BAB 2
LANDASAN TEORI

2.1 Golf

Golf adalah permainan luar ruang yang dimainkan secara perorangan atau tim yang berlomba memasukkan bola ke dalam lubang-lubang yang ada di lapangan dengan jumlah pukulan tersedikit mungkin [1]. Bola golf dipukul dengan menggunakan satu set tongkat pemukul yang disebut klab (stik golf). Golf adalah salah satu dari permainan yang tidak memiliki lapangan permainan yang standar, melainkan dimainkan di padang golf yang masing-masing memiliki desain unik, dan biasanya terdiri dari 9 atau 18 hole (lubang). Aturan utama dalam golf adalah "memainkan sebuah bola dengan stik golf dari daerah tee (teeing ground) ke dalam lubang dengan satu pukulan atau beberapa pukulan berikutnya sesuai dengan Aturan".

2.1.1 Nama Istilah Golf

A. *Par* (jumlah pukulan)

   Merupakan peringkat rata-rata penampilan permainan dan jumlah pukulan pada jarak tertentu. Yang dinilai berdasarkan kemampuan seseorang, kesalahan selama permainan dan kesulitan rintangan di lapangan; semua persyaratan tersebut diperhitungkan untuk masing-masing lubang yang dijumlahkan sampai lubang ke 18.

B. Tee (lapangan tempat memukul bola)

   Dapat digabungkan sepanjang sejarak pukul suatu permainan biasa, tetapi mungkin dipisahkan menjadi untuk permainan kejuaran bersifat kejuaraan, hadiah sebaiknya dipisahkan. pertandingan untuk pemain wanita dapat menggunakan jarak yang lebih pendek. Luas untuk ”Tee” ini adalah 300 m2 (400 m2 untuk par dengan 3 lubang).

C. Green

Ukurannya disesuikan dengan jarak dlubang permainan, atangnya bola yang dipukul dan disesuikan untuk masing-masing luasnya antara 400 m2 sampai 600 m2.

D. Fairway (Jalur lewat bola)

Lebar normal 30-40 m. jarak awal 100 m untuk “tee” pria. Jarak permainan/lubang = 60 m (90 m pada perbatasan).

E. Rough (lahan berumput tinggi)

Adalah jarak dari “tee” ke jalur “fairway”. Umumnya berupa jalur rumput pendek sekitar 10 m sebelumnya pepohonan ,semak-semak, dan sebagainya.

F. Bunker (Lapangan berpasir)

Lapangan berpasir rata-rata seluas 100 m2 dengan bentuk dan ukuran yang berbeda-beda. Untuk mempengaruhi jalannya permainan tentukan garis optimum dan sediakan pilihan dan hukuman untuk setiap kesalahan permainannya.

G. Pond (Kolam)

Kolam ini luas dan bentuknya berfariasi sesuai dengan tingkat hambatan.

H. Driving Range

Merupakan sarana latihan memukul,bola golf baik yang terbuka maupun terlindungi sinar matahari yang mana mempunyai panjang kurang lebih 200 m.

I. Caddy House

Merupakan ruang istirahat dan ruang ganti bagi caddy.

2.1.2 Teknik Bermain Golf

1. Sikap Dasar
   a. Posisi kaki depan harus sedikit di depan bola dan harus sedikit lebih lebar dari bahu Anda. Untuk pengguna tangan normal, maka kaki yang harus agak sedikit di depan adalah kaki kiri. Dan jika pemain

adalah seorang kidal, maka kaki kanan lah yang harus di tempatkan agak sedikit di depan.

b. Dekatkan bagian tengah dari tongkat golf terhadap bola dengan bahu agak sedikit membungkuk dan tangan lurus kebawah. Jangan berdiri terlalu dekat dengan bola, tekuk lutut untuk membuat badan agar sedikit membungkuk.

c. Menekuk sedikit lutut dengan menyeimbangkan tubuh saat akan mengayungkan tongkat agar ayunan tangan lebih maksimal.

