BAB 2
LANDASAN TEORI

2.1 Asisten Keuangan

Pengertian asisten menurut Kamus Besar Bahasa Indonesia (KBBI) adalah orang yang bertugas membantu orang lain dalam melaksanakan tugas profesional, misalnya dalam pekerjaan, profesi, dan kedinasan. Sedangkan keuangan menurut Kamus Besar Bahasa Indonesia (KBBI) adalah seluk-beluk uang atau urusan (masalah) uang. Dan pengertian uang menurut Kamus Besar Bahasa Indonesia (KBBI) adalah alat tukar atau standar pengukur nilai (kesatuan hitungan) yang sah, dikeluarkan oleh pemerintah suatu negara berupa kertas, emas, perak, atau logam lain yang dicetak dengan bentuk dan gambar tertentu. Jadi pengertian asisten keuangan adalah orang yang bertugas membantu orang lain dalam melaksanakan tugas profesional, dalam pekerjaan masalah uang [4].

2.2 Aplikasi Mobile

Aplikasi *mobile* saat ini sangat dibutuhkan karena alat-alat telekomunikasi yang tersebar di seluruh dunia membutuhkan aplikasi-aplikasi yang dapat memepermudah pekerjaan penggunanya dimanapun dan kapanpun terutama dalam hal informasi. Aplikasi ini dapat diakses melalui perangkat nirkabel seperti *pager*, telepon seluler dan PDA.

2.2.1 Aplikasi

Aplikasi adalah satu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas. Misalnya termasuk perangkat lunak perusahaan, *software* akuntansi, perkantoran, grafis perangkat lunak dan pemutar media. Dapat disimpulkan bahwa aplikasi merupakan *software* yang berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data [5].

Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh

aplikasi ialah program pemroses kata dan *Web Browser*. Aplikasi akan menggunakan Sistem Operasi (OS) komputer dan aplikasi yang lainnya yang mendukung.

Klasifikasi aplikasi dapat dibagi menjadi 2 (dua) yaitu :

1. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
2. Aplikasi paket, dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

2.2.2 Mobile

Mobile dapat diartikan sebagai perpindahan yang mudah dari satu tempat ke tempat yang lain, misalnya telepon *mobile* berarti bahwa terminal telepon yang dapat berpindah dengan mudah dari satu tempat ke tempat lain tanpa terjadi pemutusan atau terputusnya komunikasi. Adapun karakteristik perangkat *mobile* yaitu :

1. Ukuran yang kecil : Perangkat *mobile* memiliki ukuran yang kecil. Konsumen menginginkan perangkat yang terkecil untuk kenyamanan dan mobilitas mereka.
2. *Memory* yang terbatas : Perangkat *mobile* juga memiliki *memory* yang kecil, yaitu *primary* (*RAM*) dan *secondary* (*disk*).
3. Daya proses yang terbatas : Sistem *mobile* tidaklah setangguh rekan mereka yaitu *desktop*.
4. Mengkonsumsi daya yang rendah : Perangkat *mobile* menghabiskan sedikit daya dibandingkan dengan mesin *desktop*.
5. Kuat dan dapat diandalkan : Karena perangkat *mobile* selalu dibawa kemana saja, mereka harus cukup kuat untuk menghadapi benturan-benturan, gerakan, dan sesekali tetesan-tetesan air.
6. Konektivitas yang terbatas : Perangkat *mobile* memiliki *bandwith* rendah, beberapa dari mereka bahkan tidak tersambung.
7. Masa hidup yang pendek : Perangkat-perangkat konsumen ini menyala dalam hitungan detik kebanyakan dari mereka selalu menyala.

2.3 Android

Pembahasan mengenai *android* dan *tools* yang digunakan dalam pembuatan aplikasi, akan dijelaskan pada sub bab berikut :

2.3.1 Pengertian Android

Android adalah sebuah sistem operasi pada *handphone* yang bersifat terbuka dan berbasis pada sistem operasi *Linux*. *Android* bisa digunakan oleh setiap orang yang ingin menggunakannya pada perangkat mereka. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri yang akan digunakan untuk bermacam peranti bergerak.

Awalnya, *Google Inc*. membeli *Android Inc*., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google*, *HTC*, *Intel*, *Motorola*, *Qualcomm*, *T-Mobile*, dan *Nvidia*.

Pada saat perilisan perdana *Android*, 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, *Google* merilis kode–kode *Android* di bawah lisensi *Apache*, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. [6]

2.3.2 Sejarah dan Perkembangan Android

Pada mulanya terdapat berbagai macam sistem operasi pada perangkat selular, diantaranya sistem operasi *Symbian*, *Microsoft Windows Mobile*, *Mobile Linux*, *iPhone*, dan sistem operasi lainnya. Namun diantara sistem operasi yang ada belum mendukung standar dan penerbitan *API* yang dapat dimanfaatkan secara keseluruhan dan dengan biaya yang murah. Kemudian *Google* ikut berkecimpung didalamnya dengan *platform Android*, yang menjanjikan keterbukaan, keterjangkauan, *open source* dan *framework* berkualitas.

Pada tahun 2005, *Google* mengakuisisi perusahaan *Android Inc*. untuk mamulai perkembangan *platform Android*. Dimana terlibat pengembangan ini

adalah Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada pertengahan 2007 sekelompok pemimpin industri bersama-sama membentuk analisis aliansi perangkat selular terbuka, *Open Handset Alliance* (*OHA*). Bagian dari tujuan aliansi ini adalah berinovasi dengan cepat dan menaggapi kebutuhan konsumen dengan lebih baik, dengan produk awalnya adalah *platform Android*. Dimana *Android* dirancang untuk melayani kebutuhan operator telekomunikasi, manufaktur *handset*, dan pengembangan aplikasi, *OHA* berkomitmen untuk membuat *Android open source* dengan lisensi *Apache Versi 2.0*. [6]

2.3.3 Versi Android

Android pertama kali diluncurkan pada 5 November 2007, dan *smartphone* pertama yang menggunakan sistem operasi *Android* dikeluarkan oleh *T-Mobile* degnan sebutan G1 pada bulan September 2008. Hingga saat ini *Android* telah merilis beberapa versi *Android* untuk menyempurnakan versi sebelumnya. Selain berdsarkan penomoran, pada setiap versi *Android* terdapat kode nama berdasarkan nama-nama kue. Hingga saat ini sudah terdapat beberapa versi yang telah diluncurkan, diantaranya:

1. Android 1.0 (API level 1) dirilis pada 23 September 2008.
2. Android 1.1 (API level 2) dirilis pada 09 Februari 2009.
3. Android 1.5 Cupcake (API level 3) dirilis pada 30 April 2009.
4. Android 1.6 Donut (API level 4) dirilis pada 15 September 2009.
5. Android 2.0 Eclair (API level 5) dirilis pada 26 Oktober 2009.
6. Android 2.0.1 Eclair (API level 6) dirilis pada 3 Desember 2009.
7. Android 2.1 Eclair (API level 7) dirilis pada 12 Januari 2010.
8. Android 2.2 s/d 2.2.3 Froyo (API level 8) dirilis pada 20 Mei 2010, 18 Januari 2011, 22 Januari 2011, 21 November 2011.
9. Android 2.3 s/d 2.3.2 Gingerbread (API level 9) dirilis pada 6 Desember 2010, Desember 2010, Januari 2011.
10. Android 2.3.3 s/d 2.3.7 Gingerbread (API level 10) dirilis pada 9 Februari 2011, 28 April 2011, 25 Juli 2011, 2 September 2011, 21 September 2011.

11. Android 3.0 Honeycomb (API level 11) dirilis pada 22 Februari 2011.
12. Android 3.1 Honeycomb (API level 12) dirilis pada 10 Mei 2011.
13. Android 3.2 Honeycomb (API level 13) dirilis pada 15 Juli 2011, 20 September 2011, 30 Agustus 2011, Desember 2011, Januari 2012, Februari 2012.
14. Android 4.0 s/d 4.0.2 Ice Cream Sandwich (API level 14) dirilis pada 19 Oktober 2011, 21 Oktober 2011, 28 November 2011.
15. Android 4.0.3 s/d 4.0.4 Ice Cream Sandwich (API level 15) dirilis pada 16 Desember 2011, 29 Maret 2012.
16. Android 4.1 s/d 4.1.2 Jelly Bean (API level 16) dirilis pada 9 Juli 2012, 23 Juli 2012, 9 Oktober 2012.
17. Android 4.2 s/d 4.2.2 Jelly Bean (API level 17) dirilis pada 13 November 2012, 27 November 2012, 11 Februari 2013.
18. Android 4.3 Jelly Bean (API level 18) dirilis pada 24 Juli 2013.
19. Android 4.4 KitKat (API level 19) dirilis pada Oktober 2013. [6]

2.3.4 Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya *Symbian*, *iPhone*, *Windows Mobile*, *BlackBerry*, *Java Mobile Edition*, *Linux Mobile* (*LiMo*), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan *Android*. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada *platform* lain, *Android* adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap system karena berbasiskan *Linux* dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar *royality*. Sementara pengembang *software* menyukai karena *Android* dapat digunakan diperangkat manapun dan tanpa terikat oleh vendor manapun.
2. Arsitektur komponen dasar *Android* terinspirasi dari teknologi internet Mashup. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi

lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.

