BAB II

TINJAUAN PUSTAKA

II.1. Uraian Teoritis

II.1.1. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus komputer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu tehnik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, "Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu". (Juansyah, Andi ; 2015 : 2)

II.1.2. Algoritma *Levenshtein*

Algoritma *Levenshtein* ditemukan oleh ilmuan asal Rusia bernama Vladimir Levenshtein pada tahun 1963, algoritma ini juga disebut dengan algoritma *Edit Distance*. Perhitungan *editdistance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan *string* antara dua *string*, sebagai contoh hasil penggunaan algoritma ini, *string* "komputer" dan "*computer*" memiliki *distance* 1 karena hanya perlu dilakukan satu operasi saja untuk

mengubah satu *string* ke *string* yang lain. Dalam kasus dua *string* di atas, *string* "*computer*" dapat menjadi "komputer" hanya dengan melakukan satu penukaran karakter "c" menjadi "k".

Algoritma *Levenshtein* digunakan secara luas dalam berbagai bidang, misalnya mesin pencari, pengecek ejaan (*spellchecking*), pengenal pembicaraan (*speechrecognition*), pengucapan dialek, analisis DNA, pendeteksi pemalsuan, dan lain-lain. Algoritma ini menghitung jumlah operasi *string* paling sedikit yang diperlukan untuk mentransformasikan suatu *string* menjadi *string* yang lain. Algoritma *Levenshtein* bekerja dengan menghitung jumlah minimum pentranformasian suatu *string* menjadi *string* lain yang meliputi penghapusan, penyisipan, dan penukaran. Selisih perbedaan antar *string* dapat diperoleh dengan memeriksa apakah suatu *string* sumber sesuai dengan *string* target. Nilai selisih perbedaan ini disebut juga *edit distance* atau jarak *Levenhstein*. Jarak *Levenshtein* antar *string* "s" dan *string* "t" tersebut adalah fungsi D yang memetakan (s,t) ke suatu bilangan *realnonnegatif*, sebagai contoh diberikan dua buah *string* s = s(1)s,(2),s(3),...,s(m) dan t = t(1),t(2),t(3),...,t(n) dengan | s | = m dan | t | = n sepanjang *alfabet* V berukuran r sehingga "s" dan "t" anggota dari V*. s(j) adalah karakter pada posisi ke-j pada *string* "s" dan t(i) adalah karakter pada posisi ke-i pada *string* "t". Sehingga jarak *Levenshtein* dapat didefinisikan sebagai.

$$D(s,t) = d(s1,t1) + d(s2,t2) + \ldots + d(s1,t1) \ldots\ldots\ldots\ldots (1)$$

D(s,t) adalah banyaknya operasi minimum dari operasi penghapusan, penyisipan dan penukaran untuk menyamakan *string* s dan t. Pada implementasi pencocokan antar *string*, ketiga operasi tersebut dapat dilakukan sekaligus untuk

menyamakan *string* sumber dengan *string* target. (Arnawa, Ida Bagus Ketut Surya ; 2017 : 47)

II.1.3. Pemrograman *Java*

Java adalah sebuah bahasa pemrograman yang populer dikalangan para akademisi dan praktisi komputer. *Java* pertama kali dikembangkan untuk memenuhi kebutuhan akan sebuah bahasa komputer yang ditulis satu kali dan dapat dijalankan dibanyak sistem komputer berbeda tanpa perubahan kode berarti. Pada umumnya, para pakar pemrograman berpendapat bahwa bahasa *Java* memiliki konsep yang konsisten dengan teori pemrograman objek dan aman untuk digunakan.

Java sampai saat ini masih merupakan bahasa pemrograman yang masih sangat di minati dan banyak digunakan oleh para *programer* dan *software developer* untuk mengembangkan berbagai tipe aplikasi, mulai dari aplikasi *console*, aplikasi *desktop*, *game*, dan *applet* (aplikasi yang berjalan di lingkungan *webbrowser*), sampai ke aplikasi-aplikasi yang berskala *enterprise*. Untuk memenuhi kebutuhan tipe aplikasi yang beragam tersebut, *Java* dikategorikan menjadi tiga edisi, yaitu : J2SE (*Java* 2 *Platform Standart Edition*) untuk membuat aplikasi-aplikasi *desktop* dan *apllet*, J2EE (*Java* 2 *Platform Enterprise Edition*) untuk membuat aplikasi-aplikasi *multitier* berskala *enterprise*, dan J2ME (*Java* 2 *Platform Micro Edition*) untuk membuat aplikasi-aplikasi yang dapat dijalankan dilingkungan perangkat-perangkat mikro seperti *handphone*, PDA dan *Smartphone*. (Wardani dan Yaqin ; 2013 : 473-477)

II.1.3.1.Edisi *Java*

Sebagian besar bahasa pemrograman modern berdiri di atas pustaka kelas-kelas yang telah ada untuk mendukung fungsionalitas. Pada bahasa *Java*, kelompok-kelompok kelas yang berkaitan erat dimasukkan di satu paket, bervariasi sesuai edisi *Java*. Masing-masing paket untuk maksud tertentu :*applet*, aplikasi standar, skala *enterprise*, dan produk konsumer.