2. Memegang Grip

a. Pegang tongkat golf dengan tenang untuk memberikan akurasi lebih pada pukulan sehingga bisa menjangkau jarak yang diinginkan dengan baik.

b. Pegang tongkat golf seperti halnya memegang tongkat baseball.

3. Mengayunkan Tongkat Golf

a. Melakukan gerakan backswing saat mengayunkan tongkat ke belakang hingga tongkat mencapai atas kepala. Kemudian memutar badan saat melakukan backswing.

b. Memutar tubuh saat melakukan Downswing yaitu gerakan ayunan kebawah setelah backswing. Tubuh harus mengikuti alur gerakan dari tangan agar mendapatkan momentum pukulan yang pas.

c. Melakukan gerakan Backswing dan Downsing secara berulang sebelum memukul bola untuk memastkan pukulan akan mengenai bola secara akurat.

2.1.3 Alat-alat Dalam Permainan Golf

1. Stik Golf

Ada tiga tipe stik golf (club) yang merupakan perlengkapan dasar untuk permainan golf, yaitu: wood, iron, dan putter.

a. Wood

Wood digunakan untuk pukulan jauh, termasuk driver untuk kepeluan tee-shot. Head Club golf pada awalnya dulu berbahan dasar kayu namun

saat ini Head Club golf ini tidak terbuat dari bahan kayu lagi, namun terbuat dari logam berongga, bisa titanium atau baja.

Tongkat golf wood dirancang untuk menghasilkan jarak terbesar dengan setiap stroke. Dari 14 klub dalam satu set standar, woods membuat kurang dari setengah. Klub golf wood biasanya digunakan untuk pukulan lurus dan panjang ke lapangan golf, karena karakternya memungkinkan untuk jarak jauh namun dengan akurasi yang tidak seakurat jenis stick yang lain. Biasanya dibutuhkan latihan yang rutin agar bisa menjaga bola tetap lurus di lapangan.

2. Iron

Iron digunakan untuk pukulan menengah, mengincar area green (tempat hole). Iron merupakan tongkat golf yang paling sering digunakan dalam permainan golf.

Iron jauh lebih fleksibel jika dibandingkan dengan 2 jenis tongkat yang lain sehingga jenis tongkat ini biasanya selalu lebih banyak berada di dalam tas pemain golf. Iron bisa mencapai jarak yang cukup jauh namun tidak bisa sejauh woods.

Iron biasanya dimulai dari no 3 sd 9 plus P (Pitching), Iron 3 yang paling panjang shaftnya dan memiliki sudut kemiringan permukaan yang kecil, makin besar nomor ironnya makin pendek shaftnya dan memiliki sudut kemiringan permukaan yang makin besar.

3. Putter

Putter merupakan jenis stik yang digunakan untuk keperluan membuat goal memasukkan bola di green. Tongkat golf putter biasanya digunakan pada putting green pada akhir pukulan ataupun pada bagian lain dari permainan yang memerlukan akurasi halus. Jenis klub putter dirancang khusus untuk menjadi tongkat dengan akurasi yang sangat tinggi dan jarak yang paling pendek yang biasanya hanya mempunyai jarak beberapa meter dari target.

Bola yang dipukul menggunakan putters tidak bertujuan untuk terbang di udara namun sebaliknya putters hanya mendorong ringan di tanah di

mana mereka akan menggulung dengan lembut ke target. Bisa dibilang puting ini adalah bagian yang paling penting dalam permainan golf.

4. Bola Golf

Pada mulanya, bola golf terbuat dari kayu. Pada abad ke-17, bola kayu digantikan dengan bola dari bulu angsa yang dibungkus kantong pembungkus dari kulit sapi. Bola dipres ketika bulu angsa dan kantong pembungkus masih basah lalu dijahit dan dicat. Sesudah kering, kantong kulit menyusut dan bulu angsa mengembang sehingga dihasilkan bola yang keras.