3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, *database SQL*, *browser* dan penggunaan peta. Semua itu sudah tertanam pada *android* sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja *system* menjadi lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.
5. Dukungan grafis dan suarat terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh *Flash* menyatu dalam 3D menggunakan *OpenGL* memungkinkan membuat aplikasi maupun *game* yang berbeda.

Portabilitas aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman *Java* dan dieksekusi oleh mesin. [6]

2.4 Java

Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di *Sun Microsystems* saat ini merupakan bagian dari *Oracle* 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 berbasis *web*. [7]

2.4.1 Edisi Java

Java adalah bahasa yang dapat dijalankan dimanapun dan disembarang *platform* apapun, diberagam lingkungan : *internet*, *consumer Electronic products* dan *computer Applications*. *The Java 2 platform* tersedia dalam 3 edisi untuk keperluan berbeda. Untuk beragam aplikasi yang dibuat dengan bahasa *java*, *java* dipaketkan dalam edisi-edisi berikut :

1. Java 2 Standard Edition (J2SE)

   The Java 2 Standard Edition menyediakan lingkungan pengembangan yang kaya akan fitur, stabil, aman dan *cross-platform*. Edisi ini mendukung konektivitas basis data, rancangan antar muka pemakai, masukan/keluaran dan pemograman jaringan dan termasuk sebagai paket-paket dasar bahasa *Java*.

2. Java 2 Enterprise Edition (J2EE)

   The Java Enterprise Edition menyediakan *tool* untuk membangun dan menjalankan *multitier enterprise applications*. *J2EE* bersisi paket-paket di *J2SE* ditambah paket-paket untuk mendukung pengembangan *Enterprise Java Beeans*, *Java Servlets*, *Java Server Pages*, *XML* dan kendali transaksi yang fleksibel.

3. Java 2 Micro Edition (J2ME)

   The Java 2 Micro Edition selain menyedikan bahasa *Java* yang sama, unggul dalam portabilitas (kemampuan dapat dijalankan dimanapun), *safe network delivery*, seperti *J2SE* dan *J2EE*. Aplikasi aplikasi dapat diskalakan (dimampukan) agar dapat bekerja dengan *J2SE* dan *J2EE*.

   J2ME untuk beragam *consumer electronic product* seperti *pager*, *smart card*, *cell phone*, *handled PDA*, dan *set top box*. [8]

2.4.2 Gambaran Java 2 Micro Edition (J2ME)

The Java 2 Micro Edition selain menyedikan bahasa *Java* yang sama, unggul dalam portabilitas (kemampuan dapat dijalankan dimanapun), *safe network*

delivery, seperti *J2SE* dan *J2EE*. Aplikasi aplikasi dapat diskalakan (dimampukan) agar dapat bekerja dengan *J2SE* dan *J2EE*.

Program *Java ME*, seperti semua program *Java* adalah diterjemahkan oleh *Virtual Machine* (*VM*). Program-program tersebut dikompile ke dalam *bytecode* dan diterjemahkan oleh *Java Virtual Machine* (*JVM*). Ini berarti bahwa program-program tersebut tidak berhubungan langsung dengan perangkat keras. *Java ME* menyediakan suatu *interface* yang sesuai dengan perangkat keras. Aplikasi-aplikasi tersebut cukup dikompile sekali dan mampu dijalankan pada mesin yang berbeda. Inti dari *Java ME* terletak pada *configuration* dan *profile-profile*.

Suatu *configuration* menggambarkan lingkungan *runtime* dasar dari suatu sistem *Java ME*. Ia menggambarkan *core library*, *virtual machine*, fitur keamanan dan jaringan [8].



Gambar 2. 1 *Arsitektur Java ME*

Sebuah *profile* memberikan *library* tambahan untuk suatu kelas tertentu pada sebuah perangkat. *profile-profile* menyediakan *user interface* (*UI*) *API*, *persistence*, *messaging library*, dan sebagainya. Satu set *library* tambahan atau *package* tambahan menyediakan kemampuan program tambahan. Pemasukan *package* ini ke dalam perangkat *Java ME* dapat berubah-ubah karena tergantung pada kemampuan sebuah perangkat.

Sebagai contoh, beberapa perangkat *MIDP* tidak memiliki *Bluetooth builtin*, sehingga *Bluetooth API* tidak disediakan dalam perangkat ini. [8]

2.4.3 Configuration

Suatu *configuration* menggambarkan fitur minimal dari lingkungan lengkap *Java runtime*. Untuk menjamin kemampuan portabilitas dan interoperabilitas yang

optimal diantara berbagai macam perangkat yang dibatasi sumber dayanya (*memory*, prosesor, koneksi yang dibatasi), configuration tidak menggambarkan fitur tambahan. Suatu configuration *Java ME* menggambarkan suatu komplemen yang minimum dari teknologi *Java*. Adalah merupakan tugas *profile-profile* untuk menggambarkan tambahan *library* untuk suatu kategori perangkat tertentu. *Configuration* menggambarkan :

a) Subset bahasa pemrograman Java
b) Kemampuan Java Virtual Machine (JVM)
c) Core platform libraries
d) Fitur sekuriti dan jaringan.

J2ME terdiri dari dua configuration yaitu :

1) *Connected Limited Device Configuration* (*CLDC*) Digunakan pada *device* yang terbatas dalam hal memori yang kecil dan prosesor yang tidak begitu cepat. Hanya terdiri dari sebagian kecil *class library J2SE* seperti *java.lang* dan *java.util*, kemudian ditambahkan paket untuk aplikasi *mobile javax.microedition.io.CLDC* banyak digunakan pada *handphone* dan *PDA* (*Personal Digital Asistants*).
2) *Connected Device Configuration* (*CDC*) Berisikan semua dari *java virtual machine* (*JVM*), dan ditambahkan paket untuk aplikasi *mobile* sehingga membutuhkan memori yang lebih besar dan prosesor yang lebih cepat dari *CLDC*.*CDC* banyak digunakan pada *internet tv*, *car tv* dan *nokia communicator*.

2.4.4 Profile

Suatu *profile* menggambarkan set-set tambahan dari *API* dan fitur untuk pasar tertentu, kategori perangkat atau industri. Sementara *configuration* menggambarkan *library* dasar, *profile-profile* menggambarkan *library* yang penting untuk membuat aplikasi-aplikasi efektif. *Library* ini memasukkan *user interface*, jaringan dan penyimpanan *API*.

Profile merupakan *class library* tambahan pada *configuration* yang digunakan untuk pengelompokan spesifikasi tertentu (*domainspesific*), dan

digunakan untuk mendukung paket-paket opsional agar dapat mengakses *virtual machine*. *Profile* merupakan kebalikan dari *configuration* yaitu mengatur hal-hal yang spesifik untuk sebuah *device*. Terdapat beberapa *profile*, namun *profile* pertama yang selesai dirilis adalah *Mobile Information Device Profile* (*MIDP*) yang bertugas untuk menjalankan aplikasi pada *handphone* atau pada *pager*. Satu *device* dapat terdiri dari beberapa *profile* seperti pada *PDA*, terdapat *MIDP*, dan *Personal Basis Profile* (*PBP*) yang digunakan untuk membuat tampilan grafik seperti *window*.

2.4.5 Optional Packages

Optional package adalah kumpulan *API* yang berisikan fungsi-fungsi tertentu yang tidak terdapat pada *configuration* dan *profile*. *Optional package* adalah kumpulan *API* yang berisikan fungsi-fungsi tertentu yang tidak terdapat pada *configuration* dan *profile*.

2.4.6 MIDP

The Mobile Information Device Profile (*MIDP*) berada di atas dari *CLDC*. Kita tidak bisa menulis aplikasi *mobile* hanya dengan menggunakan *CLDC API*. Kita harus tetap memanfaatkan *MIDP* yang mendefinisikan *UI*. Spesifikasi *MIDP*, kebanyakan seperti *CLDC* dan *API* lainnya sudah digambarkan melalui *Java Community Process* (*JCP*). *JCP* melibatkan sebuah kelompok ahli berasal dari lebih dari 50 perusahaan, yang terdiri atas pabrik perangkat *mobile*, pengembang *software*. *MIDP* terus berkembang, dengan versi-versi masa depan yang telah lulus dari proses ketat *JCP*.

