BAB II
LANDASAN TEORI

2.1 Pengertian Implementasi

Menurut Kamus Besar Bahasa Indonesia (KBBI), implementasi adalahsuatu proses pelaksanaan atau penerapan, sementara itu menurut beberapa ahli, penerapan adalah tindakan mempraktekkan sebuah teori, metode, dan hal lain untuk dapat mencapai tujuan dan kepentingan tertentu yang diinginkan oleh kelompok ataupun golongan yang sebelumnya telah disusun dan direncanakan.

Penerapan (*implementasi*) bermuara pada sebuah aktivitas, aksi, tindakan, atau adanya mekanisme suatu sistem implementasi bukan sekedar aktivitas, tetapi suatu kegiatan yang terencana dan untuk mencapai tujuan kegiatan tertentu. (Usman, 2002).

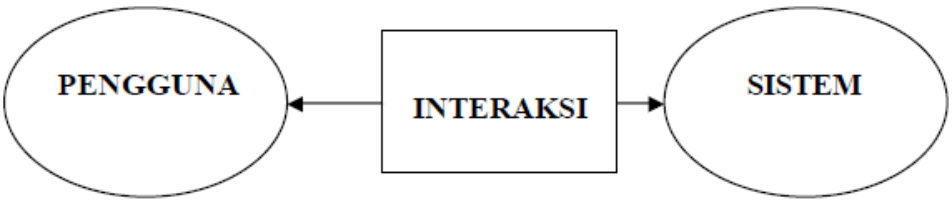
Penerapan (*implementasi*) adalah perluasan sebuah aktivitas yang saling menyesuaikan proses interaksi antara tujuan dan tindakan untuk mencapainya serta memerlukan jaringan pelaksana, serta birokrasi yang efektif. (Setiawan, 2002)

Berdasarkan pengertian-pengertian tersebut dapat disimpulkan bahwa kata penerapan (*implementasi*) bermuara pada aktifitas, adanya aksi, tindakan, atau mekanisme suatu system. Ungkapan mekanisme mengandung arti bahwa penerapan (*implementasi*) bukan sekedar aktifitas, tetapi suatu kegiatan yang terencana dan dilakukan secara sungguh-sungguh berdasarkan acuan norma tertentu untuk mencapai tujuan kegiatan.).

2.2 *Human Computer Interaction* (HCI)

Dari perspektif disiplin ilmu komputer, HCI atau Interaksi Manusia dan Komputer (IMK) berfokus pada perancangan dan evaluasi antarmuka pengguna *(user interface)*. Antarmuka pengguna adalah bagian dari sistem komputer yang agar manusia dapat berinteraksi dengan komputer.

Menurut Shneiderman dan Plaisant (2010, p15), IMK adalah disiplin ilmu yang berhubungan erat dengan proses perancangan, evaluasi, serta implementasi sistem komputer interaktif yang digunakan oleh manusia, selain itu juga mempelajari fenomena-fenomena besar yang berhubungan dengannya.



Gambar II-1 Model Interaksi Manusia dan Komputer

Model interaksi antara pengguna dengan sistem terdiri dari tiga komponen utama yaitu, pengguna, interaksi, dan sistem itu sendiri. Seperti yang terlihat pada gambar II-1. Kunci utama pada HCI adalah *usability*, yang artinya sistem harus mudah dimengerti dan digunakan oleh *end user* , dan memberi keleluasaan pada pengguna, serta mudah dipelajari.

2.3 Antarmuka Pengguna *(User Interface/*UI)

Antarmuka pengguna atau UI *(User Interface)* terkadang juga digunakan sebagai pengganti istilah *Human Computer Interaction* (HCI) dimana semua aspek dari interaksi pengguna dan komputer. Menumt Lastiansah (2012), *user interface* adalah *earn program* dan pengguna untuk berinteraksi. Semua yang terlihat dilayar, membaca dokumen dan dimanipulasi dengam *keyboard* atau *mouse* juga merupakan bagian dari *user interface.*

User interface berfungsi sebagai penghubung atau penterjemah informasi antara pengguna dengan sistem operasi, sehingga komputer dapat digunakan. Konsep *user interface* terdiri dari banyak aspek yang harus diperhatikan, karena akan berimplikasi pada beragam aplikasi teknologi seperti *electronic display*, aplikasi *web*, *mobile* dan lainnya, dengan demikian dapat disimpulkan bahwa UI bisa juga diartikan sebagai mekanisme inter-relasi atau *integrate* total antara

perangkat keras dan lunak sehingga membentuk suatu pengalaman berkomputer bagi pengguna.