Bola golf makin tahan lama dipakai setelah ditemukannya bola getah yang disebut gutty karena dibuat dari getah perca (bahasa Inggris: gutta percha) yang dipanaskan. Kepopuleran bola golf gutty berlangsung dari tahun 1848 hingga 1890-an. Kelenturan bola getah memungkinkan dipakainya stik golf dengan kepala stik dari besi.

Inti bola berupa karet padat yang dibungkus benang-benang karet sebelum dilapis dengan getah perca. Bola golf dengan inti dari karet menggantikan bola getah pada tahun 1899. Setelah Walter Travis memenangi kejuaraan golf amatir Amerika Serikat dengan bola golf dengan inti dari karet, bola golf dari getah perca tidak dipakai lagi.

5. Ukuran Bola Golf

   a. Diameter bola golf: minimum 42,67 mm
   b. Massa bola golf (Berat bola golf): lebih dari 45,93 gram.
   c. Bentuk: Permukaan bola Golf tidak rata atau cekung-cekung.

Bola serupa dengan cekungan-cekungan yang sesuai dapat mencapai dua kali lipat jarak itu. Cekungan tadi berguna untuk mengurangi daya hambat udara sehingga dapat memberi kemampuan pada bola golf untuk meluncur lebih jauh.

6. Shaft

Shaft (tangkai) digunakan di antara grip dan kepala club.

2.1.4 Skor dalam golf

Skor dalam golf berbeda dengan skor dalam kebanyakan permainan lainnya. Pemain akan menang bila mendapat skor yang sedikit (bahkan hingga minus). Berbeda dalam sepak bola atau bola basket di mana grup yang mendapatkan skor terbanyak yang akan menang.

Berikut adalah skor dalam golf:

(-3) Double eagle : tiga pukulan dibawah par.

(-2) Eagle : 2 pukulan dibawah par

(-1) Birdie : 1 pukulan dibawah par

(0) Par : jumlah pukulan dibawah par

(+1) Boogey : satu puklan diatas par

(+2) Double Boogey : dua pukulan diatas par

(+3) Triplle boogey : tiga pukulan diatas par

Selain di atas, ada juga yang dinamakan hole in one yaitu memasukkan bola dari teeing ground ke lubang dalam satu kali pukulan. Biasanya hole in one hanya terjadi pada hole ber-par 3.

2.2 Android

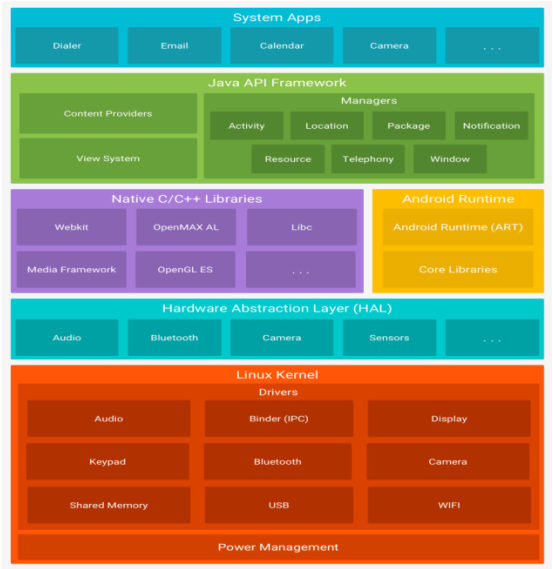
Android merupakan sistem operasi mobile yang tumbuh di tengah sistem operasi lainnya yang berkembang dewasa ini. Sistem operasi lainnya seperti Windows Mobile, i-Phone OS, Symbian dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat hardware yang ada. Akan tetapi, 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 [2].

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 [2].