Spesifikasi *MIDP* menggambarkan suatu perangkat *MID* yang memiliki karakteristik-karateristik ini sebagai batas minimum :

1) Tampilan
    a) Ukuran Layar: 96x54
    b) kedalaman tampilan: 1-bit
    c) Ketajaman *pixel*: sekitar 1:1
2) Masukkan

a) Satu atau lebih mekanisme *user-input*: satu *keybboard*, dua *keyboard*, atau *touch screen*

3) Memory
   a) *256 kilobytes of non-volatile memory* untuk implementasi *MIDP*.
   b) *8 kilobytes of non-volatile memory for applicationcreated persistent* data
   c) 128 kilobytes of volatile memory for the Java runtime (e.g., the Java heap)
4) Jaringan
   a) dua jalur, wireless, bandwidth terbatas
5) Sound
   a) Kemampuan untuk memainkan nada-nada

2.4.7 MIDlet

Suatu aplikasi *MIDP* disebut *MIDlet*. Perangkat *application management software* (*AMS*) berinteraksi langsung dengan *MIDlet* dengan *method MIDlet create*, *start*, *pause*, dan *destroy*. *MIDlet* adalah bagian dari *package javax.microedition.midlet*. Sebuah *MIDlet* harus di-*extend* dari *class MIDlet*. Dan dapat meminta parameter dari *AMS* seperti dirumuskan dalam *java application descriptor* (*JAD*). Suatu *MIDlet* tidak harus memiliki (dan memang harus tidak mempunyai) sebuah *method public static void main*(*String[] argv*). *Method* tersebut tidak akan dikenal lagi oleh *AMS* sebagai titik awal sebuah program.

2.4.8 Library Pendukung

Aplikasi ini memerlukan beberapa *software* pendukung untuk menambah kebutuhan sistem. *Software* pendukung itu antara lain :

a) AvetanaOBEX

   AvetanaOBEX adalah salah satu paket pendukung atau paket tambahan yang ada pada bahasa pemrograman *java*, yang berfungsi untuk membantu dalam pengiriman *file*. *J2SE* secara *default* tidak mendukung *JSR82*, paket

tambahan untuk *JSR82* khusus untuk *OBEX* pada *J2SE* salah satunya adalah *AvetanaOBEX* yang dapat di-*download* pada *http://avetana.sourceforge.net*

b) Bluecove

Bluecove merupakan paket *bluetooth API* yang digunakan untuk *operating system window*, *bluecove* mengimplementasikan semua *JSR82* tanpa *OBEX*. *Bluecove* dapat di-*download* pada *http://bluecove.sourceforge.net.*

c) COMM API

COMM API juga merupakan paket tambahan pada program *java* yang berfungsi untuk mengakses *serial port* yang ada pada komputer. *J2SE* secara *default* tidak dapat melakukan akses pada *serial port* seperti *USB* atau *COM port*. Sedangkan *device bluetooth* yang digunakan adalah *USB bluetooth* yang terinstal pada *operating system*, sehingga dibutuhkan *COMM API* untuk melakuskan kontrol *hardware*.

2.5 UML (Unified Modeling Language)

UML adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya. Selain itu *UML* adalah bahasa pemodelan yang menggunakan konsep orientasi *object*. *UML* dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson dibawah bendera *Rational Software Corp*. *UML* menyediakan notasinotasi yang membantu memodelkan sistem dari berbagai perspektif. *UML* tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. [9]

2.5.1 Diagrams (Diagram)

Suatu sistem yang kompleks harus dapat dipandang dari sudut yang berbeda-beda sehingga didapat pemahaman secara menyeluruh. Untuk keperluan tersebut UML menyediakan 9 jenis diagrams yang dapat dikelompokkan

berdasarkan sifatnya; Statis atau Dinamis. Kesembilan diagram ini tidak mutlak digunakan atau dibuat sesuai kebutuhan. Pada pemodelan UML dimungkinkan menggunakan diagram lain sejauh diagram tersebut dapat membantu pemahaman mendalam tentang suatu sistem atau perangkat lunak. Berikut adalah diagram yang digunakan :

2.5.1.1 Use Case Diagram

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) system secara keseluran yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada diagram ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use case* menjelaskan secara sederhana fungsi sistem dari sudut pandang user. Adapun komponen-komponen dalam *Use Case Diagram* anataranya [10]:

1. *Actor*

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



Gambar 2. 2 Aktor

2. *Use Case*

Use case merupakan gambaran umum dari fungsi atau proses utama yang menggambarkan tentang salah satu perilaku sistem. Perilaku sistem ini terdefinisi dari proses bisnis sistem yang akan dimodelkan. Tidak semua proses bisnis digambarkan secara fungsional pada *use case*, tetapi yang digambarkan hanya

fungsionalitas utama yang berkaitan dengan sistem. *Use case* menitik beratkan bagaimana suatu sistem dapat berinteraksi baik antar sistem maupun diluar sistem.



Gambar 2. 3 Use Case

3. *System*

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



Gambar 2. 4 System

4. *Association*

Mengidentifikasikan interaksi antara setiap *actor* tertentu dengan setiap *use case* tertentu. Digambarkan sebagai garis antara *actor* terhadap *use case* yang bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian.

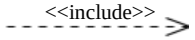
Gambar 2. 5 Association

5. *Dependency*

Dependensi <<include>>

i. Mengidentifikasi hubungan antar dua *use case* dimana yang satu memanggil yang lain.

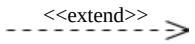
ii. Jika pada beberapa *use case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan *use case* tersendiri dengan relasi dependensi setiap *use case* semula ke *use case* yang baru ini sehingga memudahkan pemeliharaan.
iii. Digambarkan dengan garis putus-putus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



Gambar 2. 6 Dependency <<include>>

Dependensi <<extend>>

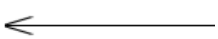
i. Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<extend>>.
ii. Digambarkan serupa dengan dependensi <<include>> kecuali arah panah berlawanan.
iii. Note: konsep "*extend*" ini berbeda dengan "extend" dalam Java.



Gambar 2. 7 Dependency <<extend>>

6. *Generalization*

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

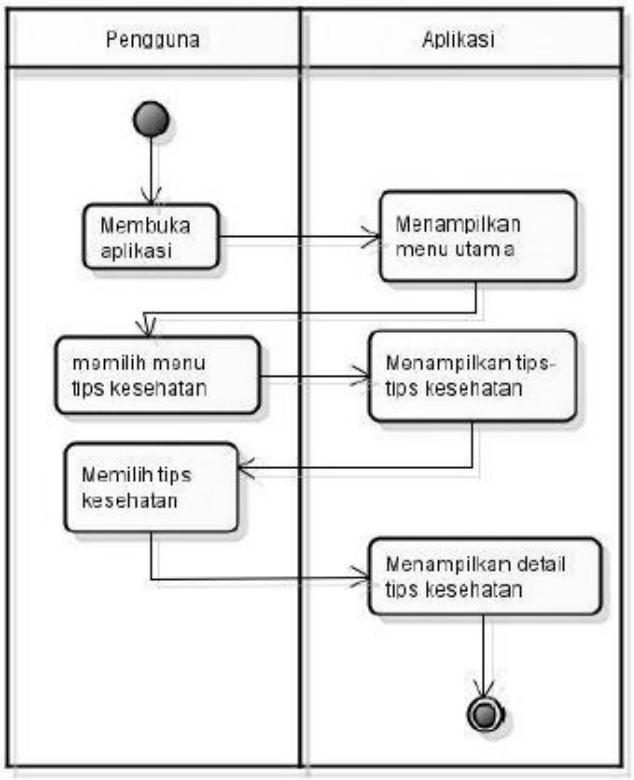


Gambar 2. 8 Generalization

2.5.1.2 Activity Diagram

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

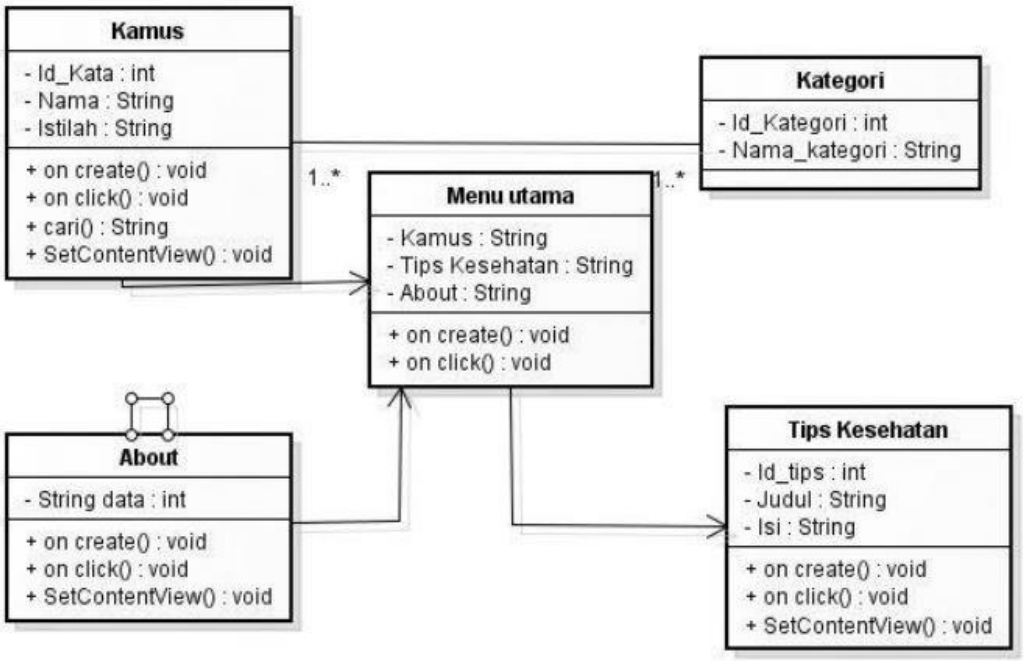
Flowchart atau *data flow diagram* pada perancangan terstruktur. *Activity Diagram* dibuat berdasarkan sebuah atau beberapa *use case* pada *Use Case Diagram* [10].



