *record* tengah dan mengakhiri *record* dari Chunked Payload. TYPE_LENGTH harus bernilai nol.
8. 0x07 (Reserved)
   Milik NFC Forum untuk digunakan di masa depan.
9. TYPE_LENGTH
   TYPE_LENGTH adalah 8bit integer yang menentukan panjang oktet dari TYPE
10. ID_LENGTH
    ID_LENGTH adalah 8bit integer yang menentukan panjang oktet dalam ID. ID_LENGTH muncul apabilan IL diset menjadi 1 di dalam *record header*. ID_LENGTH boleh bernilai 0 oktet namun ID harus dihilangkan dari NDEF *record*.
11. PAYLOAD_LENGTH
    PAYLOAD_LENGTH adalah yang menentukan panjang oktet dari PAYLOAD.

12. TYPE

    Nilai TYPE adalah identifikator yang menjelaskan jenis nilai PAYLOAD. Nilai dari TYPE harus sesuai dengan struktur, encoding dan format pada TNF.

13. ID

    Nilai dari ID adalah identifier dalam bentuk resferensi URI yang dijelaskan dalam RFC 3986. Referensi URI dapat bersifat relative dan absolut karena NDEF tidak mendefiniskan basis URI.

14. PAYLOAD

    Konten yang dapat berisi apa saja dalam ukuran beberapa bytes.

2.2.7 Web Service

Web service Sebuah layanan web adalah standar yang digunakan untuk bertukar informasi antara aplikasi atau sistem jenis *heterogen*. Aplikasi perangkat lunak yang ditulis dalam berbagai bahasa pemrograman dan berjalan pada berbagai platform dapat menggunakan layanan *web* untuk bertukar informasi melalui Internet dengan menggunakan protokol http. Interoperabilitas ini dapat dicapai antara aplikasi Java dan Dot net, atau PHP dan aplikasi Java [13].

Beberapa keuntungan menggunakan *web service* pada Android yaitu [13]:

1. Dengan menambahkan *web service* akan membuat perangkat pengguna bekerja menjadi lebih ringan, baik dari segi daya CPU yang diperlukan dan bandwidth yang digunakan.
2. *Web service* juga bisa dimanfaatkan dengan signifikan dengan menggunakan kembali fungsi yang ada sebagai metode web.
3. *Database* bisa di akses melalui aplikasi Android dari jarak jauh.

2.2.8 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke 3 – Desember 1999. JSON

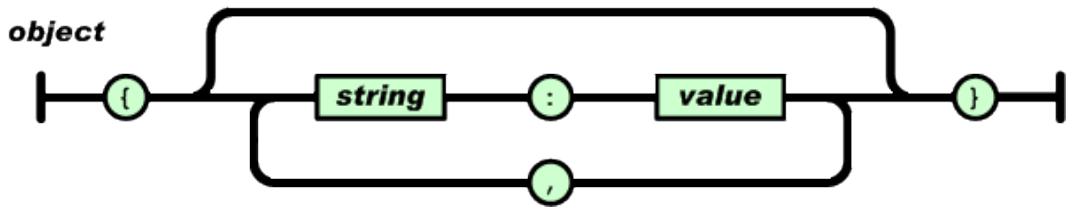
merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), daftar berkunci (*keyed list*), atau *associative* array [14].
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*) [14].

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung sruktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama / nilai yang tidak terurutkan. Seperti yang diilustrasikan pada Gambar 2.14 objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [14].

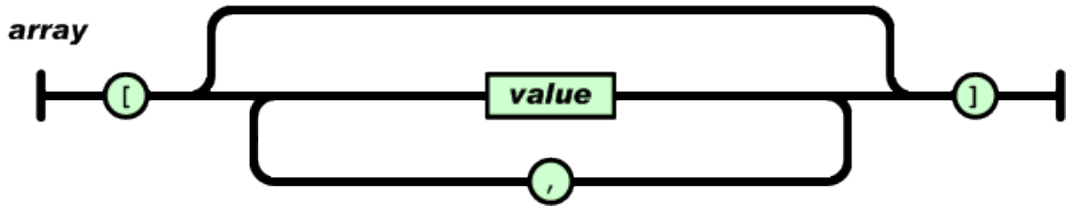


Gambar 2. 14 Objek JSON [14]

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh

, (koma) seperti yang diilustrasikan pada Gambar 2.15. Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat [14].