2.2.1 Arsitektur Android

Arsitektur Android dapat digambarkan seperti pada Gambar 2.1 [3]



Gambar 2.1 Arsitektur Android

1. *System Apps*

   System Apps adalah adalah Aplikasi yang Secara *Default* atau *Built-in*, terinstal di berbagai perangkat Android. Aplikasi ini sering kita gunakan sehari-hari, seperti Dialer untuk Menelpon, Email untuk mengirim Email, SMS untuk mengirim pesan text, dan lain-lain. Tentunya berbagai macam perangkat dan OS Android, akan berbeda pada *System Apps*-nya. Tergantung *brand* atau *vendor* dari perangkat tersebut. Aplikasi ini juga umumnya tidak dapat di-*Uninstall*, harus menggunakan Aplikasi pihak ketiga. Bagi para opreker mungkin sering menggunakan Aplikasinya untuk melegakan memori internal. Bagi para *developer* Android sering menggunakan *System Apps*, untuk melengkapi fitur pada Aplikasi yang sedang dikembangkan [4].

2. *JAVA API Framework*

   JAVA API *Framework* adalah kumpulan *library* Java yang yang digunakan untuk pengembangan Aplikasi Android. Nah di sinilah peran

Android Developer dalam membuat dan mengembangkan Aplikasi mereka. Terdapat 3 Komponen utama, yaitu *View System, Content Provider* dan *Managers*. Dalam *Managers* terdiri dari berbagai macam *library* seperti, *Activity, Location, Package, Notification* dan lain-lain [4].

3. *Native C/C++ Libraries*

   Ini adalah dukungan *library* yang diperuntukan bagi kalian *developer*, yang menggunakan bahasa pemrograman C/C++ untuk mengembangkan Aplikasi Android. Terdapat berbagai macam library yang dapat kita kembangkan, seperti *WebKit*, *OpenGL ES*, Media *Framework* dan lain-lain [4].

4. *Android Run Time*

   Android pada dasarnya menggunakan Java, sebagai bahasa pemrograman utama. Java mengcompile ke dalam *bytecode*, kemudian dieksekusi oleh *Java Virtual Machine*, yang tersedia di berbagai macam perangkat dan sistem operasi . Intinya *bytecode* ini adalah hasil *binary code* (.class) dari instruksi *code* java (.java) yang telah di-*compile*, dan hanya dapat dibaca oleh *Java Virtual Machine*, yang kemudian dilanjutkan untuk digunakan oleh Sistem operasi seperti di Windows, *Linux* dan *Mac*.

   Sama halnya untuk Android yang telah dikembangkan oleh Google, yang membuat sebuah virtual machine baru yang bernama **Dalvik**, yang dikhususkan untuk perangkat mobile seperti Android. Jadi ketika ketika kita menulis baris code akan dikompile dua kali yaitu dengan menggunakan Java *Bytecode* dan *Dalvik bye code*, sebelum diteruskan ke *Dalvik Virtual Machine*.

   Di *Android Run Time* (ART) ini yang dikembangkan sejak OS Android KitKat, akan meng-*compile Dalvik byte code* ke dalam system binary, sehingga aplikasi yang dikembangkan untuk *Dalvik* akan bekerja, ketika menggunakan *ART*. Jika di *Dalvik virtual Machine*, akan tereksekusi setiap Aplikasi dijalankan (JIT/*Just in time*), berbeda dengan *Android Run Time*, yang akan tereksekusi sekali saja, ketika Aplikasi sudah terinstall di perangkat Android [4].