Gambar 2. 9 Contoh Activity Diagram

2.5.1.3 Class Diagram

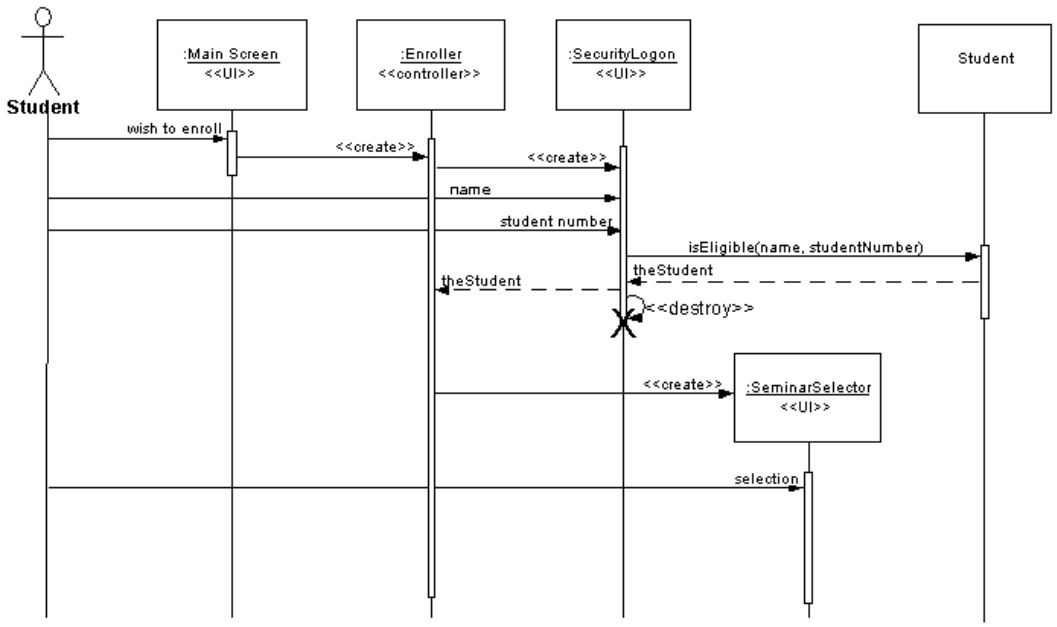
Class diagram merupakan diagram yang selalu ada di pemodelan system berorientasi objek. Class diagram menunjukkan hubungan antar class dalam system yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi property dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti public, private dan protected [10].



Gambar 2. 10 Contoh Class Diagram

2.5.1.4 Sequence Diagram

Sequence Diagram adalah suatu diagram yang digunakan untuk menggambarkan dan menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari use case. Adapun urutan proses yang dijelaskan yaitu interaksi yang terjadi antar class, operasi yang terlibat, urutan antar operasi, dan informs yang diperlukan oleh masing-masing operasi. Komponen utamanya adalah objek yang digambarkan dengan kotak segi empat atau bulat, message yang digambarkan dengan garis penuh, dan waktu yang ditunjukkan dengan progress *vertical* [10].



Gambar 2. 11 Contoh Sequence Diagram

2.5.2 Tujuan UML

Tujuan UML diantaranya yaitu :

1. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan. [9]

2.6 Database (Basis Data)

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur, dan juga batasan-batasan data yang akan disimpan. Basis data merupakan aspek yang sangat penting dalam sistem informasi dimana basis data merupakan gudang penyimpanan data yang

akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghidari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga update yang rumit.

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

Tujuan utama dari DBMS adalah untuk memberikan tinjauan abstrak data kepada user (pengguna). Jadi sistem menyembunyikan informasi tentang bagaimana data disimpan, dipelihara, dan tetap dapat diambil (akses) secara efisien. Pertimbangan efisien disini adalah bagaimana merancang struktur data yang kompleks tetapi masih tetap bisa digunakan oleh pengguna awam tanpa mengetahui kompleksitas strukturnya. Database dibagi menjadi dua jenis yaitu [11]:

1. Basis Data Flat File

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

2. Basis Data Relasional

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

2.7 Perangkat Lunak Pendukung Pembangunan Aplikasi Android

Adapun beberapa *tool* yang dipakai dalam pembangunan perangkat lunak ini adalah sebagai berikut :

2.7.1 Web Service

Web service Sebuah layanan web adalah standar yang digunakan untuk bertukar informasi antara aplikasi atau sistem jenis *heterogen*. Aplikasi perangkat lunak yang ditulis dalam berbagai bahasa pemrograman dan berjalan pada berbagai platform dapat menggunakan layanan *web* untuk bertukar informasi melalui Internet dengan menggunakan protokol http. Interoperabilitas ini dapat dicapai antara aplikasi Java dan Dot net, atau PHP dan aplikasi Java [12].

Beberapa keuntungan menggunakan *web service* pada Android yaitu [12]:

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

3. *Database* bisa di akses melalui aplikasi Android dari jarak jauh.

2.7.2 JSON

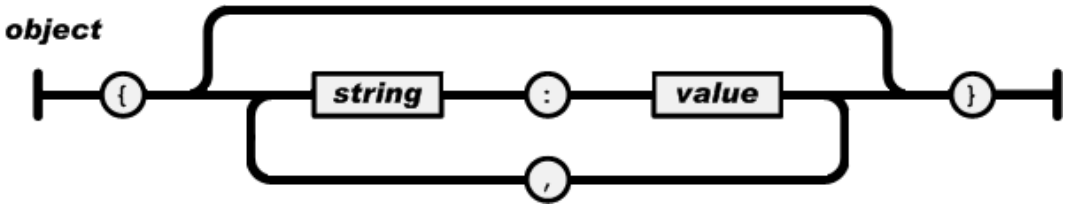
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke 3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur [13]:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), daftar berkunci (*keyed list*), atau *associative* array.
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*).

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

1. Objek

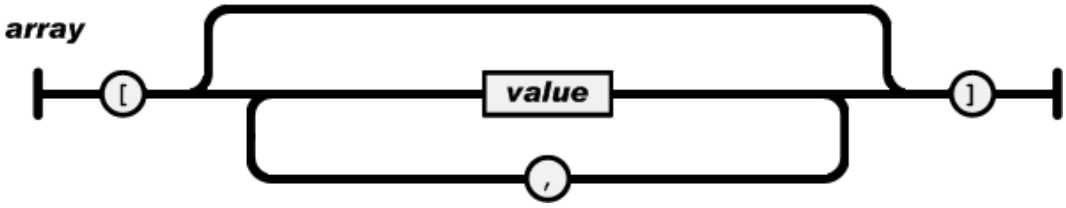
Objek adalah sepasang nama/nilai yang tidak terurutkan. Seperti yang diilustrasikan pada Gambar 2. 2 objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [13].



Gambar 2. 12 Objek JSON

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma) seperti yang diilustrasikan pada Gambar 2. 3. Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat [13].