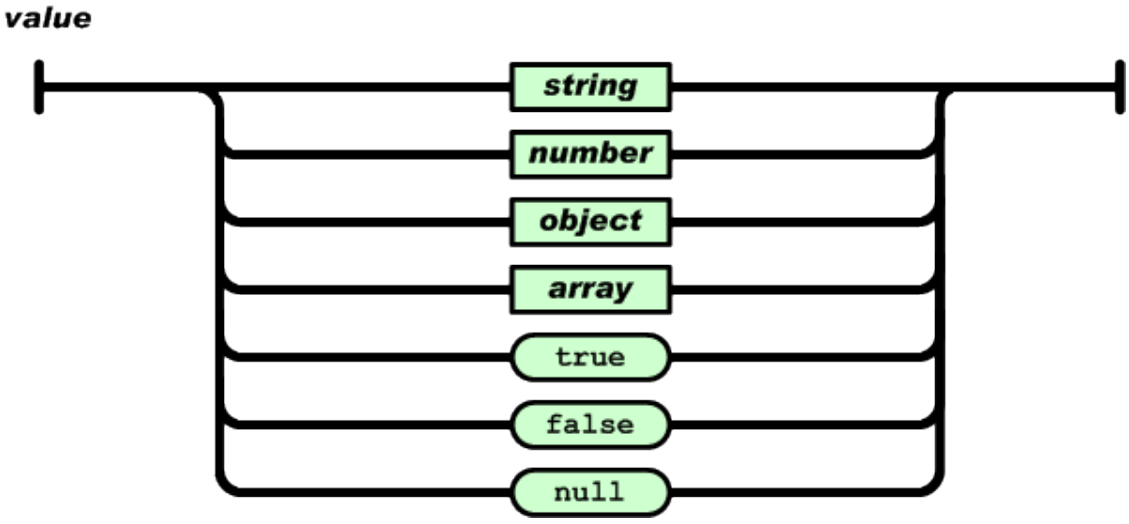


Gambar 2. 15 Array JSON [14]

Bentuk data JSON objek dan larik dapat saling dikombinasikan untuk mendukung struktur data yang lebih kompleks. JSON mendukung beberapa tipe data untuk menjadi value seperti Angka, String, Bollean dan Nilai NULL [14].

3. Nilai

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat [14].

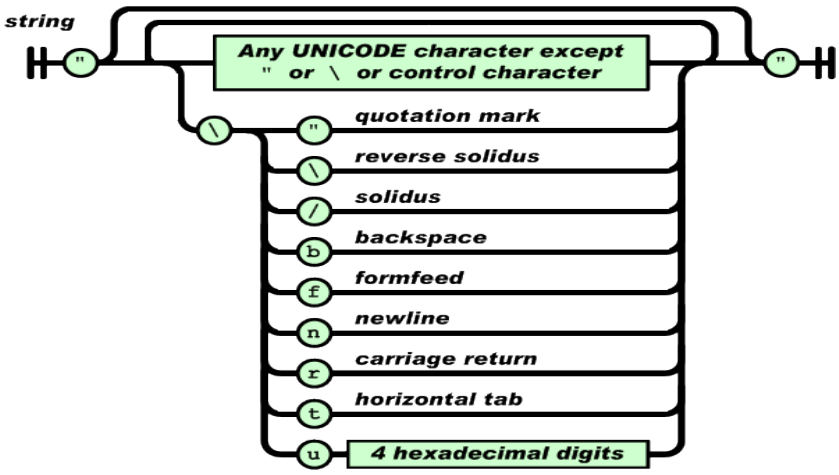


Gambar 2. 16 Value JSON [14]