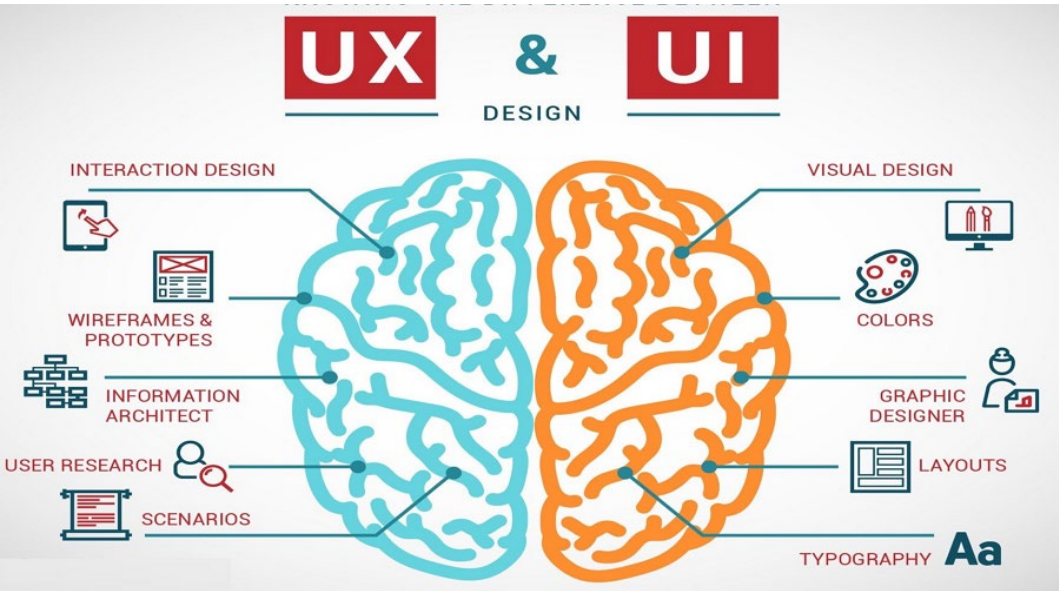
Jika dilihat dari sisi *software*, UI bisa berbentuk sebuah *Graphical User Interface* (GUI) atau berupa *Command Line Interfae* (CLI), sedangkan jika dilihat dari sisi *hardware* dapat berupa *Aplle Desktop Bus* (ADB), USB, danlainnya.

2.4 Pengalaman Pengguna *(User Experience/*UX)

(Creative Business Jakarta, 2013*) user experience* (UX) mempakan salah satu strategi mendesain produk yang berfokus pada perspektif pengguna. Strategi *User Experience* dibangun dan diimplementasikan bersamaan dengan strategi bisnis dan produk pemsahaan untuk melihat produk kita dari perspektif pengguna. Strategi *User Experience* juga dapat digunakan untuk melihat akan seperti apa interaksi pelanggan dengan pemsabaan melalui beragam produk yang telah dihasilkan.

Dalam *user experience* memiliki 3 poin utama, yaitu mendefinisikan value yang dapat dibe1ikan ke penggmna dan mengeksplorasi produk apakah dapat mencapai tujuan bisnis.

1. Mengidentifikasikan setiap kesempatan yang dapat meningkatkan kualitas produk dan mengeksplorasi setiap fase interaksi agar dapat mengidentifikasi sesuai dengan komponen yang sudah ditentukan.
2. Merencanakan pengembangan produk dan peningkatas secara terus-menerus.
3. Mengidentifikasi kesuksesan produk dan metode yang digunakan utnuk melakukan validasi keberhasilan produk tersebut.



Gambar II-2 Relasi UX dan UI *(opensystemtech.com)*

2.5 Grafis Antarmuka Pengguna *(Graphical User Interface* /GUI)

Menurut Lastiansah (2012) *Graphical User Interface* (GUI) merupakan tipe antarmuka yang bertujuan untuk melakukan interaksi antara sistem operasi dengan pengguna melalui gambar, grafik, ikon, dan menggunakan perangkat pendukung lainnya sebagai penunjuk *(pointing device)* contohnya *trackball* atau *mouse*.

Setiap Sistem Operasi akan memiliki nama tersendiri untuk setiap komponen GUI-nya, sebagai contoh; Microsoft member nama GUI Windows Vista sebagai Aero pada Windows XP sebagai Lunar, sedangkan Apple Mac OS X, GUI-nya dikenal sebagai Aqua, pada Linux OS, ada dua pengembangan utama *desktop environment*, yang masing-masing menghasilkan produk yaitu GNOME dan KDE.

2.6 UCD *(User Centered Design)*

Konsep dari UCD adalah pengguna sebagai pusat dari proses pengembangan sistem, dan tujuan/sifat-sifat, konteks serta lingkungan sistem yang keseluruhannya didasarkan dari pengalaman pengguna (Simatupang, 2014). *User Centered Design* pertama kali muncul dan diperkenalkan oleh Donald Norman's pada tahun 1980 (Abras, Maloney-Krichmar, & Preece, 2004) di laboratorium

University of California San Diego (UCSD), dan menjadi terkenal setelah penerbitan buku yang berjudul ―*User- Centered System Design: New Perspectives on Human-Computer Interaction* (Norman & Draper, 1986).