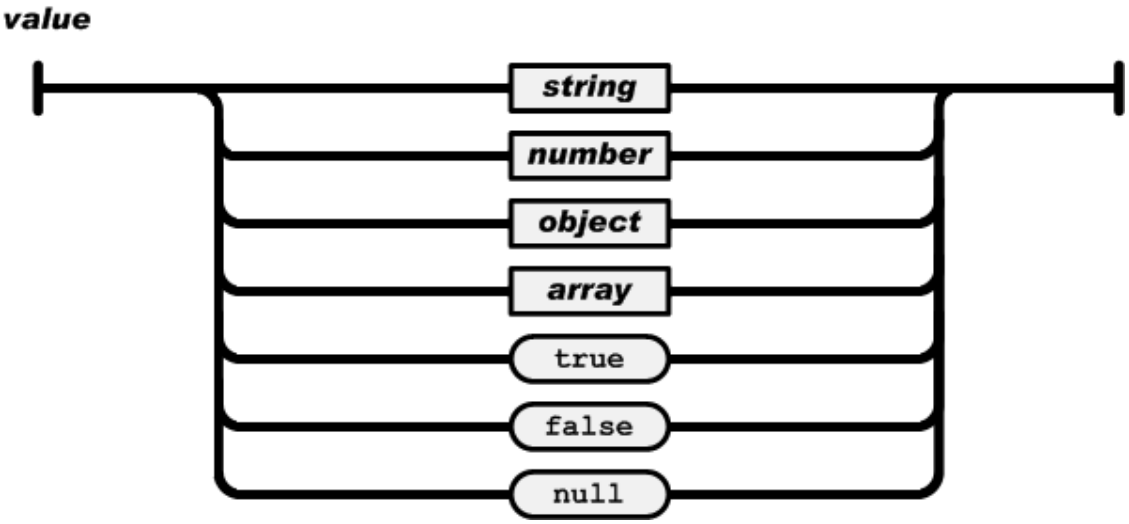


Gambar 2. 13 Array JSON

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

3. Nilai

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat [13].

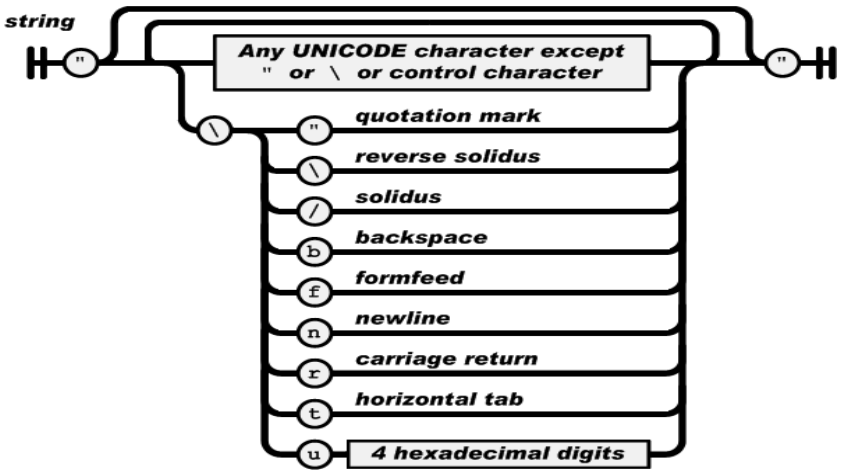


Gambar 2. 14 Value JSON

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash *escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java [13].

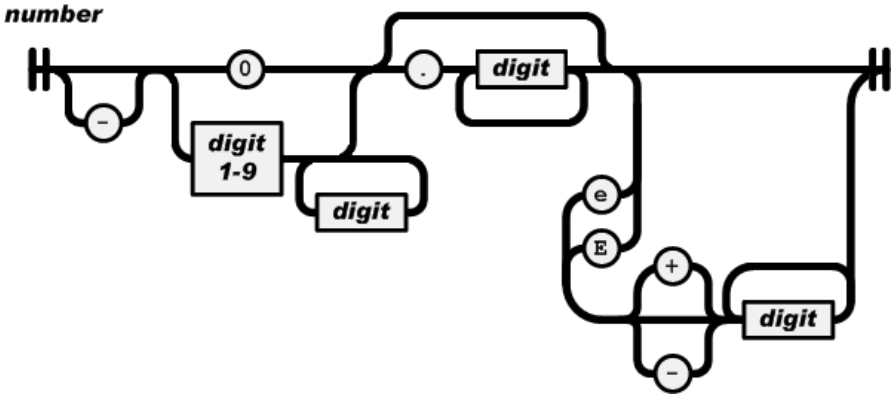
.



Gambar 2. 15 String JSON

5. Angka

   Angka, Format oktal dan heksadesimal tidak digunakan [13].



Gambar 2. 16 Number JSON

2.7.3 API

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

Dengan adanya API, maka memudahkan seorang programmer untuk membongkar suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Suatu rutin standar yang memungkinkan *developer* menggunakan *system function*. Proses ini dikelola melalui *operating system*. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya untuk saling berinteraksi [14].

Keuntungan dengan menggunakan API adalah sebagai berikut :

1. Portabilitas.

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

2. Lebih Mudah Dimengerti

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

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

Perintah open() ini merupakan perintah dari API dan bukan perintah yang langsung dimengerti oleh kernel sistem operasi. Oleh karena itu, agar keinginan pengguna dapat dimengerti oleh sistem operasi, maka perintah open() tadi diterjemahkan ke dalam bentuk *system call* oleh *system call interface*. Implementasi perintah open() tadi bisa bermacam-macam tergantung dari sistem operasi yang digunakan [15].

Cara menggunakan API :

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

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

2.7.4 Object Oriented Analysis Design

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement

(syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem [10].

2.9 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL* AB membuat *MySQL* tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL [16].

Tidak seperti PHP atau Apache yang merupakan software yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu *MySQL* AB. *MySQL* AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan *MySQL* AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Adapun fitur-fitur *MySQL* antara lain [16]:

1. *Relational Database System*. Seperti halnya *software database* lain yang ada di pasaran, MySQL termasuk RDBMS.
2. Arsitektur *Client-Server*. MySQL memiliki arsitektur *client-server* dimana *server database* MySQL terinstal di *server*. *Client* MySQL dapat berada di komputer yang sama dengan *server*, dan dapat juga di komputer lain yang berkomunikasi dengan *server* melalui jaringan bahkan *internet*.
3. Mengenal perintah SQL standar. SQL (*Structured Query Language*) merupakan suatu bahasa standar yang berlaku di hampir semua software *database*. MySQL mendukung SQL versi SQL:2003.
4. Mendukung Sub Select. Mulai versi 4.1 MySQL telah mendukung select dalam select (*sub select*).
5. Mendukung *Views*. MySQL mendukung *views* sejak versi 5.0

6. Mendukung *Stored Prosedured* (SP). MySQL mendukung SP sejak versi 5.0.
7. Mendukung *Triggers*. MySQL mendukung *trigger* pada versi 5.0 namun masih terbatas. Pengembang MySQL berjanji akan meningkatkan kemampuan trigger pada versi 5.1.
8. Mendukung *replication*.
9. Mendukung transaksi.
10. Mendukung *foreign key*.

Perintah-perintah yang ada di SQL antara lain :

1. Menambah data (*Insert*)

   Perintah SQL yang digunakan untuk menambah data ke *database* adalah perintah *INSERT*. Perintah ini mempunyai bentuk umum sebagai berikut:
   INSERT [INTO] nama_tabel [(daftar *field*/kolom)] VALUES (daftar_data).
2. Mengakses data (*Select*)

   Perintah SQL yang digunakan untuk memilih data adalah perintah *SELECT*. Bentuk umum perintah ini adalah sebagai berikut:
   SELECT daftar_select *FROM* daftar_tabel
   [WHERE kondisi_pencarian]
   [GROUP BY daftar_group_by]
   [HAVING kondisi_pencarian]
   [ORDER BY daftar_order[ASC │ DESC]]
3. Mengedit/Mengubah data (*Update*)

   Perintah SQL yang digunakan untuk mengubah data adalah perintah *UPDATE SET*. Perintah ini mempunyai bentuk umum sebagai berikut:
   UPDATE nama_table SET field1=databaru1
   [data2=databaru2]
   [WHERE kondisi_update].
4. Menghapus data (*Delete*)

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

2.10 Internet

Internet (*interconnection networking*) adalah jaringan komunikasi global yang terbuka dan menghubungkan jutaan bahkan milyaran jaringan komputer dengan berbagai tipe dan jenis, dengan menggunakan tipe komunikasi seperti telepon, satelit dan lain sebagainya. Awalnya internet merupakan jaringan komputer yang dibentuk oleh Departemen Pertahanan Amerika Serikat pada tahun 1969 melalui sebuah proyek yang disebut dengan ARPANET. Misi awal dari proyek ini awalnya hanya untuk keperluan militer saja, tetapi lambat laun terus berkembang dan bisa dinikmati oleh semua kalangan. Terciptanya internet telah membawa perubahan yang sangat berarti dalam berbagai aspek kehidupan manusia. Selain itu, internet juga telah melahirkan dunia baru yang memiliki pola, corak sekaligus karakteristik yang berbeda dengan dunia nyata [17].

Internet yang sering disebut dengan jagad raya informasi menyajikan sekian banyak fasilitas yang dapat dinikmati oleh pengguna internet. Berikut ini beberapa fasilitas yang dapat dinikmati oleh pengguna internet yaitu www (*world wide web*), email, *mailing list*, BBS, *chatting*, *newsgroup*, FTP dan *gopher* [17].