Java adalah bahasa yang dapat dijalankan di sembarang *platform*, di beragam lingkungan :*internet*, *consumer electronic products*, dan *computer application*. *The Java* 2 *Platform* tersedia dalam tiga edisi untuk keperluan berbeda berikut :

1. *Java* 2 *Standard Edition* (J2SE)
2. *Java* 2 *Enterprise Edition* (J2EE)
3. *Java* 2 *Micro Edition* (J2ME).

Pada pengembangan *enterpris eapplications*, kita menggunakan sejumlah besar paket. Pada *consumer electronic sproduct*, hanya sejumlah kecil bagian bahasa yang digunakan. Masing-masing edisi berisi *Java* 2 *Software Development Kit* (SDK) untuk mengembangkan aplikasi dan *Java* 2 *Runtime Environment* (JRE) untuk menjalankan aplikasi. (Hariyanto, Bambang ; 2014 : 3)

II.1.4. Aplikasi *Mobile*

Aplikasi *mobile* dapat diartikan sebagai sebuah produk dari sistem komputasi *mobile*, yaitu sistem komputasi yang dapat dengan mudah dipindahkan

secara fisik dan yang komputasi kemampuan dapat digunakan saat mereka sedang dipindahkan. Contohnya adalah *personal digital assistant* (PDA), *smartphone* dan ponsel. (Ramadhan dan Utomo ; 2014 : 47-55)

II.1.5. *Android*

Menurut Arifianto Teguh, *Android* adalah sebuah *platform* pertama yang betul-betul terbuka dalam pengembangannya dan komperehensif untuk perangkat *mobile,* semua perangkat lunak yang ada difungsikan menjalankan sebuah *device mobile* tanpa memikirkan kendala kepemilikan yang menghambat inovasi pada teknologi *mobile*. Dalam definisi lain, *Android* merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware,* dan aplikasi inti yang dirilis oleh *Google*. Sedangkan *Android SDK (Software Development Kit)* menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform Android* dengan menggunakan bahasa pemrograman *Java*.

Aplikasi *Android* ditulis dalam bahasa pemrograman *java*, yaitu kode *java* yang terkompilasi bersama-sama dengan data dan *file-file* sumber yang dibutuhkan oleh aplikasi yang digabungkan oleh *app tools* menjadi paket aplikasi *Android*, sebuah *file* yang ditandai dengan akhiran *.apk.file* inilah yang didistribusikan sebagai aplikasi dan diinstal pada *handset Android*. *File* ini diunduh oleh pengguna ke perangkat *mobile* mereka. Semua kode dijadikan satu *fileapk,* dan kemudian kita sebut sebagai sebuah aplikasi.(Ahmad : 2015 : 190-200)

II.1.5.1. Fitur-Fitur *Android*

Sejak pertama kali diperkenalkan, *Android* telah dilengkapi dengan beberapa fitur utama yang harus dimiliki oleh *smartphone* maupun perangkat komunikasi pintar lainnya. Fitur-fitur yang disediakan terus dikembangkan seiring dengan perkembangan teknologi *Android* yang ditandai dengan dirilisnya versi-versi baru *Android*. Berikut adalah beberapa fitur yang disediakan oleh *Android* dan dapat dimanfaatkan oleh pengembang dalam aplikasi yang dibangunnya :

1. Media penyimpanan (*storage*) – untuk media penyimpanan internal, *Android* menggunakan SQLite. Selain itu, *Android* juga mendukung fasilitas penyimpanan data di *external* storage seperti *SD card*.
2. Koneksi jaringan (*connectivity*) – *Android* mendukung GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, dan WiMAX.
3. *Messaging* – *Android* mendukung SMS dan MMS.
4. *Web browser* – *Web browser* bawaan *Android* dibangun berbasis pada open source blink (sebelumnya *WebKit*) dan *Chrome's* V8 *JavaScript engine*.
5. Dukungan media (media support) – *Android* mendukung beragam jenis media, seperti H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, MP3, MIDI, Ogg, Vorbis, WAV, JPEG, PNG, GIF, dan BMP.
6. Dukungan perangkat keras (*hardware support*) – *Android* menyediakan beragam *hardware* yang dapat digunakan oleh pengembang aplikasi, seperti *accelerometer* sensor, *gyroscopes, barometer, magnetometer, thermometer, camera, digital compass*,*proximity and pressure sensors*, dan GPS.
7. *Multitouch* – mendukung *multitouch screen*.