Adapun beberapa manfaat dari UCD menurut (Sripathi & Sandru, 2013) adalah:

1. Menghemat waktu
2. Mengurangi biaya
3. Peningkatan penjualan dan pendapatan
4. Penurunan pelatihan dan dukungan biaya
5. Peningkatan kepuasan pengguna
6. Memberikan nilai tambah suatu produk

Konsep UCD adalah tentang partisipasi dan juga pengalaman manusia dalam proses perancangan dan ketertlibatan pengguna akhir (*end user*) yang menggunakan sistem untuk menyelesaikan pekerjaannya, sedangkan pengguna tidak langsung adalah yang menggunakan sistem untuk penggunaan lain seperti *installers, system administrators,* dan juga *demonstrators*.

2.7 Flutter *Framework*

Flutter adalah sebuah *framework* aplikasi mobil sumber terbuka *(Open Source)* yang diciptakan oleh Google. Flutter digunakan untuk pengembangan perangkat lunak aplikasi *mobile* seperti Android dan iOS, dan menjadi metode utama dalam pembuatan aplikasi Google Fuchsia. (*Wikipedia.id).* Sama seperti *react native, framework Flutter* ini dapat digunakan untuk mengembangkan atau membuat aplikasi mobile yang dapat beroperasi pada perangkat iOS dan Android. Flutter dibuat menggunakan bahasa C, C++, Skia dan Dart.

Flutter telah menjadi *framework* yang menarik danjuga *worth* untuk dipelajari dan dikembangkan. Hal yangpaling menarik pada *framework* ini yaitu semua kodenya di c*ompile* dalam kode *native* nya (Android NDK, LLVM, AOT-*compiled*) tanpa *intrepeter* pada proses *compile*-nya sehingga menjadi lebih cepat. Dalam hal penulisan kode, *Flutter Framework* berbeda dengan *react native*, lebih

identik dengan Java Android hal ini menyebabkan *developer react native* sedikit kesulitan untuk memahami struktur kode Flutter.

Google telah mengakui bahwasannya flutter merupakan *framework* yang terinspirasi oleh *React*. Dengan satu *codebase* bisa menghasilkan aplikasi *platform* Android dan IOS, sangat menghemat waktu ketika membuat *prototype* atau tugas akhir. Flutter dengan mudah dapat dipelajari karena menggunakan bahasa pemrograman Dart yang terasa familiar bagi yang terbiasa menggunakan Java atau Javascript.

2.7.1 Fitur Flutter

1. Flutter adalah sumber lisensi terbuka (*open source)*. Bisa diartikan Flutter boleh dipakai secara gratis.
2. Pengembang *(Developer)* bisa menggunakan Flutter di Android Studio atau Visual Studio Code. Karena Flutter tersedia untuk kedua IDE tersebut.
3. Flutter bisa membuat aplikasi hanya dengan satu kali koding untuk dipublikasikan di *Google Playstore* *.apk) dan Apple Appstore (*.ipa). Performa dan UI aplikasi mirip dengan aplikasi yang dibuat menggunakan java.
4. Flutter berbeda dari *framework multiplatform* lainnya, Flutter membantu kita membuat sebuah aplikasi dengan tampilan atrataktif dan menarik serta performa yang lebih baik.
5. *Widget* dan *layout* berada dibawah kontrol penuh pengembang *(developer)*.

Tidak seperti *React Native, Nativescript*, dan *Fuse* yang membedakan adalah Flutter tidak menggunakan *Webview* maupun *widget* bawaan, Flutter mempunyai *engine render* sendiri untuk menampilkan *widget* nya, hal ini menguntungkan *developer* yang ingin menciptakan UI dan UX yang menarik, unik atraktif dan konsisten karena tidak bergantung pada widget bawaan OEM.

2.8 Visual Studio Code

Visual Studio Code adalah sebuah IDE yang berfungsi sebagai editor kode sumber *(source code)* dari berbagai bahasa pemrograman seperti PHP, Java, Python, Javascript, Node.js, C/C++, Go, C# bahkan untuk *script web* seperti HTML dan CSS. VSC dikembangkan oleh Microsoft untuk pengguna Windows, MacOS dan Linux, didalamnya termasuk dukungan untuk *debugging*, kontrol Git yang disematkan, penyorotan sintaksis, penyelesaian kode cerdas, snippet, *dan refactoring code.*VSC juga menyediakan emulator terintegrasi pada mode *debugging*.

Konsep Microsoft Visual Studio Code adalah *one-stop shop* yang memungkinkan *developer* fokus pada proses pengembangan dan melupakan *tools* lainnya. Visual Studio Code memeiliki beberapa fitur sebagai berikut:

1. ***Cross platform –*** Tersedia di macOS, Linux dan Windows.
2. ***Lightweight*** – Tidak lagi menunggu lama untuk memulai *(loading).* Pengembang dapat mengontrol sepenuhnya bahasa, tema, *debugger*, *commands* dan lain-lainnya sesuai keinginan.
3. ***Powerful editor*** – Memfungsikan fitur untuk *source code editing* yang sangat produktif, seperti membuat *code snippets, IntelliSense, auto correct*, dan *formatting* sehingga menghemat waktu.
4. ***Code Debugging*** –Visual Studio Code dapat melakukan *debug* pada kode dengan cara mengawasi kode, *variable*, *call stack* dan *expression* yang pada saat koding.
6. ***Source control*** – Visual Studio Code memiliki *integrated source control* termasuk *Git support in-the-box* dan penyedia *source code control* lainnya yang sering dignakan oleh pengembang.
7. ***Integrated terminal*** – Tidak ada lagi *multiple windows* dan *alt-tabs*. Kita dapat melakukan *command-line task* sekejap dan membuat banyak terminal di dalam satu layar kerja.