2.11 Bootstrap

Bootstrap merupakan sebuah *framework* CSS dari Twitter, yang menyediakan kumpulan komponen-komponen antarmuka dasar pada web yang telah dirancang sedemikian rupa untuk digunakan bersama-sama. Selain komponen antarmuka, Bootstrap juga menyediakan sarana untuk membangun *layout* halaman dengan mudah dan rapi, serta modifikasi pada tampilan dasar HTML untuk membuat seluruh halaman web yang dikembangkan senada dengan komponen-komponen lainnya [18].



Gambar 2. 17 Bootstrap

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

2.12 Metode Pengujian

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

Perangkat lunak dapat diuji dengan dua cara, yaitu:

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

2.12.1 Black-Box Testing

Pengujian *Black Box* merupakan pendekatan komplementer dari teknik white box, karena pengujian *Black Box* diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik *white box*. Pengujian *Black Box* berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk

mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian *Black Box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *Black Box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian *Black Box* berusaha menemukan kesalahan dalam kategori :

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

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

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

2.13 Optical Character Recognition (OCR)

Optical Character Recognition merupakan konversi elektronik ataupun mekanik dari gambar yang berupa tulisan tangan maupun teks menjadi teks digital sehingga dapat digunakan secara elektronik untuk berbagai keperluan. *Optical character recognition* ini merupakan bagian dari konsep *pattern recognition* [19].

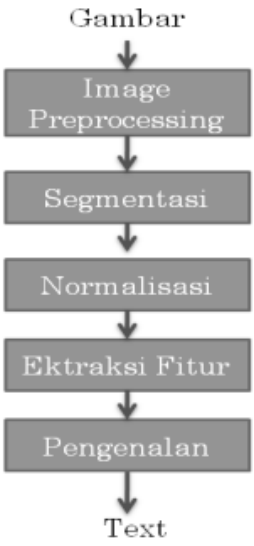
Optical Character Recognition (*OCR*) merupakan aplikasi dari teknologi pengenalan teks, yaitu suatu teknologi yang mampu mengenali teks pada citra digital dan mengalihkannya pada dokumen digital. Dalam perkembangannya, aplikasi *OCR* sering kali digunakan pada berbagai jenis dokumen, dimana beberapa dokumen memiliki ukuran *font* yang berbeda satu sama lain. Hal ini menyebabkan aplikasi *OCR* yang ada menjadi kurang maksimal dalam mengenali teks.

OCR adalah proses yang menerjemahkan gambar karakter (image character) menjadi bentuk teks dengan cara mencocokkan pola karakter per-baris dengan pola yang telah tersimpan dalam database aplikasi. Hasil dari proses OCR adalah berupa teks sesuai yang tampak pada gambar output scanner dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar [20].

Tingkat keberhasilan dari perangkat lunak aplikasi pengenalan teks sangat bergantung dari sejumlah faktor berikut :

1. Kualitas gambar teks yang ada pada dokumen yang dibaca serta tingkat kompleksitasnya (ukuran, format teks, warna, latar belakang).
2. Kualitas alat optik yang dipakai (*scanner*).
3. Kualitas perangkat lunak aplikasi pengenalan teks itu sendiri. [21]

Pada proses pengolahan gambar dengan *OCR* terdapat beberapa langkah yaitu, *segmentation*, *normalization*, *feature extraction*, *recognition*, dan hasil. Langkah *segmentation* bertujuan untuk memisahkan objek pada gambar dengan latar belakang agar mudah untuk dianalisis. Langkah *normalization* bertujuan untuk mengubah ukuran gambar dan melakukan proses *morfologi* untuk menghapus piksel *forground* yang terpilih dari gambar *biner*. Langkah *feature extraction* berfungsi sebagai *image analysist* untuk mengidentifikasi sifat yang melekat pada tiap karakter. Hasil dari langkah *feature extraction* akan digunakan dalam langkah *rocognition* sehingga menghasilkan *output* [22].



Gambar 2. 18 Proses OCR secara umum

Dalam sistem *OCR*, *input* yang dimasukkan adalah gambar yang berisi karakter yang ingin dikenali. Sebelum karakter pada gambar dapat dikenali, gambar tersebut harus terlebih dahulu dilakukan *preprocessing*. Tujuan dari proses ini ialah menghilangkan bagian-bagian dari gambar yang tidak diinginkan (bukan bagian dari karakter yang ingin dikenali) dan untuk memperbaiki kualitas gambar sehingga objek pada gambar lebih mudah untuk dikenali. *Preprocessing* pada gambar biasanya meliputi *grayscaling*, *noise removal*, dan *thresholding*. *Grayscaling* merupakan tahap awal dalam *image preprocessing*, yaitu mengubah gambar berwarna menjadi gambar yang hanya memiliki derajat keabuan saja. Selanjutnya dilakukan proses *noise filtering*, yaitu proses meredukti atau mengurangi *noise*. Proses akhir dari *image preprocessing* adalah *thresholding*, yaitu suatu proses untuk memisahkan *background* dengan objek yang ingin diamati dengan mengubah gambar menjadi hitam putih.

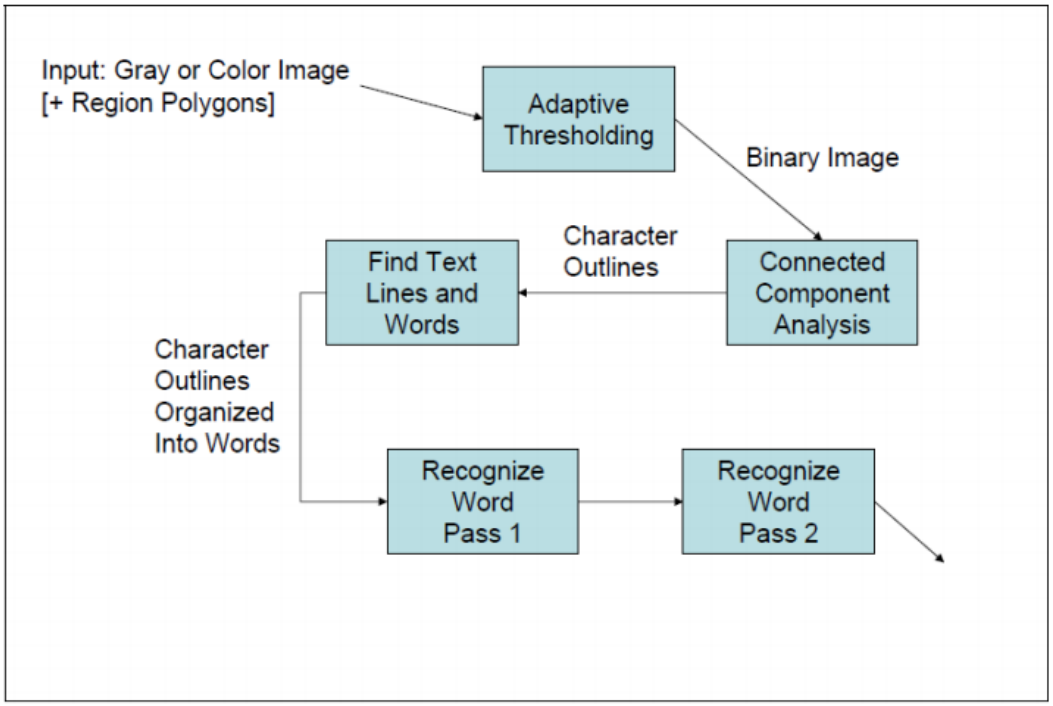
Setelah *preprocessing* selesai dilakukan, maka tahap selanjutnya ialah segmentasi. Proses ini digunakan untuk memisahkan area-area pengamatan dari setiap karakter yang ingin dikenali, seperti pemisahan kalimat ke dalam kata-kata dan pemisahan kata ke dalam karakter-karakter. Tahap selanjutnya ialah normalisasi dilakukan untuk proses normalisasi pada *OCR* adalah *scaling* dan *thinning*. Setelah normalisasi dilakukan, kemudian akan dilakukan ekstraksi *fitur* untuk mendapatkan karakteristik dari masing-masing karakter yang

membedakannya dengan karakter lain. Tahap akhir dari proses *OCR* adalah pengenalan atau *recognising*, dalam tahap ini algoritma akan membandingkan ciri-ciri fitur yang ingin dikenali dengan data yang tersimpan sebelumnya. Hasil pengenalan berupa teks dengan kemiripan paling besar antara fitur karakter yang ingin dikenali dengan informasi yang tersimpan.