8. *Multitasking* – mendukung multitasking *applications.*
9. Dukungan *Flash* – sejak *Android* 2.3 mendukung *Flash* 10.1.
10. *Tethering* – mendukung *sharing* koneksi *internet* sebagai *wired /wirelesshotspot*.
11. Fitur-fitur berbasis suara (*voice-based feature*) – *Android* menyediakan beragam aplikasi berbasis suara, seperti *Google Search*, *Google's KnowledgeGraph*, dan lain sebagainya.
12. Tangkapan layar (*screen capture*) – sejak *Android* 4.0, pengguna *Android* dapat mengambil *screenshot* dengan menekan tombol *Power + Volume Down* pada perangkatnya.(m) Aksesibilitas (*accessibility*) – *Android* menyediakan *built-in text-to-speech* dari *Talk Back* dan berbagai fasilitas lain untuk memberikan kemudahan bagi semua pengguna *Android*, terutama dari kalangan difabel.
13. *Handset layouts* – aplikasi *Android* dapat berjalan pada beragam ukuran layar dan dapat dikoneksikan ke *externalscreen* melalui HDMI atau *Miracast* (*wirelessly*). (Hansun, Seng ; 2018 : 3-4)

II.1.6. *Android Studio*

Androidstudio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi *Android* dan bersifat *opensource* atau gratis. Peluncuran *Android Studio* ini diumumkan oleh *Google* pada 16 mei 2013 pada *eventGoogle* I/O *Conference* untuk tahun 2013. Sejak saat itu, *Android Studio* mengantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi *Android.*

Android studio sendiri dikembangkan berdasarkan Intellij IDEA yang mirip dengan *Eclipse* disertai dengan ADT *plugin* (*Android Development Tools*). *Android studio* memiliki fitur :

a. Projek berbasis pada *Gradle Build*
b. *Refactory* dan pembenahan *bug* yang cepat
c. *Tools* baru yang bernama "*Lint*" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
d. Mendukung *Proguard And App-signing* untuk keamanan.
e. Memiliki GUI aplikasi *android* lebih mudah
f. Didukung oleh *Google Cloud Platfrom* untuk setiap aplikasi yang dikembangkan. (Andi Juansyah ; 2015)

II.1.7. *Android* SDK (*Software Development Kit*)

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. *Android* SDK terdiri dari *debugger*, *libraries*, *handsetemulator*, dokumentasi, contoh kode program dan tutorial. Saat ini *Android* sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk *desktop* modern), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista.Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau lebih. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan plugin *Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit *dokumen Java* dan XML serta menggunakan peralatan *commandline* untuk menciptakan, membangun, melakukan *debug* aplikasi

Android dan pengendalian perangkat *Android* (misalnya *reboot*, menginstal paket perangkat lunak). (Sulihati dan Andriyani ; 2016 : 20)

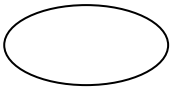
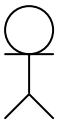
II.1.8. Pengertian UML

UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodalan umum dalam industri perangkat lunak dan pengembangan sistem (Windu dan Grace ; 2013 : 81). Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.1.8.1.*Use Case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. (Windu dan Grace ; 2013)

Tabel II.1. *Use Case* Diagram

Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah *Abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktir, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *Use Case*, tetapi tidak memiliki kontrol terhadap *use case*
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat

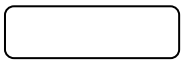
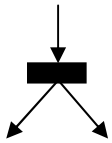
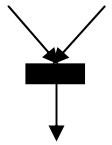
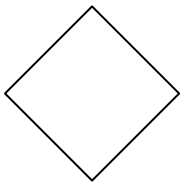
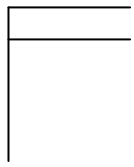
(Sumber : Ade Hendini ; 2016)

II.1.8.2.*Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Windu dan Grace ; 2013 : BIT. 10. 80-87).

Tabel II.2. *Activity Diagram*

Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas

	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

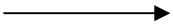
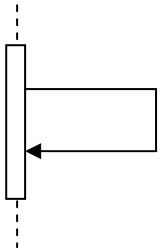
(Sumber : Ade Hendini ; 2016)

II.1.8.3.*Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Windu dan Grace ; 2013).

Tabel II.3. *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti

	tampilan *formentry* dan *form* cetak
	*Control Class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
	*Message*, simbol mengirim pesan antar *class*
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*,mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

(Sumber : Ade Hendini ; 2016)