2.9 Dart *Programing Language*

Sebagai sebuah perusahaan teknologi raksasa, Google telah mengembangkan beberapa jenis bahasa pemrograman, diantaranya adalah bahasa pemrograman Dart *(Dart Pragraming Language)*. Kelebihan bahasa pemrograman Dart menjadikan bahasa ini menjadi salah satu bahasa baru yang dapat dipelajari oleh para *developer* maupun calon *developer*.

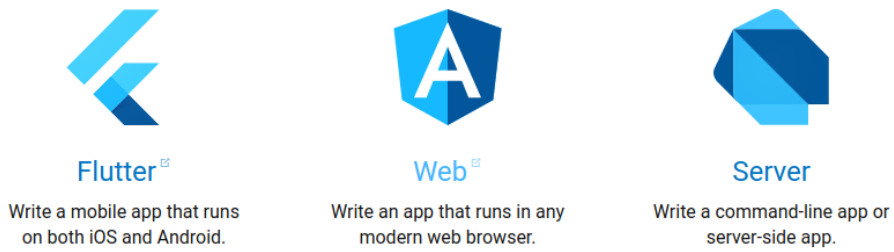
2.9.1 *Platform* Dart

Bahasa pemrograman Dart merupakan bahasa pemrograman *general-purpose* yang dirancang oleh Lars Bak dan Kasper Lund. Bahasa pemrograman ini dikembangkan sebagai bahasa pemrograman aplikasi yang dapat dengan mudah untuk dipelajari dan disebarkan. Bahasa pemrograman besutan Google ini dapat digunakan untuk mengembangkan berbagai macam platform termasuk di dalamnya adalah web, aplikasi mobile, server, dan perangkat yang mengusung teknologi *Internet of Things* (IOT).

Contoh beberapa aplikasi yang dibangun dengan Dart:

1. Google adsense *(front-end)*
2. Google adwords *(front-end)*
3. Google fiber *(front-end)*
4. Hamilton (Android dan iOS)

Dart digunakan pada 3 platform berikut:



Gambar II-3 *Platform* dukungan Dart

Bahasa pemrograman Dart dapat digunakan untuk mengembangkan aplikasi pada berbagai macam peramban modern. Selain itu Dart dapat digunakan secara bebas oleh para developer, karena dirilis secara *open-source* oleh Google di bawah lisensi BSD, Bahasa pemrograman Dart berbasis class dan berorientasi terhadap obyek dan menggunakan sintaks bahasa pemrograman C.

Dart dikenalkan oleh pihak Google sebagai pengganti *JavaScript*, tetapi secara opsional bahasa ini juga dapat dikompilasi dengan *JavaScript* menggunakan *Dart2js compiler*. Berbeda dengan bahasa pemrograman *JavaScript* yang bertipe statis , Dart merupakan bahasa bertipe dinamis.

2.9.2 Keunggulan Dart

Berikut beberapa keunggulan Dart:

1. Fleksibel

   Seperti yang dijelaskan sebelumnya, salah satu kelebihan bahasa pemrograman Dart adalah bahasa pemrograman tersebut termasuk ke dalam bahasa pemrograman bertipe dinamis. Bahasa pemrograman ini dapat dikompilasi ke dalam bahasa pemrograman *JavaScript* dengan *compiler* yang sudah disertakan di dalamnya.

   Bahasa pemrograman ini dikembangkan untuk mudah digunakan dalam pengembangan, sesuai dengan pengembangan aplikasi modern, dan memiliki implementasi berkinerja tinggi. Bahkan, bahasa pemrograman ini dapat digunakan juga sebelum dikompilasi.

   Dart VM menawarkan kemampuan untuk menjalankan secara langsung kode sumber tanpa perlu dikompilasi terlebih dulu. Bahasa pemrograman ini juga dapat langsung digunakan pada peramban Chrome tanpa perlu dikompilasi.

   Bahasa pemrograman Dart mendukung banyak arsitektur, termasuk di dalamnya IA-32, X64, MIPS, ARMv5TE, ARMv6, ARMv7, dan arsitektur ARM64. Bahasa pemrograman ini mendukung secara native pengembangan aplikasi mobile untuk ke dua platform Android dan iOS.