2.13.1 Tesseract

Tesseract adalah mesin *OCR open-source* yang dikembangkan oleh *HP* pada tahun 1984 – 1994. Pada saat ini, untuk pertama kalinya, detail arsitektur dan algoritma dipublikasikan. *Tesseract* dimulai sebagai sebuah proyek penelitian seorang *PhD* di *HP Labs, Bristol*, dan mendapatkan momentum sebagai perangkat lunak dan *add-on* perangkat keras untuk lini *HP scanner flatbed*. Proyek ini termotivasi oleh fakta bahwa mesin *OCR* komersial pada saat itu berada di tahap awal, gagal total dan hanya mementingkan kualitas cetak terbaik.

Setelah proyek bersama antara *HP Labs Bristol*, dan *Divisi scanner HP* di Colorado, *Tesseract* memimpin secara signifikan dalam akurasi yang lebih baik dibandingkan dengan mesin *OCR* komersial, tetapi *Tesseract* tetap tidak menjadi suatu produk. Tahap selanjutnya dari perkembangannya adalah kembali ke *HP Labs Bristol* untuk investigasi *OCR* sebagai alat kompresi. Pengembangan lebih berkonsentrasi pada peningkatan *rejection efficiency* dari pada tingkat akurasi. Pada akhir proyek ini, pada akhir tahun 1994, pembangunan berhenti sepenuhnya. Mesin *OCR* dikirim ke *UNLV* untuk melakukan tes akurasi pada tahun 1995, di mana mesin tersebut membuktikan bahwa kualitas *Tesseract* lebih baik dibandingkan mesin *OCR* komersial. Pada akhir tahun 2005, *HP* merilis *open source Tesseract* [23].



Gambar 2. 19 Arsitektur Tesseract

1. Adaptive Thresholding

   Proses ini digunakan untuk merubah citra warna atau greyscale yang menjadi citra masukan menjadi citra biner. Tujuannya adalah untuk memisahkan background dan foreground.

2. Connected Component Analysis

   Proses ini digunakan untuk melakukan proses segmentasi teks yang ada pada citra menjadi karakter yang nantinya akan dikenali.

3. Find Lines and Words

   Proses ini digunakan untuk mencari baris teks pada suatu citra.

4. Recognize Word Pass 1

   Proses ini digunakan untuk proses pengenalan tahap awal, yaitu membandingkan karakter yang didapat dari citra dengan data yang sudah ada.

5. Recognize Word Pass 2

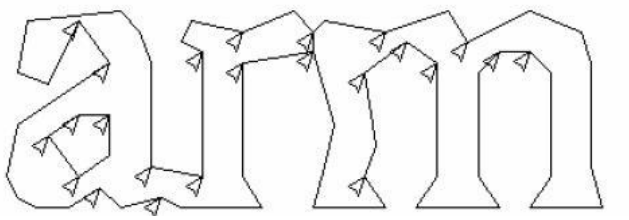
   Proses terakhir adalah proses pengenalan kembali dari keseluruhan halaman pada citra untuk memperbaiki hasil pengenalan tahap pertama.

2.13.1.1 Pengenalan Karakter Oleh Tesseract

Bagian dari proses pengenalan untuk mesin pengenalan karakter apapun ialah untuk mengidentifikasi bagaimana sebuah kata atau karakter sebaiknya disegmentasi menjadi karakter atau simbol-simbol. *Output* segmentasi awal dari pencarian baris diklasifikasikan. Sisa langkah pengenalan kata hanya dilakukan pada teks yang *non-fixed-pitch*.

1. Pemisahan Karakter Terhubung

Apabila hasil dari pengenalan kata tidak memuaskan, *Tesseract* berusaha untuk memperbaiki hasil dengan memisahkan *blob* dengan keyakinan terburuk dari *classifier* karakter [23]. Kandidat untuk titik-titik pemisahan ditemukan dari simpul cekung dari pendekatan poligonal *outline* dan mungkin saja terdapat titik cekung berlawanan lainnya atau segmen garis. Ini akan menghabiskan sampai 3 pasang titik pemotongan untuk memisahkan karakter yang terhubung dari *set* ASCII.



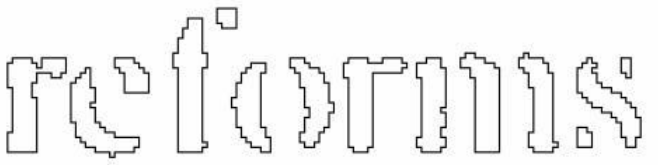
Gambar 2. 20 Kandidat Titik Pemotongan

Gambar 2. 20 menunjukkan kumpulan kandidat titik pemisahan dengan panah dan pemotongan terpilih sebagai sebuah garis melewati *outline* di mana huruf ‘r’ menyentuh huruf ‘m’. Pemotongan dilakukan sesuai dengan urutan prioritas. Setiap pemotongan yang gagal memperbaiki hasil keyakinan dibatalkan, namun tidak sepenuhnya dibuang sehingga pemotongan dapat digunakan kembali oleh *asosiator* jika dibutuhkan.

2. Asosiasi Karakter Patah

Pada tahap pemisahan karakter terhubung tidak menghasilkan karakter atau kata yang cukup bagus, kemudian dilanjutkan ke *asosiator*. *Asosiator* ini membuat pencarian A^* (*best first search*) dari graf segmentasi untuk kemungkinan kombinasi dari maksimal pemisahan *blob* menjadi karakter kandidat. Hal itu dilakukan tanpa secara nyata membangun graf segmentasi, melainkan membangun

tabel *hash* dari *state* yang dikunjungi. Hasil pencarian A^* dengan menarik kandidat *state* baru dari antrian prioritas dan mengevaluasi kandidat dengan mengklasifikasikan kombinasi fragmen yang belum terklasifikasi.



Gambar 2. 21 Karakter Terputus yang dapat dikenali dengan baik

Segmentasi pencarian A^* pertama kali diimplementasikan sekitar pada tahun 1989. Dengan algoritma pencarian A^* ini, *Tesseract* dengan mudah mengenali karakter patah seperti pada Gambar 2. 21.

3. Klasifikasi Bentuk

a. *Static Classifier*

Versi awal Tesseract menggunakan *feature* topologikal yang dikembangkan dari karya Shillman. Kemudian ide selanjutnya melibatkan penggunaan segmen dari pendekatan poligonal sebagai *feature*, namun pendekatan ini tidak terlalu kuat untuk karakter yang rusak. Solusi terobosan yang digunakan adalah *feature* yang tidak ketahui tidak perlu sama dengan *feature* yang terdapat data *training.*

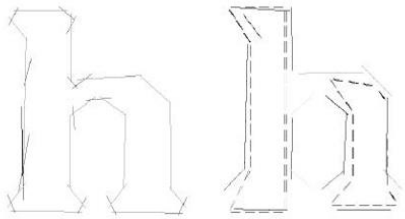
Selama pelatihan, segmen dari pendekatan poligonal digunakan untuk *feature,* namun pada proses pengenalan, *feature* kecil yang panjangnya tetap (dalam unit ternormalisasi) diekstraksi dari *outline* dan dicocokan secara *many-to-one* terhadap prototipe *feature* yang ter-*cluster* pada data pelatihan.

b. *Adaptive Classifier*

Tesseract tidak menggunakan *template classifier,* tetapi menggunakan *feature* yang sama seperti *static classifier.* Perbedaan yang signifikan antara *static classifier* dan *adaptiver classifier,* terlepas dari data pelatihan, *adaptive classifier* menggunakan normalisasi *isotropic baseline/x-height*, sedangkan *static classifier* menormalisasi karakter oleh *centroid* (momen pertama) untuk posisi dan momen kedua untuk normalisasi ukuran yang *anisotropic.*

Classifier bentuk *Tesseract* bekerja sangat baik pada 5000 karakter *Chinese* tanpa memerlukan modifikasi besar, sehingga cocok untuk ukuran kelas besar.

Feature merupakan komponen pendekatan poligonal dari *outline* sebuah bentuk. Pada *training,* vektor fitur 4 dimensi (x,posisi-y, arah, panjang) diturunkan dari setiap elemen pendekatan poligonal dan dikelompokkan untuk membentuk prototipikal vektor fitur. Pada pengenalan, elemen-elemen poligon dipecah menjadi bagian-bagian yang lebih pendek dengan panjang yang sama, sehingga dimensi panjang dieliminasi dari vektor fitur. Beberapa fitur pendek dicocokkan dengan setiap fitur prototipikal dari *training,* hal ini membuat proses klasifikasi lebih kuat terhadap karakter yang terputus.



(a) (b)

Gambar 2. 22 (a) Prototipe huruf h untuk font Times New Roman
(b) Pencocokan huruf h yang terputus terhadap prototipe

Gambar 2. 22 (a) menujukkan sebuah contoh prototipe dari huruf ‘h’ untuk huruf *Time New Roman*. Segmen garis warna hijau merepresentasikan *cluster* dari *cluster* signifikan yang berisi sampel hampir dari setiap sampel huruf ‘h’ pada *Times New Roman*. Segmen biru merupakan *cluster* yang digabung dengan *cluster* lainnya untuk membentuk *cluster* signifikan. Segmen magenta tidak digunakan karena cocok dengan *cluster* signifikan yang ada. Segmen merah tidak mengandung cukup sampel untuk signifikan dan tidak dapat digabung dengan *cluster* tetangga manapun untuk membentuk *cluster* signifikan.