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash

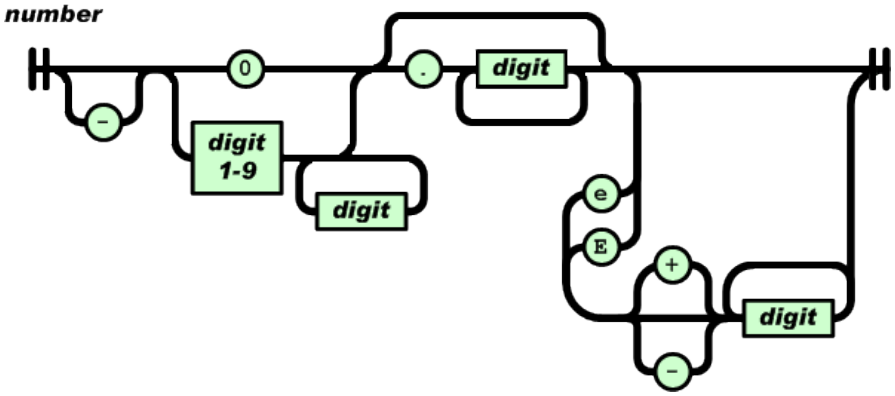
escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java [14].



Gambar 2. 17 String JSON [14]

5. Angka

   Angka, Format oktal dan heksadesimal tidak digunakan [14].



Gambar 2. 18 Number JSON [14]

2.2.9 API

API adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan system operasi. API atau *Application Programming Interface* juga merupakan suatu dokumentasi yang terdiri dari antar muka, fungsi, kelas, struktur untuk membangun sebuah perangkat lunak.

Dengan adanya API, maka memudahkan seorang programmer untuk membongkar suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Suatu rutin standar yang memungkinkan *developer* menggunakan *system function*. Proses ini dikelola melalui *operating system*. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya untuk saling berinteraksi [15].

Keuntungan dengan menggunakan API adalah sebagai berikut:

1. Portabilitas.

   Developer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah terinstal API tersebut.
2. Lebih Mudah Dimengerti

   API menggunakan bahasa yang lebih terstruktur dan mudah dimengerti daripada bahasa *system call*. Hal ini sangat penting dalam hal *editing* dan pengembangan.

System call interface ini berfungsi sebagai penghubung antara API dan *system call* yang dimengerti oleh sistem operasi. *System call interface* ini akan menerjemahkan perintah dalam API dan kemudian akan memanggil *system calls* yang diperlukan. Untuk membuka suatu file tersebut user menggunakan program yang telah dibuat dengan menggunakan bantuan API, maka perintah dari user tersebut diterjemahkan dulu oleh program menjadi perintah open().

Perintah open() ini merupakan perintah dari API dan bukan perintah yang langsung dimengerti oleh kernel sistem operasi. Oleh karena itu, agar keinginan pengguna dapat dimengerti oleh sistem operasi, maka perintah open() tadi diterjemahkan ke dalam bentuk *system call* oleh *system call interface*. Implementasi perintah open() tadi bisa bermacam-macam tergantung dari sistem operasi yang digunakan [16].

Cara menggunakan API:

1. Dilakukan dengan mengimpor package/kelas
2. Ada beberapa kelas bernama sama dipackage yang berbeda, yaitu:
   a. Import salah satu dan gunakan nama lengkap untuk yang lain
   b. Gunakan nama lengkap semua kelas

Kebanyakan Sistem Operasi seperti Windows, menyediakan fasilitas API sehingga programmer dapat melakukan aktivitas programming dengan lebih konsisten. Meskipun API didesain untuk programer, namun API juga baik untuk user karena setidaknya dapat menjamin bahwa program tersebut memiliki *interface* yang sama, sehingga lebih mudah untuk dipelajari.

2.2.10 *Object Oriented Analysis Desain*

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem [17].

