BAB II
LANDASAN TEORI

II.1 Mall

Pusat perbelanjaan atau Mall adalah suatu kompleks bangunan komersil yang dirancang dan direncanakan beserta retail-retail dan fasilitas pendukungnya untuk memberikan kenyamanan dalam aktifitas perdagangan yang diwadahinya. Aktifitas perdagangan dalam pusat perbelanjaan modern ini tidak disertai tawar menawar barang seperti halnya pasar tradisional. Pusat perbelanjaan modern merupakan pusat perbelanjaan dengan sistem pelayanan mandiri atau dilayani pramuniada, menjual berbagai jenis barang secara eceran.Pusat perbelanjaan modern biasanya terdiri dari tenant-tenant yang disewakan kepada pelaku usaha serta terdapat anchor tenant yang berupa departement store atau supermarket [3].

II.2 Promosi

Promosi merupakan fungsi komunikasi dari perusahaan yang bertanggung jawab menginformasikan dan membujuk/mengajak pembeli. promosi merupakan salah satu variabel dalam bauran pemasaran yang sangat penting dilaksanakan oleh perusahaan dalam memasarkan produk. Promosi termasuk jenis komunikasi yang memberi penjelasan yang meyakinkan calon konsumen tentang barang dan jasa. Hal ini bertujuan untuk perhatian, mendidik, mengingatkan, dan meyakinkan calon konsumen [4].

II.3 Geofencing

Geofence adalah parameter virtual untuk wilayah geografis di dunia nyata. Parameter virtual yang ditetapkan ini bisa menjadi statis ataupun dinamis. Salahsatu contoh khas dari geofence dinamis adalah radius sekitar toko atau titik lokasi. geofence statis dapat di atur dari batas yang telah ditetapkan, seperti zona kehadiran di sekolah atau batas lingkungan. Kostum digital geofencing juga dapat digunakan, jika perangkat yang menyediakan layanan berbasis lokasi (LBS) pengguna masuk atau keluar geofencing, perangkat akan menerima pesan notifikasi. Notifikasi ini

mungkin berisi informasi tentang lokasi dari perangkat. Notifikasi Geo-fence mungkin dikirimkan ke ponsel sebagai contoh meningkatkan setiap bentuk lain dari tindakan [5].

Kita bisa menggambarkan geo fencing sebagai LBS proaktif. Dalam banyak kasus sistem proaktif jauh lebih nyaman dari yang reaktif, di mana pengguna harus secara eksplisit meminta data untuk lokasi. Ada berbagai jenis lokasi Peristiwa untuk memperbaiki posisi GPS yang dapat diuji. Misalnya, apakah pengguna di dekat dengan tempat tujuan (POI) atau ke pengguna lain. Geofence sekarang ini menjadi salah satu wajah baru di Layanan Berbasis Lokasi. Sekarang pengembangan asli untuk LBS sendiri lebih atau kurang selesai. Kita bias melihat beberapa standar umum dan aplikasi (berpikir tentang berbagai implementasi layanan Tempat-tempat seperti Foursquare, Twitter places, Google place, Facebook place, dll) Semua dari mereka memudahkan Anda dalam berbagi lokasi info atau mendapatkan beberapa hal yang berada di dekatnya. Dan geo-fence membuka pintu untuk beberapa personalisasi. Anda bisa mendapatkan beberapa kustom disesuaikan dengan data ditempat yang benar, terutama ketika kedua aliran data dan tempat (area) untuk menerima itu dihasilkan secara dinamis [5].

II.4 Twitter API

Metode pengambilan APIs memungkinkan pengembang untuk mengakses data inti Twitter. Hal ini mencakup pembaruan jadwal, data status, dan informasi pengguna. Metode Search API memberikan pengembang metode untuk berinteraksi dengan pencari Twitter dan tranding data. API saat ini mendukung format berikut data yang: XML, JSON, dan RSS dan format sindikasi Atom, dengan beberapa metode hanya menerima subset dari format ini . Ada empat objek utama yang dalam API Twitter: *Tweet*, Pengguna, Entitas dan *Places* [6].

1. *Tweet*

Tweet adalah bagian dasar dari Twitter. *Tweet*, juga dikenal lebih umum sebagai *update* status. Tweet dapat dimasukkan, membalas, menyukai, *unliked* dan dihapus [6].