Gambar 2. 22 (b) menunjukkan bagaimana fitur lebih pendek yang tidak diketahui dicocokkan terhadap prototipe untuk mencapai intensitas dari karakter yang rusak. Garis pendek tebal merupakn fitur yang tidak diketahui, menjadi ‘h’ yang rusak dan garis yang lebih panjang merupakan fitur prototipe. Warna merepresentasikan kualitas pencocokan : hitam -> bagus, *magenta* -> lumayan, *cyan* -> buruk, dan kuning -> tidak cocok. Prototipe vertikal semua memiliki kecocokan yang baik, meskipun fakta bahwa ‘h’ rusak.

4. *Contextual Post-Preprocessing*

Proces *training Tesseract* mendukung sebagian perluasan model bahasa dengan menyediakan suatu cara menghasilkan kamus untuk bahasa baru dari daftar kata. Untuk kekompakan dan pencarian yang cepat, kamus ini diwakili oleh *directed acyclic word graphs* (DAWGs). Pada implementasi, data struktur DAWG digunakan secara sekuensial mencari beberapa kamus termasuk sistem kamus *pre-generated*, kamus dokumen (secara dinamis dikontruksi dari kata-kata pada dokumen yang dilakukan OCR) dan daftar kata yang disediakan oleh *user* sendiri.

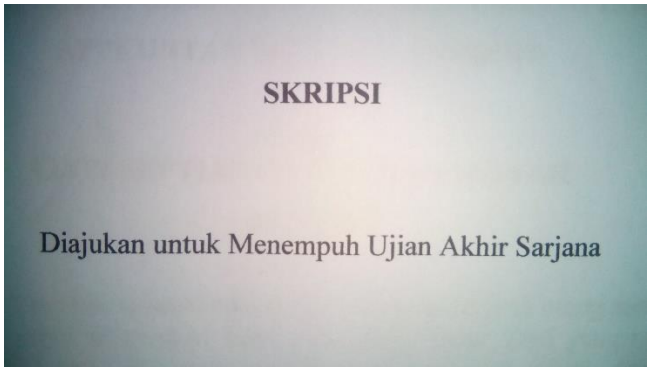
Pada mulanya setiap *edge* pada DAWG menyimpan karakter 8-bit untuk mewakili huruf atau karakter yang digunakan untuk transisi huruf yang sesuai dalam DAWG. Struktur data DAWG kemudian dimodifikasi untuk menyimpan ID *unicharset* yang digunakan oleh *classifier* karakter sehingga dapat digunakan oleh bahasa dengan karakter *multi-byte* termasuk bahasa Korea sendiri. Hal ini secara signifikan menyederhanakan proses konstruksi dan pencarian DAWG, selain itu pencarian dapat dilakukan secara paralel untuk keseluruhan DAWG yang ada. Untuk menemukan *string* yang diberikan merupakan kata yang *valid* dalam kamus, pencarian dimulai dengan inisial *set* dari DAWG yang "aktif". Karena setiap huruf dalam kata dianggap, set ini direduksi menjadi hanya berisi sebagian string yang "diterima". Pada akhir proses, *set* dari "aktif" DAWG berisi DAWG yang berisi kata.

2.13.1.2 Teori Memecah Karakter Gambar Menggunakan Tesseract

Teori memecahkan karakter gambar menggunakan *tesseract* pada *smartphone android*. Dalam memecahkan karakter gambar adalah dengan membuat algoritma yang baik sehingga *smartphone android* dapat membaca atau mengenali setiap karakter yang ada pada gambar seperti manusia yang melakukannya.

Karakter gambar perlu dipilih terlebih dahulu dengan melihat apakah kertasnya dalam keadaan baik, warna kertasnya putih dengan warna tinta yang hitam dan jelas, tidak ada huruf yang saling menimpa, kertas tidak terlipat-lipat atau

rusak. Dengan demikian, *tesseract* dapat mengenali teks pada gambar tersebut dengan baik sehingga mendapat hasil yang lebih baik pula.



Gambar 2. 23 Contoh Karakter Gambar

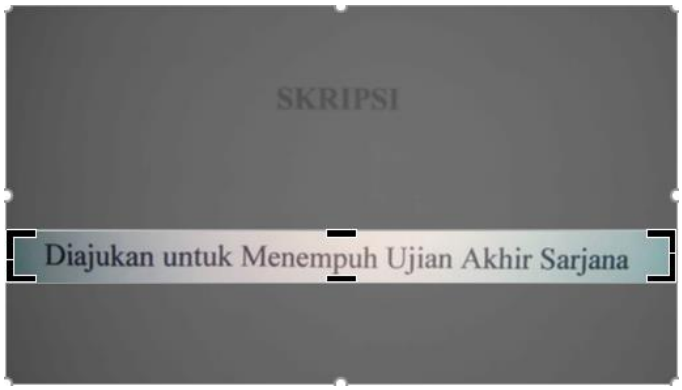
Berikut adalah langkah-langkah memecahkan data struk menggunakan *tesseract* pada *smartphone android* :

1. Ambil Gambar

Ambil gambar dengan menggunakan kamera *smartphone android* atau dengan cara mengambil gambar dari galeri.

2. Pemotongan Gambar (*Cropping*)

Setelah gambar diambil atau dipilih maka masuk ke proses pemotongan gambar. Pemotongan gambar dapat dilakukan dengan cara menggeserkan setiap titik potong, sehingga didapat data yang telah diinginkan. Lihat contoh gambar dibawah ini yaitu Gambar 2. 24 Contoh Pemotongan Karakter Gambar.



Gambar 2. 24 Contoh Pemotongan Karakter Gambar

3. Proses Pengenalan Karakter

Setelah hasil pemotongan disimpan maka sistem *smartphone android* akan melakukan proses pengenalan karakter menggunakan *tesseract*. Dimana setelah karakter gambar berhasil dibaca atau dikenali maka sistem akan menampilkan hasil data karakter gambar. *Coding tesseract* untuk proses pengenalan karakter pada gambar hingga menampilkan hasil data yang telah dibaca atau dikenali, yaitu :

```
private void prepareTesseract() {
        try {
            prepareDirectory(DATA_PATH + TESSDATA);
        } catch (Exception e) {
            e.printStackTrace();
        }
        copyTessDataFiles(TESSDATA);
    }

private void copyTessDataFiles(String path) {
        try {
            String fileList[] = getApplicationContext().getAssets().list(path);

            for (String fileName : fileList) {

                // open file within the assets folder
                // if it is not already there copy it to the sdcard
                String pathToDataFile = DATA_PATH + path + "/" + fileName;
                if (!(new File(pathToDataFile)).exists()) {

                    InputStream in = getApplicationContext().getAssets().open
                    (path + "/" + fileName);

                    OutputStream out = new FileOutputStream(pathToDataFile);
```

```
            // Transfer bytes from in to out
            byte[] buf = new byte[1024];
            int len;

            while ((len = in.read(buf)) > 0) {
                out.write(buf, 0, len);
            }
            in.close();
            out.close();

            Log.d(TAG, "Copied " + fileName + "to tessdata");
          }
        }
      } catch (IOException e) {
        Log.e(TAG, "Unable to copy files to tessdata " + e.toString());
      }
    }

private void startOCR() {
      try {
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inSampleSize = 4; // 1 - means max size. 4 - means maxsize/4
        size. Don't use value <4, because you need more memory in the heap
        to store your data.
        //Bitmap bitmap = BitmapFactory.decodeFile(imgUri.getPath(),
        options);

        result = extractText(thePic);
        textView.setText(result);
```

```
            parsingText();

        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
        }
    }

private String extractText(Bitmap bitmap) {
        try {
            tessBaseApi = new TessBaseAPI();
        } catch (Exception e) {
            Log.e(TAG, e.getMessage());
            if (tessBaseApi == null) {
                Log.e(TAG, "TessBaseAPI is null. TessFactory not returning tess
                 object.");
            }
        }

        tessBaseApi.init(DATA_PATH, lang);

//       //EXTRA SETTINGS
//        //For example if we only want to detect numbers
//       tessBaseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST,
         "1234567890");
//        //blackList Example
        //tessBaseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST,
        "!@#$%^&*()_        +=-,./<>?");

        Log.d(TAG, "Training file loaded");
        tessBaseApi.setImage(bitmap);
```

```
    String extractedText = "empty result";
    try {
        extractedText = tessBaseApi.getUTF8Text();
    } catch (Exception e) {
        Log.e(TAG, "Error in recognizing text.");
    }
    tessBaseApi.end();
    return extractedText;
}
```