5. *Hardware Abstraction*

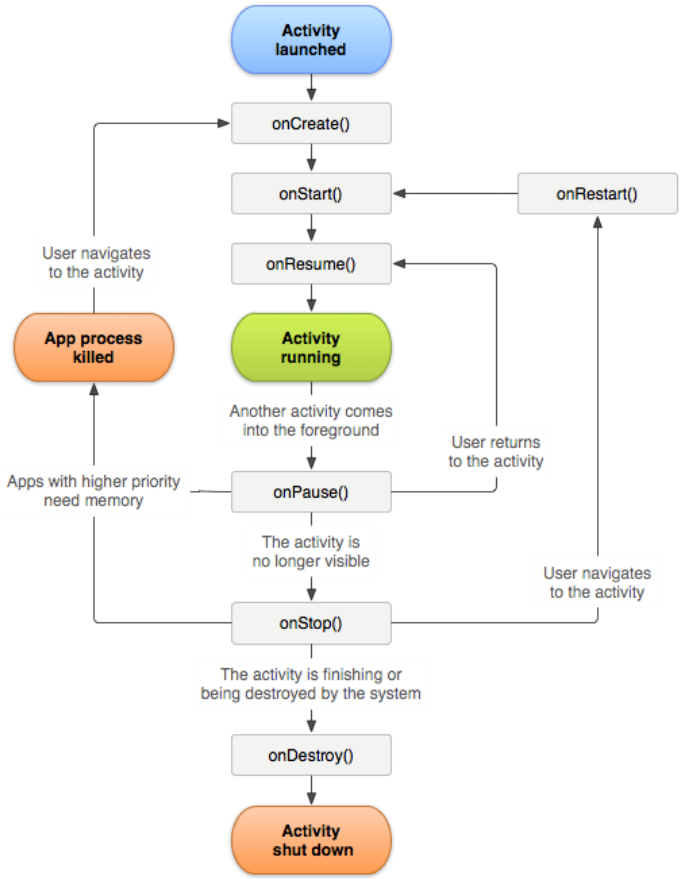
   Ini adalah komponen-komponen *hardware* pada perangkat Smartphone atau tablet pada umumnya, seperti Audio, *Bluetooth*, *Camera*, Sensors, dan masih banyak lagi [4].

6. *Linux Kernel*

   Linux Kernel ini adalah sebuah inti dari sistem operasi pada umumnya, pada perangkat Android akan menghandle *drivers*, agar komponen-komponen *hardware* yang ada pada perangkat Android bisa berjalan dengan baik [4].

2.2.2 Android Activity Lifecycle

Terdapat beberapa *state* dalam siklus hidup Android yang terjadi seperti diilustrasikan pada Gambar 2.2.



Gambar 2.2 *Android Lifecycle*

Akan tetapi hanya beberapa dari *state* tersebut yang menjadi statis diantaranya:

1. Resumed

   Resumed terjadi ketika aplikasi berjalan setelah state *paused. State* ini akan menjalankan perintah program yang ditulis pada *method* onResume() [5].

2. Paused

   Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalan *state* ini.

3. Stopped

   Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan *service* di *background* [5]. State lain seperti *Create and Started* bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem onCreate() dipanggil, dengan cepat sistem akan memanggil *method* onStart(), kemudian diikuti oleh onResume().

2.2.3 Fitur

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

1. *Multiproses* 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 [6].

   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*. *Multi-proses* dapat memberikan manfaat berupa *user experience* yang lebih banyak, namun pengguna fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [6].

2. *Touch Gesture dan Multitouch*

   Touchscreen adalah user interface intuitif yang digunakan banyak smartphine 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 [6].

3. *Hard and 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 [6].

2.2.4 Prinsip Desain

Android memiliki beberapa prinsip desain yang dapat menjadi acuan dalam membuat desain aplikasi android di antaranya sebagai berikut:

1. *Enchant Me*

   Antarmuka yang cantik, animasi yang ditempatkan dengan hati-hati, atau efek suara di saat yang tepat sungguh menyenangkan untuk dinikmati. Efek yang lembut menimbulkan perasaan serba mudah dan kesadaran bahwa kekuatan yang bisa diandalkan ada dalam genggaman. Android juga mengijinkan kita untuk menyentuh dan memanipulasi objek dalam aplikasi. Ini membuat pengguna merasakan pengalaman terbaik dalam sistem. Selain itu Android memberikan pengalaman untuk menambahkan konfigurasi personal terhadap konfigurasi default yang cantik tetapi tidak menggangu tugas utama. Android dapat mempelajari konfigurasi personal yang dilakukan oleh pengguna secara berulang dan dapat ditampilkan sehingga pengguna dapat melakuan konfigurasi tersebut lebih mudah [7].