2.3 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah termasuk ke dalam rumpun jenis pemodelan notasi grafis yang didukung oleh model-model tunggal. Pemodelan ini berguna untuk membantu dalam menjelaskan data rancang perangkat lunak yang dibangun dengan *object-oriented* (OO). UML merupakan standar terbuka yang dikelola *Open Management Group* (OMG) yang berada dibawah naungan perusahaan-perusahaan konsorium terbuka. UML merupakan suatu bahasa pemodelan yang terdiri banyak model diantaranya adalah [17]:

2.3.1 *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) system secara keseluran yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada diagram ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use case* menjelaskan secara sederhana fungsi sistem dari sudut pandang user. Adapun komponen-komponen dalam *Use Case Diagram* anataranya [17]:

1. *Actor*

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu. Bisa merupakan manusia, sistem, atau *device* yang memiliki peranan dalam keberhasilan operasi dari sistem.

2. *Use Case*

Use case merupakan gambaran umum dari fungsi atau proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *use case*, tetapi yang digambarkan hanya fungsionalitas utama yang berkaitan dengan sistem. *Use case* menitik beratkan bagaimana suatu sistem dapat berinteraksi baik antar sistem maupun diluar sistem.

3. *System*

Menyatakan batasan sistem dalam relasi dengan *actor-actor* yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). Digambarkan dengan segi empat yang membatasi semua *use case* dalam sistem terhadap pihak mana sistem akan berinteraksi. Sistem disertai label yang menyebutkan nama dari sistem, tapi umumnya tidak digambarkan karena tidak terlalu memberi arti tambahan pada diagram.

4. *Association*

Mengidentifikasikan interaksi antara setiap *actor* tertentu dengan setiap *use case* tertentu. Digambarkan sebagai garis antara *actor* terhadap *use case* yang

bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian.

5. *Dependency*

   Dependensi <<include>>

   i. Mengidentifikasi hubungan antar dua *use case* dimana yang satu memanggil yang lain.
   ii. Jika pada beberapa *use case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan *use case* tersendiri dengan relasi dependensi setiap *use case* semula ke *use case* yang baru ini sehingga memudahkan pemeliharaan.
   iii. Digambarkan dengan garis putus-putus bermata panah dengan notasi <<include>> pada garis.
   iv. Arah mata panah sesuai dengan arah pemanggilan

   Dependensi <<extend>>

   i. Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<extend>>.
   ii. Digambarkan serupa dengan dependensi <<include>> kecuali arah panah berlawanan.
   iii. Note: konsep "*extend*" ini berbeda dengan "extend" dalam Java.

6. *Generalization*

Mendefinisikan relasi antara dua *actor* atau dua *use case* yang mana salah satunya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya. Penggambaran menggunakan garis bermata panah kosong dari yang meng-*inherit* mengarah ke yang di-*inherit.*

2.3.2 *Activity Diagram*

Activity Diagram adalah sebuah tahapan yang lebih fokus kepada menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dimana biasanya dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis. *Activity Diagram* ini sendiri memiliki struktur diagram yang mirip

Flowchart atau *data flow diagram* pada perancangan terstruktur. *Activity Diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *Use Case Diagram* [17].

2.3.3 *Class Diagram*

Class diagram merupakan diagram yang selalu ada di pemodelan system berorientasi objek. Class diagram menunjukkan hubungan antar class dalam system yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi property dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti public, private dan protected [17].

2.3.4 *Sequence Diagram*

Sequence Diagram adalah suatu diagram yang digunakan untuk menggambarkan dan menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case. Adapun urutan proses yang dijelaskan yaitu interaksi yang terjadi antar class, operasi yang terlibat, urutan antar operasi, dan informs yang diperlukan oleh masing-masing operasi. Komponen utamanya adalah objek yang digambarkan dengan kotak segi empat atau bulat, message yang digambarkan dengan garis penuh, dan waktu yang ditunjukkan dengan progress *vertical* [17].

2.4 Structured Analysis and Design

Structured analysis and design adalah pendekatan sistem analisis dan desain sistem informasi secara hierarki. SADM diproduksi untuk CCTA, kantor pemerintah Inggris yang berkaitan dengan penggunaan teknologi di pemerintahan dari tahun 1980 dan seterusnya [18].

2.4.1 ERD

ERD (*Entity Relationship Diagram*) adalah satu model jaringan yang menggunakan susunan data yang disimpan dalam sistem. ERD merupakan model jaringan data yang menekankan pada struktur dan hubungan antardata [19].

2.4.2 *Diagram Konteks*

Diagram Context, adalah data flow diagram tingkat atas (DFD Top Level), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem) [19].

2.4.3 *Data Flow Diagram*

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data dimana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD terdiri dari empat komponen utama diantaranya adalah:

1) Entitas.
2) Proses.
3) *Data Store.*
4) *Data Flow*.

DFD dapat dikatakan merupakan *breakdown* dari *diagram konteks*. Analisis mengenai sistem yang lebih detail dapat dilakukan dengan melakukan *breakdown* pada DFD level 1, 2 dan seterusnya [19].

2.4.4 Spesifikasi Proses

Spesifikasi Proses adalah tabel yang berisi keterangan atau deskripsi dari semua proses yang terdapat pada DFD. Logika proses yang ditulis pada spesifikasi proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau

pseudecode tetapi tidak boleh mengkombinasikan kedua bahasa tersebut [19]. Berikut adalah Tabel 2.6 Contoh Spesifikasi Proses:

Tabel 2. 6 Contoh Spesifikasi Proses

No	Proses	Keterangan
1.	No Proses	Menyatakan no proses
2.	Nama Proses	Menyatakan nama proses
3.	Source (Sumber)	(Menyatakan sumber data input menuju proses)
4.	Input	(menyatakan isi data yang masuk ke proses)
5.	Output	(menyatakan informasi yang keluar dari proses)
6.	Destination	(menatakan tujuan informasi output dari proses)
7.	Logika Proses	(menyatakan algoritma dari proses)

2.4.5 Kamus Data

Kamus data adalah tabel yang berisi deskripsi dari data yang mengalir pada DFD [19].

Tabel 2. 7 Contoh Kamus Data

Nama aliran data	Data Siswa Nama dari data yang digunakan
Where used/How used	Daftar pada poin proses apa data digunakan
Keterangan	Uraian singkat dari data yang digunakan
Struktur Data	Daftar komponen data yang ada pada data yang digunakan

Deskripsi	Jenis data dalam representasi komputer untuk masing-masing data.

2.5 *Database*

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur, dan juga batasan-batasan data yang akan disimpan. Basis data merupakan aspek yang sangat penting dalam sistem informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghidari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga update yang rumit.

Proses memasukkan dan mengambil data ke dan dari media penyimpanan data memerlukan perangkat lunak yang disebut dengan sistem manajemen basis data (database management system | DBMS). DBMS merupakan sistem perangkat lunak yang memungkinkan user untuk memelihara, mengontrol, dan mengakses data secara praktis dan efisien. Dengan kata lain semua akses ke basis data akan ditangani oleh DBMS. Ada beberapa fungsi yang harus ditangani DBMS yaitu mengolah pendefinisian data, dapat menangani permintaan pemakai untuk mengakses data, memeriksa sekuriti dan integriti data yang didefinisikan oleh DBA (Database Administrator), menangani kegagalan dalam pengaksesan data yang disebabkan oleh kerusakan sistem maupun disk, dan menangani unjuk kerja semua fungsi secara efisien.

Tujuan utama dari DBMS adalah untuk memberikan tinjauan abstrak data kepada user (pengguna). Jadi sistem menyembunyikan informasi tentang bagaimana data disimpan, dipelihara, dan tetap dapat diambil (akses) secara efisien. Pertimbangan efisien disini adalah bagaimana merancang struktur data yang kompleks tetapi masih tetap bisa digunakan oleh pengguna awam tanpa mengetahui kompleksitas strukturnya. Database dibagi menjadi dua jenis yaitu [20]:

1. Basis Data Flat File

Basis data flat-file ideal untuk data berukuran kecil dan dapat dirubah dengan mudah. Pada dasarnya, mereka tersusun dari sekumpulan string dalam satu atau lebih file yang dapat diurai untuk mendapatkan informasi yang disimpan. Basis data flat-file baik digunakan untuk menyimpan daftar atau data yang sederhana dan dalam jumlah kecil. Basis data flat-file akan menjadi sangat rumit apabila digunakan untuk menyimpan data dengan struktur kompleks walaupun dimungkinkan pula untuk menyimpan data semacam itu. Salah satu masalah menggunakan basis data jenis ini adalah rentan pada korupsi data karena tidak adanya penguncian yang melekat ketika data digunakan atau dimodifikasi.