2. Berdiri Sendiri

Kelebihan bahasa pemrograman Dart lainnya adalah ketersediaan SDK yang dilengkapi dengan macam tools pengembangan. Salah satu tool-nya adalah Dart VM, dimana tool tersebut akan membantu developer untuk menjalankan kode dalam lingkungan *tampilan command line.*

Selain itu, dalam SDK tersebut juga terdapat *Dart2js compiler* yang dapat digunakan untuk mengkompilasi Dart ke daalam bahasa pemrograman JavaScript. SDK tersebut juga dilengkapi dengan manajer paket yang disebut dengan pup, yang dapat digunakan untuk menggunakan kode pihak ketiga atau berbagi kodingan.

3. *Concurrency*

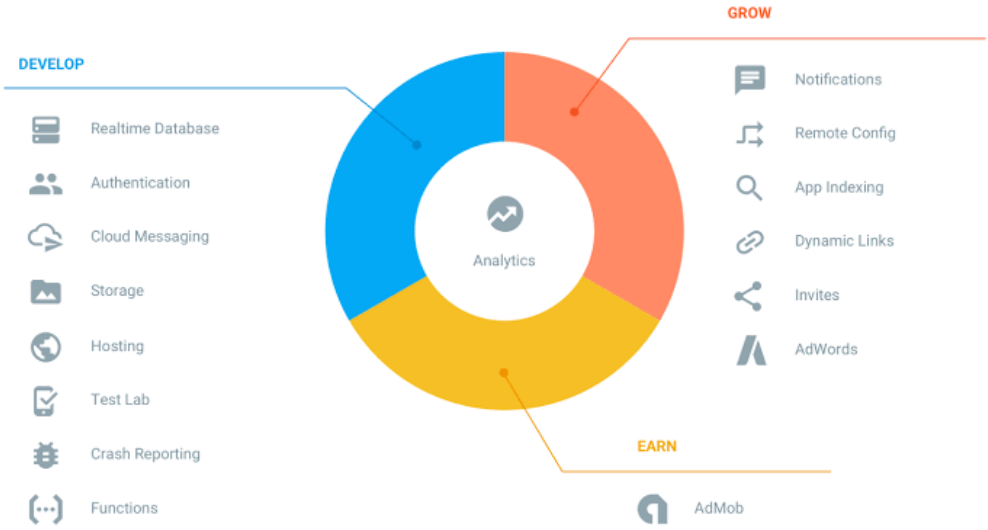
Bahasa pemrograman Dart memiliki kelebihan dengan adanya konstruksi nyata dari *concurrency* dan paralelisme. Kelebihan bahasa pemrograman Dart satu ini ditawarkan dengan bentuk Dart *Isolates*. Dengan adanya Dart Isolates, program-program akan terisolasi untuk bekerja secara independen tanpa adanya pembagian memori, akan tetapi tetap terdapat komunikasi diantaranya. Setiap program Dart menggunaka setidaknya satu buah isolasi.

2.10 *Firebase*

Firebase Realtime Database (RDB) menyediakan layanan *Backend Service* yang disediakan oleh Google. Layanan ini juga menyediakan pengembangan aplikasi API yang memungkinkan data aplikasi yang akan disinkron diimpan oleh client di *Firebase*, layanan untuk menyimpan data tersebut disimpan dalam format JSON *(Javasript Objek Notation)* dan dapat tersinkron secara *realtime* pada semua perangkat yang terhubung dan menggunakannya. Fitur unggulan *Firebase Realtime Database* diantaranya: *realtime, offline*. (Google Developers, 2017).

Cara kerja *Firebase* RDB adalah dengan cara menyediakan sambungan akses aman menuju *database* secara langsung dari perangkat *client*. Selain data tersebut disimpan pada server, data juga disimpan pada sisi *client* secara lokal,

proses sinkronisasi data terjadi ketika perangkat *client* memiliki jaringan internet. *Firebase* RDB dapat menggabungkan dan menyelesaikan konflik transaksi data secara otomatis ketika perangkat melakukan transaksi data pada waktu offline (Google Developers, 2017).



Gambar II-4 Fitur-fitur Pada Firebase

2.11 Android

Android merupakan sistem operasi (OS) untuk perangkat *mobile* yang menyertakan *middleware* (*virtual machine*) dan aplikasi utama lainnya. Android adalah modifikasi dari kernel Linux (Andry, 2011). Awalnya Android dikembangkan oleh perusahaan bernama Android Inc, sebuah perusahaan *start-up* kecil berlokasi di Palo Alto, California, Amerika Serikat yang didirikan oleh Andy Rubin, Rich Miner, Nick Sears, dan Chris White.

Awalnya Android diciptakan untuk kamera digital. Tetapi, karena pangsa pasar kamera digital tidak menjanjikan, maka tujuan tersebut beralih ke *Smartphone*. Kemudian Android menjadi pesaing dari perangkat mobile ber-*platform Symbian* dan *Windows Mobile*. Juli 2005, perusahaan tersebut diakuisisi Google dan para pendirinya bergabung ke Google, Andy Rubin kemudian menjabat sebagai Wakil Presiden divisi *Mobile*.