2. *Simplify My Life*

   Desain Android memudahkan pengguna dengan konsep desain yang simple dan tidak membuat pengguna merasa rumit dalam penggunaannya. Seperti, menggunakan frasa pendek dengan kata-kata sederhana dalam

pertanyaan dan kalimat. Karena orang cenderung melewatkan kalimat-kalimat panjang. Mengubah text nama menjadi gambar atau foto yang relevan. Dan menampilkan apa yang hanya kita butuhkan dalam suatu perintah [7].

3. *Make Me Amazing*

   Android selalu mengedepankan pengguna. Pengguna merasa senang ketika mereka memahami sendiri sesuatu. Jadikan aplikasi Anda lebih mudah dipelajari dengan memanfaatkan pola visual dari aplikasi Android lainnya. Misalnya, gerakan menggeser dapat menjadi pintasan navigasi yang bagus. Dan Android selalu memberikan tanggapan atau umpan balik terhadap perintah yang kita berikan [7].

2.3 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:

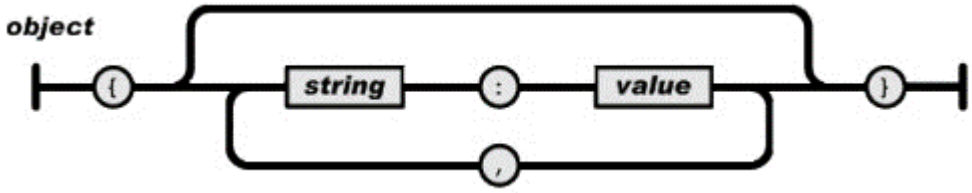
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), daftar berkunci (*keyed list*), atau *associative* array [8].
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*) [8].

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

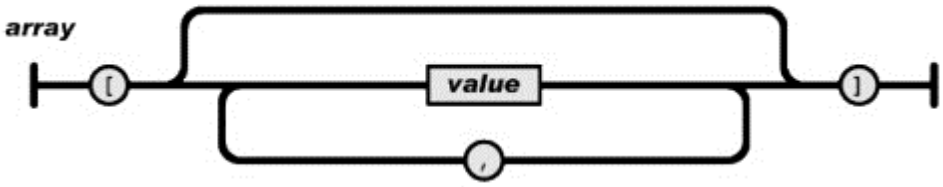
   Objek adalah sepasang nama / nilai yang tidak terurutkan. Seperti yang diilustrasikan pada Gambar 2.3 objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [8].



Gambar 2.3 Objek JSON

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 2.6. Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat. [9]



Gambar 2.4 Array JSON

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 [9].

2.4 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. 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. [10]

Cara menggunakan API :

1. Dilakukan dengan mengimpor package/kelas
2. Ada beberapa kelas bernama sama dipackage yang berbeda, yaitu:

- Import salah satu dan gunakan nama lengkap untuk yang lain
- 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.

2.5 *OOAD (Object Oriented Analysis Design)*

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 [11].

2.5.1 *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 [11]:

1. *Use Case Diagram*

   Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem 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:

   a) *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.

   b) *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.

   c) *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.

d) *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 [12].