2. Basis Data Relasional

Basis data ini mempunyai struktur yang lebih logis terkait cara penyimpanan. Kata "relasional" berasal dari kenyataan bahwa tabel-tabel yang berada di basis data dapat dihubungkan satu dengan lainnya. Basis data relasional menggunakan sekumpulan tabel dua dimensi yang masing-masing tabel tersusun atas baris (tupel) dan kolom (atribut). Untuk membuat hubungan antara dua atau lebih tabel, digunakan key (atribut kunci) yaitu primary key di salah satu tabel dan foreign key di tabel yang lain. Saat ini, basis data relasional menjadi pilihan karena keunggulannya. Beberapa kelemahan yang mungkin dirasakan untuk basis data jenis ini adalah implementasi yang lebih sulit untuk data dalam jumlah besar dengan tingkat kompleksitasnya yang tinggi dan proses pencarian informasi yang lebih lambat karena perlu menghubungkan tabel-tabel terlebih dahulu apabila datanya tersebar di beberapa tabel.

2.6 *MySQL*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL* AB membuat *MySQL* tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL [21].

Tidak seperti PHP atau Apache yang merupakan software yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu *MySQL* AB. *MySQL* AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan *MySQL* AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Adapun fitur-fitur *MySQL* antara lain [12]:

1. *Relational Database System*. Seperti halnya *software database* lain yang ada di pasaran, MySQL termasuk RDBMS.
2. Arsitektur *Client-Server*. MySQL memiliki arsitektur *client-server* dimana *server database* MySQL terinstal di *server*. *Client* MySQL dapat berada di komputer yang sama dengan *server*, dan dapat juga di komputer lain yang berkomunikasi dengan *server* melalui jaringan bahkan *internet*.
3. Mengenal perintah SQL standar. SQL (*Structured Query Language*) merupakan suatu bahasa standar yang berlaku di hampir semua software *database*. MySQL mendukung SQL versi SQL:2003.
4. Mendukung Sub Select. Mulai versi 4.1 MySQL telah mendukung select dalam select (*sub select*).
5. Mendukung *Views*. MySQL mendukung *views* sejak versi 5.0
6. Mendukung *Stored Prosedured* (SP). MySQL mendukung SP sejak versi 5.0.
7. Mendukung *Triggers*. MySQL mendukung *trigger* pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
8. Mendukung *replication*.
9. Mendukung transaksi.
10. Mendukung *foreign key*.

Perintah-perintah yang ada di SQL antara lain:

1. Menambah data (*Insert*)

   Perintah SQL yang digunakan untuk menambah data ke *database* adalah perintah *INSERT*. Perintah ini mempunyai bentuk umum sebagai berikut:

   INSERT [INTO] nama_tabel [(daftar *field*/kolom)] VALUES (daftar_data).

2. Mengakses data (*Select*)

   Perintah SQL yang digunakan untuk memilih data adalah perintah *SELECT*. Bentuk umum perintah ini adalah sebagai berikut:

   SELECT daftar_select *FROM* daftar_tabel

   [WHERE kondisi_pencarian]

   [GROUP BY daftar_group_by]

   [HAVING kondisi_pencarian]

   [ORDER BY daftar_order[ASC │DESC]]

3. Mengedit/Mengubah data (*Update*)

   Perintah SQL yang digunakan untuk mengubah data adalah perintah *UPDATE SET*. Perintah ini mempunyai bentuk umum sebagai berikut:

   UPDATE nama_table SET field1=databaru1

   [data2=databaru2]

   [WHERE kondisi_update].

4. Menghapus data (*Delete*)

   Perintah SQL yang digunakan untuk menghapus data adalah perintah *DELETE FROM*. Perintah ini mempunyai bentuk umum sebagai berikut:

   DELETE FROM tabel_hapus WHERE kondisi_hapus.