2. Pengguna

Pengguna bisa siapapun atau apapun. Yang mereka tweet, ikuti, membuat daftar, memiliki jadwal sehari-hari, dapat disebutkan, dan dapat melihat ke dalamjumlah besar [6].

3. Entitas

Entitas menyediakan metadata dan informasi kontekstual tambahan tentang konten yang diposting di Twitter. Entitas yang tidak pernah lepas dari konten yang mereka jelaskan. Dalam API v1.1, entitas dikembalikan dimanapun Tweet ditemukan dalam API. Entitas merupakan hal yang penting dalam menyelesaikan URL [6].

4. *Places*

Places adalah tempat yang spesifik, lokasi di beri nama dengan sesuai koordinat geo. Mereka dapat dipasangkan pada *Tweet* dengan menentukan place_id ketika *tweeting*. *Tweet* terkait dengan tempat-tempat yang belum tentu dikeluarkan dari lokasi itu tetapi juga bisa berpotensi menjadi sekitar lokasi tersebut. Tempat dapat dicari. *Tweet* juga dapat ditemukan dengan place_id [6].

Saat ini, hanya koneksi SSL yang diperbolehkan dalam Twitter sebagai titik akhir API. Berkomunikasi melalui SSL melindungi privasi pengguna dengan melindungi informasi antara pengguna dan *Twitter* API saat dikirimkan ke publik [6].

II.5 Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008 [7].

II.5.1 Android Life Cycle

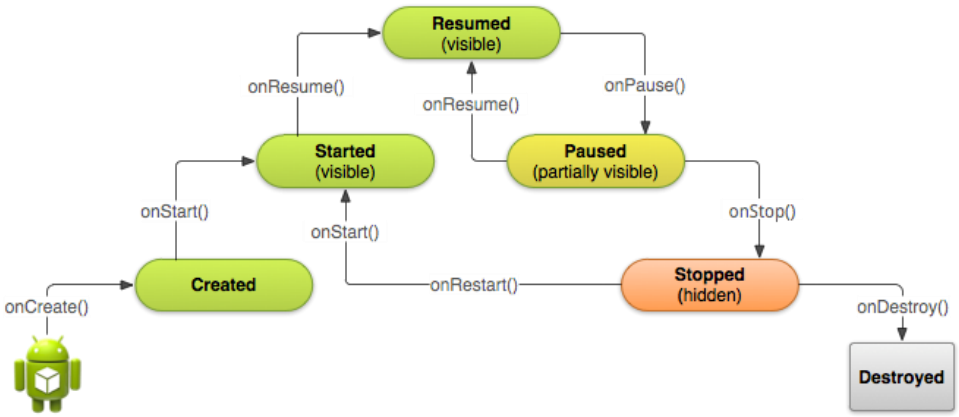
Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada , klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [8].

Tabel II. 1 Komponen Aplikasi Android

Functionality	Java Base Class	Examples
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 retrieve 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* [8].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana 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 desktop 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 II. 1

Terdapat beberapa *state* dalam siklus hidup android yang terjadi seperti diilustrasikan pada Gambar II.1, 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() [7].

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 dalam state ini [7].

3. **Stopped**

   Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service dibackground [7].

State lain seperti Created dan 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(), krmudian diikuti oleh onResume() [8].

II.5.2 Fitur

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

1. Multi-proses dan App Widgets

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 Widgets adalah mini aplikasi yang dapat *embedded* dalam aplikasi seperti home screen. App widgets 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 penggunaan fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [8].

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 dapat melakukan tracking lebih dari satu tangan dalam satu waktu, Fitur ini sering digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [8].

3. Hard dan Soft Keyboard

Salah satu fitur pada perangkat smartphone adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoperasian android.. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol

tersebut dalam aplikasi. Tombol non fisik adalah tombol yang dibuat oleh sistem operasi seperti keyboard virtual, dan tombol navigasi aplikasi [8].

II.5.3 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 II.2, 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, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet [9].



Gambar II. 2

2. Touch Feedback

Touch Feedback dalam android digunakan sebagai respon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [9].



Gambar II. 3

3. Pattern Gesture

Touch gesture berguna untuk meningkatkan experience pengguna dalam menggunakan aplikasi. Terdapat beberapa gesture yang didukung oleh android diataranya adalah:

1. 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 [9].