e) *Depency*

Dependensi <<include>>

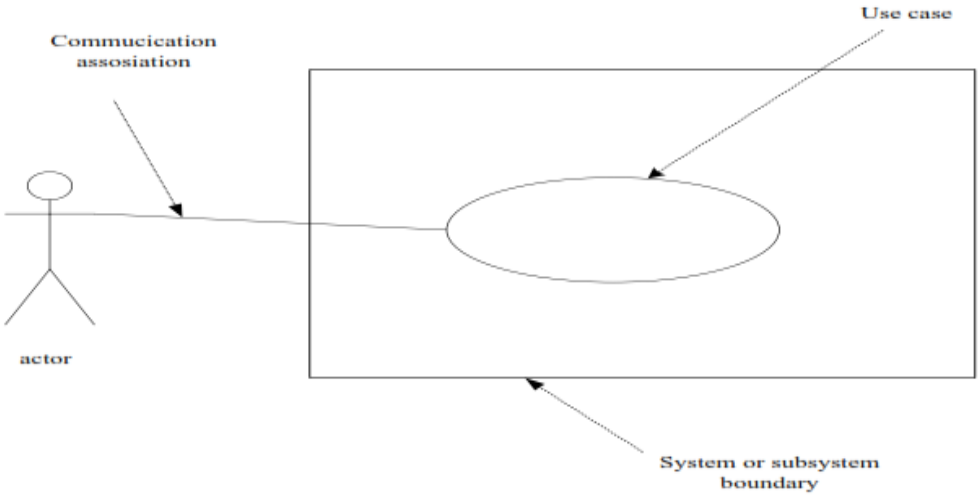
1) Mengidentifikasi hubungan antar dua *use case* di mana yang satu memanggil yang lain.
2) Jika pada beberapa *use case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan use case tersendiri dengan relasi dependensi setiap use case semula ke use case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

Dependensi <<extend>>

1) Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<extend>>.
2) Digambarkan serupa dengan dependensi <<include>> kecuali arah panah berlawanan.
3) Note: konsep “extend” ini berbeda dengan “extend” dalam Java.

f) *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.*



Gambar 2.5 *Generalization*

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.*

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.

Tabel 2.1 *Class Diagram*

Nama class
Atribut
Operasi/method

Atribut dan Operasi/method dapat memiliki tiga sifat berikut:

1. *Public*, dapat dipanggil oleh class apa saja.
2. *Protected*, hanya dapat dipanggil atau diakses oleh class yang bersangkutan dan *class* turunannya.
3. *Private*, hanya dapat dipanggil oleh dirinya sendiri (tidak dapat diakses dari luar class yang bersangkutan).

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.

2.6 *Gyroscope*

Menurut Kamus Besar Bahasa Indonesia (KBBI) *gyroscope* adalah alat berupa cakram yang sumbunya berputar antara dua penopang dan tetap dalam posisinya apabila tidak ada pengaruh kekuatan luar.

2.7 *Rangefinder*

Rangefinder adalah alat untuk mengukur jarak dari titik pengamatan ke obyek. Berbagai macam alat yang digunakan untuk mengukur jarak seperti laser cahaya, gelombang suara, maupun gelombang radio. Seiring dengan berkembangnya teknologi, rangefinder juga dapat menggunakan GPS dengan memanfaatkan Google Maps.

Prinsip dalam mengukur jarak pada setiap alat pada dasarnya sama, yakni mengambil garis lurus dari setiap titik pengamatan pada obyek yang dituju [12] [13]. Dari hasil setiap pengukuran, akan mendapatkan jarak yang cukup akurat untuk dimanfaatkan sesuai dengan kebutuhan.

2.8 Google Maps

Google Maps adalah layanan pemetaan web yang dikembangkan oleh Google. Layanan ini memberikan citra satelit, peta jalan, panorama 360°, kondisi lalu lintas, dan perencanaan rute untuk bepergian dengan berjalan kaki, mobil, sepeda (versi beta), atau angkutan umum.

Tampilan satelit Google Maps adalah "top-down". Sebagian besar citra resolusi tinggi dari kota adalah foto udara yang diambil dari pesawat pada ketinggian 800 sampai 1.500 kaki (240–460 meter), sementara sebagian besar citra lainnya adalah dari satelit.