2.7 Internet

Internet (*interconnection networking*) adalah jaringan komunikasi global yang terbuka dan menghubungkan jutaan bahkan milyaran jaringan komputer dengan berbagai tipe dan jenis, dengan menggunakan tipe komunikasi seperti telepon, satelit dan lain sebagainya. Awalnya internet merupakan jaringan komputer yang dibentuk oleh Departemen Pertahanan Amerika Serikat pada tahun 1969 melalui sebuah proyek yang disebut dengan ARPANET. Misi awal dari proyek ini awalnya hanya untuk keperluan militer saja, tetapi lambat laun terus berkembang dan bisa dinikmati oleh semua kalangan. Terciptanya internet telah membawa perubahan yang sangat berarti dalam berbagai aspek kehidupan manusia. Selain itu, internet juga telah melahirkan dunia baru yang memiliki pola, corak sekaligus karakteristik yang berbeda dengan dunia nyata [22].

Internet yang sering disebut dengan jagad raya informasi menyajikan sekian banyak fasilitas yang dapat dinikmati oleh pengguna internet. Berikut ini beberapa fasilitas yang dapat dinikmati oleh pengguna internet yaitu www (*world wide web*), email, *mailing list*, BBS, *chatting*, *newsgroup*, FTP dan *gopher* [22].

2.8 Bootstrap

Bootstrap merupakan sebuah *framework* CSS dari Twitter, yang menyediakan kumpulan komponen-komponen antarmuka dasar pada web yang telah dirancang sedemikian rupa untuk digunakan bersama-sama. Selain komponen antarmuka, Bootstrap juga menyediakan sarana untuk membangun *layout* halaman dengan mudah dan rapi, serta modifikasi pada tampilan dasar HTML untuk membuat seluruh halaman web yang dikembangkan senada dengan komponen-komponen lainnya.



Gambar 2. 19 Bootstrap [23]

Lisensi yang digunakan oleh Bootstrap yaitu lisensi Apache 2.0, sebuah lisensi yang sangat terbuka sehingga kita dapat dengan bebas dan mudah menggunakan Bootstrap, tanpa perlu khawatir akan ancaman legal dari Twitter atau pihak lainnya.

2.9 Metode Pengujian

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Perangkat lunak dapat diuji dengan dua cara, yaitu:

1) Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal, loop, logika, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.
2) Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

2.9.1 *White-Box Testing*

Pengujian *white box (glass box*) adalah pengujian yang didasarkan pada pengecekan terhadap detil perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasikan kasus uji tambahan.

Tujuan penggunaan *white box* untuk menguji semua statement program. Penggunaan metode pengujian *white box* dilakukan untuk:

1) Memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali.
2) Menggunakan semua keputusan logis untuk semua kondisi true atau false.
3) Mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4) Menggunakan struktur data internal untuk menjamin validitas jalur keputusan.

2.9.2 *Black-Box Testing*

Pengujian *Black Box* merupakan pendekatan komplementer dari teknik white box, karena pengujian *Black Box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik *white box.* Pengujian *Black Box* berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian *Black Box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan

untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *Black Box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian *Black Box* berusaha menemukan kesalahan dalam kategori:

1) Fungsi-fungsi yang tidak benar atau hilang
2) Kesalahan *interface*
3) Kesalahan dalam struktur data atau akses *database* eksternal
4) Kesalahan kinerja
5) Inisialisasi dan kesalahan terminasi.

Berbeda dengan pengujian *white box*, pengujian *Black Box* cenderung diaplikasikan selama tahap akhir pengujian. Pengujian *Black Box* harus dapat menjawab pertanyaan sebagai berikut:

1) Bagaimana validitas fungsional diuji
2) Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
3) Apakah sistem akan sangat *sensitive* terhadap harga input tertentu
4) Bagaimana batasan dari suatu data diisolasi
5) Kecepatan data apa dan *volume* data apa yang akan ditoleransi oleh system
6) Apa pengaruh kombinasi tertentu dari data terhadap sistem operasi.