Pada dasarnya *Operating System* Android dibuat untuk menyediakan *platform* sumber terbuka *(Open Source)*, dengan tujuan memudahkan orang mengakses internet menggunakan ponsel. Selain itu, Android dirancang untuk memudahkan developer atau pengembang aplikasi dengan batasan yang minim agar lebih berkembang (Andry,2011).

Dengan konsep *Open Source* artinya pengembang bebas dalam memodifikasi sisstem operasi ini, tidak ada ketentuan yang mutlak dalam konfigurasi *Software* dan *Hardware*. Dalam waktu hampir 11 tahun Android telah mengeluarkan 15 versi, diantaranya:

1. ***Astro boy***, dirilis 23 September 2008.
2. ***Bender***, Android yang memiliki kode 1.1.
3. ***Cupcake 1.5***, dirilis Mei 2009.
4. ***Donut***, 1.6 dirilis September 2009.
5. ***Éclair 2.0 dan 2.1***
6. ***Froyo***, dirilis Mei 2010.
7. ***Gingerbread 2.3***, dirilis Desember 2010.
8. ***Honeycomb***, dirilis Mei 2011.
9. ***Ice Cream Sandwich 4.0***, dirilis Desember 2011.
10. ***Jelly Bean***, dirilis Juli 2012.
11. ***KitKat 4.4***, dirilis Oktober 2013.
12. ***Lollipop*** 5.x, dirilis Oktober 2014.
13. ***Marshmallow 6.x***, dirilis Agustus 2015.
14. ***Nougat 7.x***, dirilis Agustus 2016.
15. ***Oreo 8.x***, dirilis Agustus 2017.

2.12 Android Studio

Android Studio adalah *Integrated Development Enviroment* (IDE) ytang dibuat untuk pengkodean Android, dibangun diatas perangkat lunak *JetBrains IntelliJ IDEA* dan didesain secara khusus untuk pengembangan OS Android.

IDE ini adalah pengganti dari *Eclipse* Android *Development Tools* (ADT), sebelumnya merupakan *base* IDE guna pengembangan aplikasi Android. Android

studio pertama sekali diumumkan di *Google I/O conference* 16 Mei 2013. Ini adalah tahap *preview* dari versi 0.1 pada bulan Mei 2013, dan telah memasuki tahap BETA sejak versi 0.8 hingga rilis pada bulan Juni 2014.

Versi stabil pertama dirilis pada Desember 2014, dimulai sejak versi 1.0. Versi stabil hingga kini adalah versi 3.13 yang rilis pada bulanJuni 2018.

Fitur Fitur yang tersedia saat ini dalam stable version:

a) Dukungan *Gradle-based build.*

b) Android-*specific refactoring* dan perbaikan cepat.

c) *Lint tools* untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya.

d) Integrasi *Proguard* dan kemampuan penananda tanganan aplikasi.

e) *Template-based wizards* untuk membuat *template design* umum seperti *drawer* atau *empty activity.*

f) Mendukung untuk pengembangan aplikasi Android *Wear*.

g) Editor tata letak yang memungkinkan pengguna untuk menyeret dan menjatuhkan *(drag-and-drop)* komponen UI, opsi untuk melihat tata letak pada beberapa konfigurasi layer.

h) Dukungan bawaan untuk *Google Cloud Platform*, memungkinkan integrasi dengan *Firebase Cloud Messaging* (‘Perpesanan Google Cloud’ Sebelumnya) dan *Google App Engine.*

i) Android *Virtual Device* (Emulator) untuk menjalankan dan men-*debug* aplikasi di Android Studio.

2.13 Android *Software Development Kit* (SDK)

Android SDK adalah tools API *(Application Programming Interface)* yang diperlukan untuk pengembangan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi *Operating System* , *middleware* dan aplikasi kunci yang akan di-*release* oleh pihak Google.

Saat ini disediakan Android SDK *(Software Development Kit)* sebagai sebuah *tool* API untuk mulai pengembangan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi yang netral, Android memberikan kesempatan kepada pengembang aplikasi untuk membuat aplikasi yang dibutuhkan, bukan hanya aplikasi bawaan *(stock* Android *apps)* pada *Smartphone* pengguna (Developers,2014).

2.14 *Application Programming Interface* (API)

Antarmuka pemrograman aplikasi *(Application Programming Interface/API)* adalah sekumpulan perintah, fungsi, dan *protocol* yang dapat digunakan oleh *programmer* ketika membangun sebuah perangkat lunak untuk system operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

Pada program sederhana saja , akan dibutuhkan setidaknya ribuan *system calls*/detik, oleh karenanya para *programmer* membuat aplikasi dengan menggunakan *Application Programming Interface* (API). Dalam API terdapat fungsi-fungsi/perintah-perintah untuk menggantikan bahasa yang digunakan dalam *system calls* dengan penyajian bahasa yang terstruktur dan mudah untuk dimengerti oleh *programmer*. Fungsi yang dibuat menggunakan API tersebut akan memanggil *system calls* sesuai dengan sistem operasinya, tidak tertutup kemungkinan penamaan dari *system calls* sama dengan penamaan di API.