2.9 Global Positioning System (GPS)

Global Positioning System (GPS) merupakan sebuah alat atau sistem yang dapat digunakan untuk menginformasikan penggunanya dimana dia berada (secara global) dipermukaan bumi yang berbasis satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital.

2.9.1 Definisi Global Positioning System (GPS)

GPS (Global Positioning System) adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (Departemen of Defense) Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24

satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS reciever yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama Way-point nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik.

GPS merupakan satu-satunya sistem satelit navigasi global untuk penentuan lokasi, kecepatan, arah, dan waktu yang telah beroprasi secara penuh didunia saat ini [13]. GPS menggunakan konstelasi 27 buah satelit yang mengorbit bumi, dimana sebuah GPS receiver menerima informasi dari tiga atau lebih satelit. GPS receiver harus berada dalam line-of sight (LoS) terhadap ketiga satelit tersebut untuk menentukan posisi, sehingga GPS hanya ideal untuk diguakan dalam outdoor positioning.

2.10 Metode Topsis

TOPSIS adalah metode multi kriteria yang digunakan untuk mengidentifikasi solusi dari himpunan alternative berdasarkan minimalisasi simultan dari jarak titik ideal dan memaksimalkan jarak dari titik terendah [14]. TOPSIS dapat menggabungkan bobot relatif dari kriteria penting

Menentukan matriks keputusan yang ternormalisasi (R), seperti persamaan 1.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{m} x_{ij}^2}}, (i = 1,2,\dots,n; j = 1,2,\dots,m)$$

keterangan:

xij merupakan rating kinerja alternatif ke-i terhadap atribut ke-j

rij adalah elemen dari matriks keputusan yang ternormalisasi.

a. Menentukan matriks keputusan yang terbobot (Y), seperti persamaan 2.

$$y = \begin{bmatrix} y_{11} & y_{12} & \cdots y_{ij} \\ y_{21} & y_{22} & \cdots y_{2j} \\ \vdots & \vdots & \ddots \vdots \\ y_{i1} & y_{i2} \cdots & y_{ij} \end{bmatrix} untuk\ y_{ij} = w_j r_{ij}$$

keterangan:

wj adalah bobot dari kriteria ke-j

yij adalah elemen dari matriks keputusan yang ternormalisasi terbobot.

a. Menentukan matriks solusi ideal positif (A+) dan matriks solusi ideal negatif (A), seperti persamaan 3 dan persamaan 4.

$$A^+ = (y_1^+, y_2^+, \cdots, y_j^+)$$

Dengan $A^- = (y_1^-, y_2^-, \cdots, y_j^-)$

$$y_j^+ = \begin{cases} \max\limits_i y_{ij}, jika\ j = keuntungan \\ \min\limits_i y_{ij}, jika\ j = biaya \end{cases}$$

$$y_j^- = \begin{cases} \max\limits_i y_{ij}, jika\ j = keuntungan \\ \min\limits_i y_{ij}, jika\ j = biaya \end{cases}$$

d. Menentukan jarak nilai alternatif dari matriks solusi ideal positif (di+) dan matriks solusi ideal negatif (di-), jarak solusi ideal positif (di+) seperti persamaan 7.

$$d_i^+ = \sqrt{\sum_{j=1}^{m} (y_{ij} - y_j^+)^2}$$

keterangan:

adalah elemen dari matriks solusi ideal positif jarak solusi ideal negatif (di-) seperti persamaan 8.

$$d_i^- = \sqrt{\sum_{j=1}^{m} (y_{ij} - y_j^-)^2}$$

keterangan:

adalah elemen dari matriks solusi ideal negatif

b. Menentukan nilai preferensi (ci) untuk setiap alternatif. Nilai preferensi merupakan kedekatan suatu alternatif terhadap solusi ideal, seperti persamaan 9.

$$c_i = \frac{d_i}{d_i^- + d_i^+}$$

keterangan:

nilai ci yang lebih besar menunjukkan prioritas alternatif.