2. Long Press

Biasanya digunakan untuk seleksi data , dengan gesture ini dimungkinkan untuk memilih satu atau lebih item dalam sebuah tampilan dan menjalankan suatu fungsionalitas tertent [9].

3. Swipe Or Drag

Swipe adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. Swipe biasanya dilakukan untuk kebutuhan berpindah layar atau tergantung dari aplikasi. Swipe dapat dilakukan dari dan ke arah mana saja [9].

4. Double Touch

Pada Smartphone dan tablet android, melakukan dua kali tapping secara berturut-turut pada satu obyek, fungsinya berbeda dengan double klik mouse komputer. Pada Android, teknik ini biasanya dipakai untuk melakukan zoom in dan zoom out atau memperbesar / memperkecil sebuah obyek gambar [9].

5. Pinch Open

Teknik lain yang biasa digunakan adalah dengan menggunakan dua jari, di mana kedua jari tersebut menyentuh dua titik pada layar yang terpisah di mana ujung dari dua jari tersebut tidak bersentuhan, kemudian kedua jari tersebut sambil tetap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu [9].

6. Pinch Close

Teknik Pinch Close adalah kebalikan dari Pinch Open di mana spread dilakukan dengan berawal dari dua jari yang 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 [9].

II.6 Web Services

W3C mendefinisikan web service sebagai sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefinisikan, dideskripsikan, dan dimengerti oleh XML atau JSON dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML atau JSON melalui protokol internet Web service adalah sebuah software aplikasi yang tidak terpengaruh oleh platform, menyediakan method-method yang dapat diakses oleh network. Web Service juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entities bisnis yang berbeda. Beberapa karakteristik dari web service adalah:

1. *Message-based*
2. *Standards-based*
3. *Programming language independent*
4. *Platform-neutral*

Beberapa key standard di dalam web service adalah: JSON, XML, SOAP, WSDL and UDDI [10].

II.7 JSON

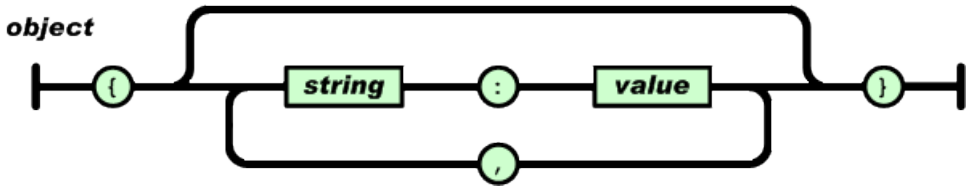
JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON 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#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur [11]:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array [11].
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) [11].

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut [11]:

A. Objek

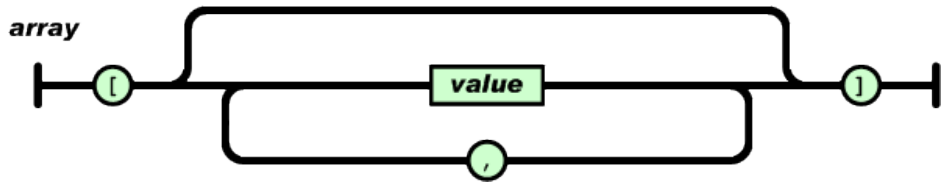
Objek adalah sepasang nama / nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [11].



Gambar II. 4

B. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma). Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat [11].



Gambar II. 5

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 [11].

II.8 Object Oriented Analysis Desain

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika 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 [12].

II.8.1 Unified Modeling Language (UML)

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

1. Use Case Diagram

Use case diagram merupakan suatu pemodelan yang menitik beratkan pada fungsionalitas utama dalam suatu konteks sistem. Use case diagram juga menekankan tingkah laku fungsional utama dalam sistem berinteraksi dengan objek diluar sistem tersebut. Selain itu, use case diagram juga telah menitik beratkan jenis hubungan diantara fungsi utama. Adapun komponen-komponen dalam use case diagram antaranya [12]:

a. Aktor

Aktor merupakan suatu entitas yang berkaitan dengan sistem tapi bukan dari bagian dalam sistem itu sendiri. Aktor berada diluar sistem namun berkaitan erat dengan fungsionalitas didalamnya. Aktor dapat memiliki hubungan secara langsung terhadap fungsi utama baik terhadap salah satu atau semua fungsionalitas utama. Aktor juga dapat dibagi terhadap berbagai jenis atau tingkatan dengan cara digeneralisasi atau dispesifikasi tergantung kebutuhan sistemnya. Aktor biasanya dapat berupa pengguna atau database yang secara pandang berada dalam suatu ruang lingkup sistem tersebut [12].

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 system [12].

2. Class Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. Class diagram menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi properti 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 [12].

3. Sequence Diagram

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case: interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi [12].

II.9 Metode Pengujian *Black Box*

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. Salah satunya yaitu metode pengujian *black box*.

Pengujian black box merupakan pendekatan komplementer dari teknik *white box*, karena pengujian *black box* diharapkan mampu mengungkap kelas 67 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 [13].

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.

Pengujian *black box* cenderung diaplikasikan selama tahap akhir pengujian. Pengujian *black box* harus dapat menjawab :

1. Bagaimana validitas fungsional diuji
2. Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
3. Apakah *system* 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 *system* operasi.

II.9.1 Graf Based Testing

Langkah pertama pada pengujian black box adalah memahami objek yang terdapat dalam model perangkat lunak dan menentukan hubungan yang dimiliki antara objek-objek tersebut. Pengujian berbasiskan model graf dilakukan terhadap perilaku *system*. *Graf Based Testing* menggambarkan graf yang mewakili hubungan antar objek pada modul sehingga tiap objek dan hubungannya dapat diuji. Pengujian ini dimulai dari mendefinisikan semua simpul dan bobot simpul, dimana

objek dan atribut diidentifikasikan, serta memberikan indikasi titik mulai dan berhenti.Terdapat tiga pola *link weight*, yaitu :

1. Transitivitas, yaitu hubungan antara tiga objek atau lebih yang menentukan bagaimana pengaruh hubungan tersebut menyebar pada objek yang ditentukan.
2. Simetris, yaitu hubungan antara dua objek secara dua arah.
3. Refleksif, yaitu hubungan yang mengarah pada node itu sendiri atau *loop null.*

II.9.2 Equivalence Partitioning (Partisi ekuivalensi)

Partisi ekuivalensi adalah metode yang membagi domain input dari suatu program ke dalam kelas data, menentukan kasus pengujian dengan mengungkapkan kelas-kelas kesalahan, sehingga akan mengurangi jumlah keseluruhan kasus pengujian. Bila suatu link weight mempunyai pola transitivitas, simetris, dan refleksif maka akan terdapat kelas ekuivalensi. Kelas ekuivalensi merepresentasikan serangkaian kondisi valid dan invalid untuk kondisi inputan. Secara khusus, suatu kondisi input dapat berupa harga *numeric*, suatu rentang harga, serangkaian harga yang terkait, atau suatu kondisi *Boolean*.

Penentuan Kelas Ekuivalensi :

1. Bila kondisi input menentukan suatu range, maka satu kelas ekuvalensi valid dan dua yang invalid ditentukan
2. Bila suatu kondisi input memerlukan suatu harga khusus, maka satu kelas ekuivalensi valid dan dua yang invalid ditentukan
3. Bila suatu kondisi menentukan anggota suatu himpunan, maka satu kelas ekuivalensi valid atau dua yang invalid ditentukan
4. Bila suatu kondisi input adalah *Boolean*, maka satu kelas valid dan satu yang lain ditentukan.

II.9.3 Boundary Value Analysis (Analisis Nilai Batas)

Analisis nilai batas adalah teknik desain proses yang melengkapi partisi ekuivalensi, dengan berfokus pada domain *output*. Pedoman untuk menentukan analisis nilai batas :

1. Bila suatu kondisi input mengkhususkan suatu range dibatasi oleh nilai a dan b, maka pengujian harus didesain dengan nilai a dan b, persis di atas dan di bawah a dan b secara bersesuaian
2. Bila suatu kondisi input mengkhususkan sejumlah nilai, maka pengujian harus dikembangkan dengan menggunakan jumlah minimum dan maksimum. Nilai tepat di atas dan di bawah minimum dan maksimum juga diuji.
3. Pedoman 1 dan 2 juga diaplikasikan ke kondisi output.
4. Bila struktur data program telah memesan suatu batasan, maka pengujian akan dilakukan sesuai dengan batasan struktur data tersebut.

II.9.4 Comparison Testing

Pengujian perbandingan adalah metode pembangkitan data uji yang dilakukan pada perangkat lunak yang dibuat redundan. Perangkat lunak yang redundan mempunyai dua tim pengembang yang masing-masing mengembangkan perangkat lunak sendiri-sendiri untuk spesifikasi yang sama. Metode pengujian perbandingan digunakan untuk membangkitkan data uji untuk salah satu perangkat lunak, yang kemudian digunakan sebagai masukan pada pengujian perangkat lunak yang lain.

Comparison Testing digunakan untuk *system* yang menganut *redundancy*, kasus uji yang dirancang untuk satu versi perangkat lunak dijadikan masukan pada pengujian versi perangkat lunak lainnya, dan diharapkan hasil kedua versi perangkat lunak harus sama. Jika hasil pengujian kedua perangkat lunak tersebut berbeda maka kedua perangkat lunak itu akan dicek untuk mencari yang salah.

II.10 Skala Likert

Skala *likert* digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang kejadian atau gejala sosial. Dengan menggunakan skala likert, maka variabel dijabarkan menurut urutan variabel, sub variabel, indikator, deskriptor. Dan deskriptor ini dapat dijadikan titik tolak untuk membuat butir instrumen berupa pernyataan atau pertanyaan yang perlu dijawab oleh responden.

Item item likert menyediakan repon dengan kategori yang berjenjang. Biasanya banyaknya jenjang adalah lima, yaitu sangat setuju, setuju, ragu ragu, tidak setuju dan sangat tidak setuju. Setiap kategori respon selanjutnya diberi skor. Untuk item positif skor terbesarnya adalah lima, skor terendah adalah satu dan sebaliknya jika item negatif. Untuk menghitung total skor tiap responden adalah dengan cara menjumlahkan skor skor item yang diperoleh reponden. Oleh karena itu, prosedur penskalaan likert sering disebut sebagai *Likert's Summeted Rating.*

Langkah langkah menyusunnya adalah sebagai berikut [14]:

1. Tentukan secara tegas sikap terhadap topik apa yang akan diukur.
2. Tentukan secara tegas sub varibel/dimensi yang menyusun sikap tersebut, kognitif, afektif dan konatif.
3. Susun pernyataan/pertanyaan atau item yang merupakan alat pengukur dimensi yang menyusun sikap yang akan diukur sesuai dengan indikator.
4. Setiap item diberi respon yang sifatnya tertutup.
5. Untuk setiap respon, jawaban diberi skor berdasarkan kriteria sebagai berikut: apabila item positif maka angka terbesar diletakkan pada respon sangat setuju, sedangkan bila item negatif maka angka terbesar diletakkan pada respon sangat tidak setuju.
6. Untuk mengetahui posisi setiap responden tentang suatu variabel, tentukan skor maksimal dan skor minimal yang mungkin dicapai oleh responden.

Langkah langkah untuk mengetahui bagaimana sikap tiap responden terhadap kualitas produk adalah :

1. Menentukan skor maksimal, yaitu skor jawaban terbesar dikali banyak item.
2. Menentukan skor minimal, yaitu skor jawaban terkecil dikali banyak item.
3. Menentukan nilai median, yaitu hasil penjumlahan skor maksimal dengan skor minimal dibagi dua.
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan skor minimal dengan median dibagi dua.
5. Menentukan nilai kuartil 3, yaitu hasil penjumlahan skor maksimal dengan median dibagi dua.
6. Membuat skala yang menggambarkan skor minimal, nilai kuartil 1, median, nilai kuartil 3 dan skor maksimal.
7. Mencari batas batas skor untuk masing masing kategori sikap.
8. Membuat tabel distribusi frekuensi sikap tiap responden terhadap kualitas produk.

Langkah langkah untuk mengetahui bagaimana sikap responden secara keseluruhan terhadap kualitas produk adalah :

1. menentukan skor maksimal, yaitu skor maksimal yang diperoleh tiap responden dikali banyaknya responden.
2. Menentukan skor minimal, yaitu skor minimal yang diperoleh tiap responden dikali banyaknya responden.
3. Menentukan nilai median, yaitu hasil penjumlahan total skor maksimal dengan total skor minimal dibagi dua.
4. Menentukan nilai kuartil 1, yaitu hasil penjumlahan total skor minimal dengan median dibagi dua.
5. Menentukan nilai kiartil 3, yaitu hasil penjumlahan total skor maksimal dengan median dibagi dua.
6. Membuat skala yang menggambarkan total skor minimal, nilai kuartil 1, median, kuartil 3 dan total skor maksimal.
7. Mencari batas batas skor untuk masing masing kategori sikap.
8. Menentukan skor total yang diperoleh seluruh responden.
9. Mengintrepretasikan skor total responden dengan skala pada point.

II.11 Model Metriks Kualitas Perangkat Lunak

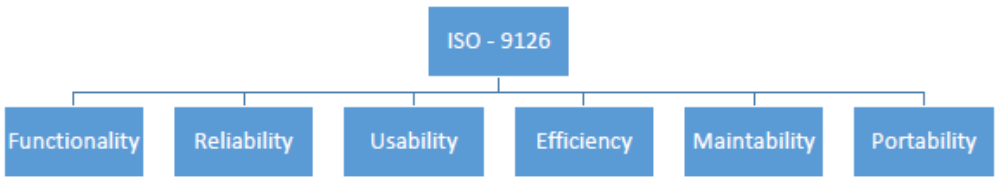
Dalam keilmuan rekayasa perangkat lunak terdapat beberapa model kualitas perangkat lunak. Pada setiap model ini terdapat beberapa faktor yang menjadi poin-poin utama dalam penilaian kualitas sebuah perangkat lunak. Berikut adalah penjelasan tentang model kualitas perangkat lunak yang akan diterapkan. Model ISO-9126 dikenalkan pertama kali pada tahun 1991 sebagai standarisasi kualitas produk perangkat lunak. Standarisasi ini dibuat karena banyaknya model kualitas yang ditawarkan sebagai faktor kualitas perangkat lunak [15].

Dalam dokumen pertama model ISO-9126 terdiri dari beberapa bagian model kualitas untuk kualitas sebuah produk perangkat lunak, di antaranya:

1. *Internal quality*
2. *External quality model*
3. *Quality in use model*

II.11.1 Model Kualitas Eksternal

Bagian pertama dari kualitas model tersebut menentukan 6 karakteristik yang mereka bagi kedalam 21 sub karakteristik untuk kualitas internal dan kualitas eksternal. Berikut ini merupakan pengertian dari masing-masing faktor kualitas dan sub pada ISO-9126.



Gambar II. 6 Faktor ISO 9126

1. Functionality

Faktor ini berhubungan dengan seberapa jauh sebuah perangkat lunak dapat memenuhi kebutuhan penggunanya.

a. Suitability

Sub-faktor suitability berhubungan dengan tingkat kemampuan dan kelayakan dari sebuah perangkat lunak untuk dapat menyediakan fungsionalitas untuk kebutuhan yang spesifik.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Functional adequacy	X = 1-A/B A = jumlah fungsi yang terdapat masalah dalam evaluasi B = Jumlah fungsi yang dievaluasi	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Functional implementation completeness	X = 1-A/B A = jumlah fungsi yang hilang terdeteksi dalam evaluasi B = Jumlah fungsi yang dijelaskan dalam spesifikasi kebutuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Functional implementation coverage	X = 1-A/B A = jumlah yang dilakasanakan dengan benar atau fungsi yang hilang dalam evaluasi B = jumlah fungsi yang dijelaskan dalam	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Functional specification stability (volatility)	X = 1-A/B A = Jumlah fungsi perubahan setelah memasuki operasi dimulai B = Jumlah fungsi yang dijelaskan dalam spesifikasi kebutuhan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

b. Accuracy

Accuracy berhubungan dengan kemampuan sebuah perangkat lunak untuk dapat memberikan hasil yang benar dan sesuai dengan fungsi yang dijalankan.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Accuracy to expectation	X = A / T	0 <= X Semakin dekat ke 0 semakin baik

	A = jumlah kasus yang dihadapi oleh pengguna dengan perbedaan terhadap hasil yang diharapkan T = Waktu operasi	
Computational Accuracy	X = A / T A = jumlah kasus yang dihadapi oleh pengguna dengan perbedaan terhadap hasil yang diharapkan T = Waktu operasi	0 <= X Semakin dekat ke 0 semakin baik
Precision	X = A / T A = jumlah kasus yang dihadapi oleh pengguna dengan perbedaan terhadap hasil yang diharapkan T = Waktu operasi	0 <= X Semakin dekat ke 0 semakin baik

c. *Interoperability*

Sub-faktor *interoperability* menggambarkan tentang kemampuan sebuah perangkat lunak untuk dapat berinteraksi dengan sistem yang lain atau dengan sistem tertentu

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Data exchangeability (Data format based)	X = A/B A = jumlah format yang data yang disetujui untuk ditukar dengan perangkat lunak lain atau sistem pengujian bursa data B = Total jumlah format data yang akan diperlukan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

Data exchangeability (User's success attempt based)	X = 1-A/B A = Jumlah kasus dimana pengguna gagal untuk pertukaran data dengan perangkat lunak atau sistem lain B = Jumlah kasus di mana pengguna upaya	0 <= X Semakin dekat ke 0 semakin baik

2. Usability

Usability berhubungan dengan seberapa baik perangkat lunak dapat dipahami, dipelajari, dan digunakan

a. Understandability

Sub-faktor ini berhubungan dengan seberapa jauh sebuah perangkat lunak dapat dipahami oleh pengguna baik dari secara konsep logis dan penerapan penggunaan perangkat lunak tersebut.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Completeness of description	X = A/B A =Jumlah kasus di mana pengguna berhasil melihat demonstrasi ketika pengguna mencoba untuk meliha demonstrasi B =Jumlah kasus di mana pengguna mencoba untuk melihat demonstrasi selama pengamatan	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Demonstration accessibility	X = A/B A = Jumlah demonstrasi / tutorial yang pengguna berhasil mengaksesnya B =Jumlah demonstrasi /tutorial yang tersedia	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Demonstration accessibility in use	X = A/B A =Jumlah kasus di mana pengguna berhasil	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik

	melihat demonstrasi ketika pengguna mencoba untuk meliha demonstrasi B =Jumlah kasus di mana pengguna mencoba untuk melihat demonstrasi selama pengamatan	
Demonstration effectiveness	X = A/B A =Jumlah fungsi yang di operasikan berhasil B = Jumlah demonstrasi / tutorial yang di akses	0 <= X <= 1 Semakin dekat ke 1.0 semakin baik
Function understand-ability	X = A/B A =Jumlah fungsi antarmuka yang tujuanya dengan benar dijalankan oleh pengguna B =Jumlah fungsi yang tersedia dari antarmuka	
Understandable input and output	X = A/B A =Jumlah input dan output data item yang pengguna berhasil mengerti B =Jumlah item data input dan output yang tersedia dari antarmuka.	

b. Learnability

Learnability menggambarkan tentang sebuah perangkat lunak dapat dipelajari dengan baik oleh penggunanya.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Ease of function learning	T= Berarti waktu yang dibutuhkan untuk pembelajaran menggunakan fungsi dengan benar	0 < T Semakin pendek semakin baik
Effectiveness of the user documentation and/or help system	= Berarti waktu yang dibutuhkan untuk pembelajaran	0 < T Semakin pendek semakin baik

	menggunakan fungsi dengan benar	

II.11.2 Model Quality In Use

Model *Quality In Use* mempunyai 4 Faktor yaitu *Effectiveness, Productivity, Safety dan Satisfaction*. Masing masing faktor juga mempunyai beberapa metriks. Berikut penjelasannya :

a. *Effectiveness* (Efektivitas)

Effectiveness berhubungan dengan seberapa efektif suatu perangkat lunak terhadap penggunanya.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Effectiveness	*Task completion*	X = A/B A = jumlah tugas diselesaikan B = jumlah tugas yang dicoba
	Error frequency	X = A/T A = jumlah kesalahan yang dibuat oleh pengguna T = jumlah tugas dicoba

b. *Productivity* (Produktivitas)

Productivity berhubungan dengan seberapa produktif kah perangkat lunak terhadap penggunanya.

Nama Metriks	**Pengukuran**	**Intrepretasi nilai**
Productivity	*Task completion*	X = A/B A = jumlah tugas diselesaikan B = jumlah tugas yang dicoba
	Error frequency	X = A/T A = jumlah kesalahan yang dibuat oleh pengguna T = jumlah tugas dicoba