API menyediakan fungsi dan perintah dengan bahasa yang lebih terstruktur dan mudah untuk dipahami oleh *programer* bila dibandingkan dengan *System Calls*, hal ini sangat penting untuk aspek *editing* dan pengembangan, sehingga para *programmer* dapat melakukan pengembangan sistem dengan mudah. API dapat diimplementasikan pada semua Sistem Operasi dengan syarat sudah tersedia paket API nya.

API menjelaskan cara kerja sebuah tugas *(task)* tertentu untuk dilakukan. Dalam pemrograman prosedural seperti bahasa C, aksi biasanya dilakukan dengan media pemanggilan fungsi.

Keuntungan memprogram dengan menggunakan API adalah:

1. Portabilitas. Programmer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah ter- *install* API tersebut. Sedangkan system call berbeda antar sistem operasi, dengan catatan dalam implementasinya mungkin saja berbeda.
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 user 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 kita gunakan.

2.15 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon seluler. Bahasa ini awalnya dibuat oleh James Gosling dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal.

Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam *p-code (bytecode)* dan dapat dijalankan pada berbagai Mesin *Virtual Java* (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik *(general purpose),* dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan

aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi.

Kelebihan:

a) *Multiplatform*, Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi computer. Platform yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mac OS dan Sun Solaris.

b) OOP, (*Object Oriented Programming* - Pemrogram Berorientasi Objek), Java merupakan salah satu bahasa pemrograman dengan konsep OOP, yang artinya program yang dibangun berorientasikan kepada Objek. Aplikasi yang dibangun dengan konsep OOP terdiri dari *object-object* yang akan saling berhubungan.

c) *Library Class* yang lengkap, Java terkenal dengan kelengkapan library/perpustakaan yang akan memudahkan *programer* ketika membangun sebuah aplikasi, kelengkapan *library* ini bahakan di dukung oleh keberadaan komunitas Java yang besar dan terus menerus membuat *library-library* baru untuk men-*support* kebutuhan programer dalam pembuatan aplikasi.

d) Bergaya seperti C++, Java memiliki sintaks yang mirip dengan bahasa pemrograman C++ sehingga menarik bagi p*rogramer* C++ dan berkeinginan pindah ke Java. Saat ini saja pengguna Java sudah sangat banyak, sebagian besar diantaranya adalah pemrogram C++ yang pindah ke Java.

Kekurangan:

Masih ditemui beberapa permasalah kompabilitas antara platform satu dengan lainnya. Untuk J2SE, misalnya SWT-AWT *bridge* sampai saat ini belum berfungsi pada Mac OS X.

Mudah di-*decompile*. Dekompilasi adalah sebuah proses membalikkan dari kode siap pakai menjadi kode sumber *(source code).* Hal ini disebabkan karena kode Java merupakan *bytecode* yang menyimpan banyak atribut-atribut bahasa tingkat tinggi, misalnya nama kelas, metode, dan tipe data.

Penggunaan memori jauh lebih besar jika dibandingkan dengan bahasa pemrograman generasi sebelumnya seperti C/C++, Pascal (Delphi dan Object Pascal). Tetapi biasanya ini bukan merupakan masalah bagi pihak yang selalu memakai teknologi terbaru dengan alasan haraga memori(RAM) terpasang makin murah, tetapi menjadi masalah bagi pihak yang masih berkutat dengan komputer berumur lebih dari 4 tahun.

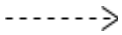
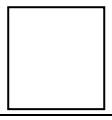
2.16 *Use Case Diagram*

Use case diagram atau dalam bahasa Indonesia dikenal dengan diagram *use case* merupakan salah satu diagram yang ada dalam UML *(Unifield Modeling Language)* yang digunakan untuk memodelkan aspek perilaku sistem dari sistem yang akan dibuat dan untuk merekam persyaratan fungsional sebuah sistem.

Rosa A. S dan M. Salahuddin Mengungkapkan bahwa (2012 : 155) "*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat." Dua hal utama yang harus ada pada *use case* menurut Rosa A. S dan M. Salahuddimn adalah :

1. Aktor merupakan orang, proses, atau aplikasi lain yang berinteraksi dengan aplikasi yang akan dibuat di luar aplikasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsi-fungsi/proses-proses yang disediakan aplikasi sebagai unit-unit yang saling bertukar pesan/berinteraksi antar unit/proses atau aktor.

Tabel II-1 Simbol *Use Case Diagram*

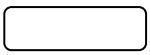
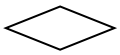
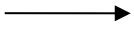
Simbol	Nama	Keterangan
	*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
	*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
	*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
	*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
	Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
	*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
	*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

2.17 *Activity Diagram*

UML menyediakan *activity diagram* sebagai pendukung *use case* yang telah dibuat dalam proses sebelumnya Dalam diagram aktivitas, masing masing alir memiliki awal, decision yang akan terjadi pada sistem dan hasil akhir yang terjadi didalam sistem tersebut. Pada dasarnya diagram aktivitas memiliki struktur yang hampir mirip dengan diagram alir *flowchart*.

Aktivitas pada suatu diagram menggunakan kotak yang berisi dan lengkung untuk menggambarkan fungsi-fungsi tertentu yang ada dalam sebuah sistem, sedangkan tanda panah menggambarkan sebuah aliran dalam sistem tersebut.

Tabel II-2 Tabel *Activity Diagram*

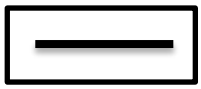
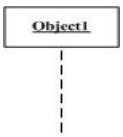
Simbol	Nama Elemen	Keterangan
	*Start state*	Titik awal atau permulaan
	*End state*	Titik akhir atau akhir dari aktivitas
	*Activity*	*Activity* atau aktivitas yang dilakukan oleh*actor*
	*Decision*	Pilihan untuk mengambil keputusan
	*Interaction*	Alur

2.18 *Sequence Diagram*

Sequence Diagram di UML terutama digunakan untuk memodelkan interaksi antara actor dan objek dalam system dan interaksi antara obyek itu sendiri. UML memiliki sintaks yang kaya untuk *Sequence Diagram*, yang memungkinkan berbagai jenis interaksi yang dimodelkan. Sesuai namanya, *Sequence Diagram* menunjukkan urutan interaksi yang terjadi antara use case. *Sequence Diagram* memiliki dua buah karakteristik yaitu:

1. Setiap objek memiliki *lifeline* yang digambarkan dengan garis putus-putus vertikal dan garis ini menunjukkan daur hidup dari sebuah objek.
2. Terdapat *focus control* yang digambarkan dengan sebuah persegi panjang yang tipis dan tinggi. Fokus kontrol ini menunjukkan periode waktu selama sebuah objek melakukan sebuah event.

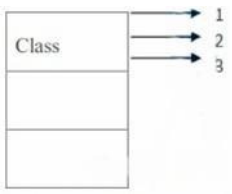
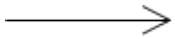
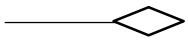
Tabel II-3 Tabel *Sequence Diagram*

Simbol	Nama Elemen	Keterangan
	*Life Line*	Objek entitas, antar muka yang saling berinteraksi
	*Activation*	Menggmbarkan hubungan antar objek dengan *message*
	*Message (call)*	Menggambarkan alur message yang merupakan kejadian objek pengrim *life line* ke objek penerima *life line*
	*Message (return)*	Menggambarkan alur pengambilan *message* ke objek pemanggil dan tanda bahwa objek penerima telah menyelesaikan prosesnya.
	*Object*	Object adalah instance dari sebuah class yang dituliskan tersusun secara horizontal diikuti lifeline

2.19 *Class Diagram*

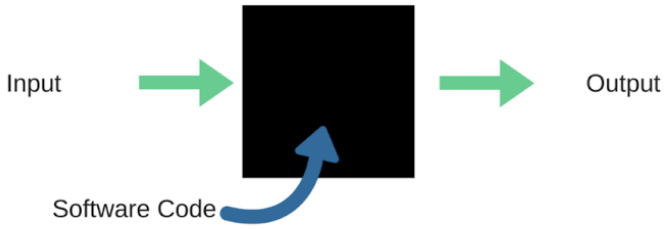
Class-class yang ada pada sbuah sistem yang saling berhubungan digambarkan dengan sebuah diagram yang disebut *Class Diagram*. Diagram ini menggambarkan alur struktur statis dari sebuh system, oleh sebab itu *Class Diagram* menjadi sangat penting pada setiap metode berorientasi objek termasuk UML *(Unified Modelling Language).*

Tabel II-4 Tabel *Class Diagram*

Simbol	Nama	Keterangan
	*Class*	Simbol untuk membangun sebuah pemrograman dengan objek Terdiri 3 bagian, bagian atas adalah nama kelas, bagian tengah adalah atribut dan bagian bawah adalah metode dari kelas tersebut.
	Generalisasi	Simbol yang menandakan adanya generalisasi dari kelas input untuk menghasilkan data yang dibutuhkan
	Asosiasi berarah/ *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
	Kebergantungan /*dependency*	Relasi antar kelas dengan makna ketergantungan antar kelas
	Agresi/ *aggregation*	Relasi antar kelas dengan makna semua – bagian *(whole-part)*

2.20 *Black Box Testing*

Black Box Testing adalah metode pengujian pada perangkat lunak yang bertujuan menguji fungsionalitas suatu aplikasi yang bertentangan dengan struktur internal atau kerja. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan. Menggunakan deskripsi eksternal perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus.



Gambar II-5 Konsep *Black Box Testing*

Pada Gambar II-5 dapat terlihat bagaimana konsep dari sebuah tes pengujian menggunakan metode *Black Box Testing*. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih *input* yang *valid* dan tidak *valid* dan menentukan *output* yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga.