BAB II
KAJIAN PUSTAKA DAN LANDASAN TEORI

2.1 Kajian Pustaka

Adhika Novandya dkk. (2012) dalam penelitiannya yang berjudul "Aplikasi Pengenalan Budaya Dari 33 Provinsi Di Indonesia Berbasis *Android*" mengatakan bahwa kebudayaan yang ada di Indonesia pada saat ini secara perlahan mulai terlupakan. Oleh karena itu diperlukan sebuah aplikasi yang menampilkan informasi mengenai kebudayaan yang ada pada setiap provinsi di Indonesia dengan tujuan membantu masyarakat dalam mendapatkan pengetahuan mengenai berbagai macam keanekaragaman budaya yang ada di Indonesia (Novandya, 2012).

Arzan Muharom dkk (2013) dalam penelitiannya yang berjudul "Pengembangan Aplikasi Sunda Berbasis Android Menggunakan *Metode Rapid Application Development* (RAD)" mengatakan bahwa budaya Sunda pada saat ini semakin mulai ditinggalkan oleh masyarakat. Sehingga diperlukan sebuah aplikasi untuk memperkenalkan budaya sunda yang memiliki fitur belajar bahasa sunda juga dapat mengetahui serta memahami budaya lokal (Muharom, 2013).

I Komang Sumita dkk (2013) dalam penelitiannya yang berjudul "Aplikasi Pembelajaran Kebudayaan dan Ilmu Agama Hindu di Indonesia Berbasis *Android*" mengatakan bahwa kurangnya informasi dan pembelajaran tentang kebudayaan dan ilmu agama Hindu seperti sejarah agama Hindu, pokok-pokok ajaran agama Hindu dan doa-doa sehingga tidak banyak masyarakat umum yang mengetahui perkembangan kebudayaan dan ilmu agama Hindu di Indonesia. Sehingga diperlukan aplikasi alternatif pembelajaran kebudayaan Hindu pada *smartphone* agar mempermudah dalam pembelajaran dan pemberitaan informasi tentang kebudayaan dan ilmu agama Hindu (Sumita, 2013).

Anjas Ardianto dkk (2016) dalam penelitiannya yang berjudul "Rancang Bangun Aplikasi Pariwisata Malang Berbasis *Android*" mengatakan bahwa Banyak

masyarakat yang masih belum tahu tentang tempat wisata yang ada di kota Malang. Rancang Bangun Aplikasi Pariwisata Malang Berbasis *Android* merupakan salah satu solusi untuk mengatasi permasalahan tentang minimnya informasi pariwisata di Kota Malang. Rancang Bangun Aplikasi Pariwisata Malang Berbasis *Android* adalah aplikasi yang berisi informasi obyek wisata di Malang yang dilengkapi gambar (Ardianto, 2016).

Dari beberapa penelitian yang telah dilakukan, Bahasa pemrograman PHP dan database MySQL banyak digunakan dalam membangun aplikasi tersebut. Dengan begitu pada penelitian ini akan dibuat aplikasi kebudayaan di DIY berbasis *android*.

Adapun perbedaan pada penelitian ini yaitu pada penelitian sebelumnya aplikasi hanya dapat menampilkan informasi-informasi sedangkan pada penelitian ini ditambahkan fitur map atau penunjuk arah untuk mengunjungi tempat event itu berlangsung.

2.2 Landasan Teori

2.2.1 Aplikasi *Mobile*

Aplikasi *mobile* yaitu istilah yang digunakan untuk medeskripsikan aplikasi internet yang berjalan pada *smartphone* atau piranti *mobile* lainnya. Aplikasi *mobile* biasanya membantu para penggunanya untuk terkoneksi dengan layanan internet yang biasa diakses pada *PC* atau mempermudah mereka untuk menggunakan aplikasi internet pada piranti yang bisa dibawa (Turban, 2012, p. 277).

2.2.2 Pengertian *Android*

Android adalah sebuah sistem operasi *open source* yang mempunyai banyak API *Library* untuk membuat aplikasi berbasis *mobile* dengan tampilan yang menarik dan berfungsi dengan bagus ((Meier, 2009, p. 1). Sedangkan menurut J.F. DiMarzio, *android* merupakan suatu sistem operasi yang menggunakan *java*

sebagai bahasa pemrograman dan berbasis *linux* serta dirancang khusus untuk telepon seluler layar sentuh seperti *smartphone* dan tablet (DiMarzio, 2008)

Google mengakuisisi perusahaan Android Inc. pada tanggal 17 Agustus 2005 dan menjadikannya sebagai anak perusahaan yang dimiliki oleh Google. Pendiri Android Inc. yaitu Rubin, Miner, serta White tetap bekerja pada perusahaan tersebut setelah diakuisisi oleh Google. Di Google, tim yang dipimpin oleh Andy Rubin mulai untuk mengembangkan sebuah platform perangkat seluler dengan menggunakan *kernel Linux*.

Sejak tahun 2008, Android mulai secara bertahap melakukan sejumlah pembaruan atau *update* untuk meningkatkan kinerja dari sistem operasi tersebut dengan menambahkan fitur baru, memperbaiki *bug* pada versi android yang sebelumnya. Setiap versi yang dirilis dinamakan secara alfabetis dengan berdasarkan nama sebuah makanan pencuci mulut, seperti cupcake, donut, dan sebagainya.

2.2.3 Arsitektur *Android*

Menurut Nazaruddin Safaat (Safaat, 2011) android memiliki komponen utama sebagai berikut:

a. Aplikasi dan *Widgets*

Aplikasi dan *Widgets* ini adalah layer di mana kita berhubungan dengan aplikasi saja, di mana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di Layer terdapat aplikasi inti termasuk *email* klien, program SMS, kalender, peta, *browser*, kontak, dan lain-lain. Semua aplikasi ditulis menggunakan bahasa pemrograman Java.

b. Aplikasi *Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*,

mengatur alarm, dan menambahkan status *notifications*, dan sebagainya. Pengembang memiliki akses penuh menuju *API framework* seperti yang dilakukan oleh aplikasi yang berkategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan.

Sehingga bisa kita simpulkan Aplikasi *Framework* ini adalah layer di mana para pembuat aplikasi melakukan pengembangan atau pembuatan aplikasi yang akan dijalankan di sistem operasi android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti *content-providers* yang berupa sms dan panggilan telepon.

c. *Libraries*

Libraries ini adalah layer di mana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasi. Berjalan di atas *kernel*, Layer ini meliputi berbagai library C/C++ inti seperti Libc dan SSL, serta:

1. *Libraries* media untuk pemutaran media audio dan video.
2. *Libraries* untuk manajemen tampilan.
3. *Libraries Graphics* mencakup *SGL* dan *OpenGL* untuk grafis 2D dan 3D.
4. *Libraries SWLite* untuk dukungan database.
5. *Libraries SSL* dan *WebKit* terintegrasi dengan web *browser* dan *security*.
6. *Libraries liveWebcore* mencakup modern web *browser* dengan *engine embeded* web *view*.
7. *Libraires* 3D yang mencakup implementasi *OpenGL* S 1.0 API's.

d. Android Runtime

Layer yang membuat aplikasi Android dapat dijalankan di mana dalam prosesnya menggunakan implementasi *Linux*. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android Run Time Time dibagi menjadi dua bagian yaitu:

1. *Core libraries*: Aplikasi Android dibangun dalam bahasa java, sementara *Dalvik* sebagai virtual mesinnya bukan *Virtual Machine Java*, sehingga

diperlukan sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa java yang ditangani oleh *Core Libraries.*

2. *Dalvik Virtual Machine*: Virtual mesin berbasis *register* yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, di mana merupakan pengembangan yang mampu membuat *linux kernel* untuk melakukan *threading* dan manajemen tingkat rendah.

e. Linux Kernel

Linux Kernel adalah layer di mana inti dari sistem operasi Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory, resource, driver*, dan sistem-sistem operasi android lainnya. *Linux kernel* yang digunakan android adalah *linux kernel relase* 2.6.

2.2.4 Java

Java merupakan bahasa berorientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat jaringan komunikasi atau jaringan internet. Melalui teknologi java, dimungkinkan perangkat audio stereo dirumah terhubung jaringan komputer. Java tidak lagi hanya untuk membuat *applet* yang memperintah halaman web tapi java telah menjadi bahasa untuk pengembangan aplikasi skala *interprise* berbasis jaringan besar (Haryanto, 2011, p. 2).

2.2.5 JSON

Menurut Deitel (M.Deitel, 2013) JSON (*JavaScript Object Notation*) adalah suatu format pertukaran data komputer. Format dari JSON adalah berbasis teks, dapat terbaca oleh manusia, digunakan untuk mempresentasikan struktur data sederhana, dan tidak bergantung dengan bahasa apapun. Biasanya digunakan pada aplikasi Ajax. Format JSON sering digunakan untuk mentransmisikan data terstruktur melalui koneksi jaringan. Secara umum, JSON digunakan untuk mentransmisikan data antara *server* dan aplikasi *web*.

Jenis media internet yang resmi untuk JSON adalah aplikasi/json. Format JSON sering digunakan untuk serialisasi dan mengirimkan data terstruktur melalui koneksi jaringan, terutama untuk pengiriman data antara server dan aplikasi *web* melayani sebagai alternatif ke XML.

2.2.6 *Android* Studio

Android Studio adalah sebuah *software* resmi yang dikembangkan Google untuk membantu para *developer* dalam mengembangkan aplikasi (Kevin Grant, 2014, p. 16). Dengan menggunakan *Android Studio* pengembangan aplikasi menjadi lebih singkat dan jauh lebih mudah dibandingkan sebelumnya (Adam Gerber, 2014, p. 25). *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas anda saat membuat aplikasi *Android*, misalnya:

1. Sistem versi berbasis Gradle yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat pengujian dan kerangka kerja yang ekstensif
7. Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain

Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*

2.2.7 *Android Virtual Device*

AVD adalah sebuah *emulator* yang mengizinkan aplikasi android untuk diuji di komputer atau laptop sendiri tanpa harus menginstal aplikasi pada perangkat android (Smyth, 2015, p. 33). AVD terdiri dari:

1. Sebuah Profil perangkat keras. Kita bisa mengatur opsi untuk menentukan fitur *hardware emulator*. Misalnya menentukan apakah menggunakan perangkat kamera, apakah menggunakan *keyboard QWERTY* fisik atau tidak, berapa banyak memori internal dan lainnya.
2. Sebuah pemetaan versi Android dimana kita dapat menentukan versi dari *platform* Android yang akan berjalan pada *emulator*.
3. Pilihan lainnya seperti kita dapat menentukan *skin* yang ingin digunakan pada *emulator* yang memungkinkan untuk menentukan dimensi layar, tampilan, dan sebagainya

2.2.8 Flowchart

Flowchart adalah sebuah bagan alir yang menggambarkan jalannya sebuah sistem (Jogiyanto, 2005).

Menurut (Krismiaji, 2010), ia menyebutkan bahwa flowchart merupakan teknik analitis yang digunakan untuk menjelaskan aspek-aspek sistem informasi secara jelas, tepat dan logis.

2.2.9 UML

UML adalah Bahasa standar untuk membuat rancangan software. UML biasanya digunakan untuk menggambarkan dan membangun, dokumen artifak dari *software intensive system* (Booch, 2005, p. 7).

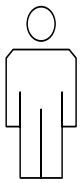
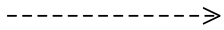
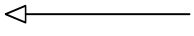
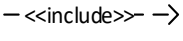
Menurut Nugroho (Nugroho, 2010, p. 6), UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma ‘berorientasi objek”. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

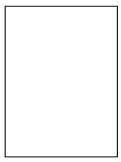
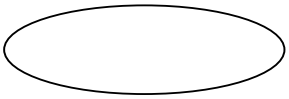
UML yang digunakan dalam pengembangan aplikasi kebudayaan di DIY, antara lain:

a. *Use Case Diagram*

Use case diagram menggambarkan *actor*, *use case* dan relasinya dengan sistem. *Use case* diagram menggambarkan siapa yang menggunakan sistem dan apa yang bisa dilakukan dalam sistem. Sebuah *use case* digambarkan sebagai elips horizontal dalam suatu diagram UML *use case*. Simbol-simbol dalam *use case* diagram dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol-simbol dalam *use case* diagram

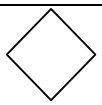
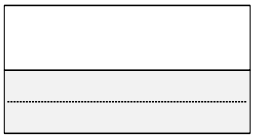
No.	Gambar	Nama	Keterangan
1.		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri (*independent*).
3.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4.		*Include*	Menspesifikasikan bahwa *use case* sumber secara ekspllisit.

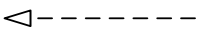
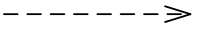
No.	Gambar	Nama	Keterangan
5.		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		*System*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pasa suatu titik yang diberikan.
8.		*Use case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
9.		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

b. Class Diagram

Class diagram menggambarkan struktur statis dari kelas dalam sistem dengan menunjukkan kelas, atribut, operasi, dan hubungan antar objek. *Class diagram* membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat. *Class diagram* terdiri atas 4 elemen, yaitu: nama kelas, atribut, operasi, dan relasi. Simbol-simbol dalam *class diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol-simbol dalam *class* diagram

No.	Gambar	Nama	Keterangan
1.		*Generalization*	Hubungan dimana anak (*descendent*) berbagi perilau dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2.		*Nary association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.

No.	Gambar	Nama	Keterangan
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Depedency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri.
7.		*Association*	Apa yang menghubungkan antara satu objek dengan objek lainnya.

c. Entity Relationship Diagram

Pengertian dari ERD (*Entity Relationship Diagram*) adalah suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

ERD untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol.

Pada dasarnya ada tiga komponen yang digunakan, yaitu:

1. Entitas

Entiti merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari entiti ini biasanya digambarkan dengan persegi panjang.

2. Atribut

Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut

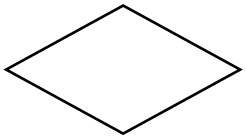
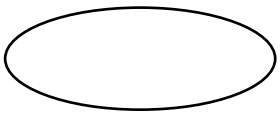
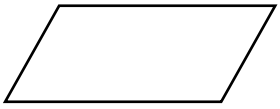
mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar atribut diwakili oleh simbol elips.

3. Hubungan atau Relasi

Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

Simbol-simbol dalam *entity relationship diagram* diagram dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol-simbol dalam *entity relationship* diagram

Notasi	Keterangan
	Entiti merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain.
	Relasi, yaitu Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.
	Atribut, yaitu untuk mendeskripsikan karakteristik dari entitas tersebut dan mengidentifikasikan isi elemen satu dengan yang lain.
	Garis, hubungan antara entity dengan atributnya dan himpunan entitas dengan himpunan relasi.
	Input atau output data, yaitu proses input atau output data, parameter, dan informasi.

2.2.10 Cpanel

Cpanel atau *Control Panel* adalah sebuah aplikasi yang ada pada *hosting* untuk mempermudah pengaturan *hosting*, aplikasi ini mirip seperti os, dan tentunya berbasis GUI, namun sebenarnya aplikasi ini merupakan kumpulan *script-script* dan *command* yang didesain sedemikian rupa untuk memudahkan *webmaster* dalam mengelola website (Andy Kristianto, 2014)

Fitur-fitur *control panel* antara lain:

a. Mengelola *hosting account* (melihat detail *account*, mengganti *password*, dll)
b. Mengelola domain (menambah domain, subdomain)
c. Mengelola *email* (*forwarding*, *email account*, *filter*, dll)
d. Mengelola Database
e. Mengelola File (FTP, Net2FTP, dll)
f. *Cron Jobs* (menjadwalkan *script*, *scheduler*)
g. Mengatur DNS.

2.2.11 *Database*

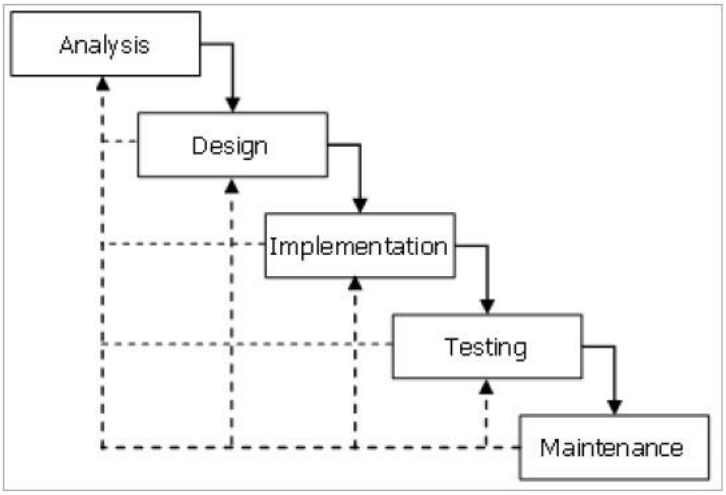
Pengertian database adalah kumpulan data yang saling berhubungan satu dengan yang lainnya, tersimpan di simpanan luar komputer dan digunakan perangkat lunak tertentu untuk memanipulasinya (Jogiyanto, 2001)

Menurut (Date, 2003) Database ialah koleksi “data operasional” yang tersimpan dan juga dipakai oleh sistem aplikasi dari suatu organisasi.

2.2.12 Metode *Waterfall*

Metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle*), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (*planning*), permodelan (*modeling*), konstruksi

(*construction*), serta penyerahan sistem ke para pelanggan atau pengguna (*deployment*), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2010).



Gambar 2.1 Metode *waterfall*

Tahapan metode waterfall, sebagai berikut:

a. *Requirement Analysis*

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

b. *System Design*

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain sistem disiapkan. Desain Sistem membantu dalam menentukan perangkat keras(*hardware*) dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

c. *Implementation*

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut unit, yang terintegrasi dalam tahap selanjutnya. Setiap unit dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai unit testing.

d. *Integration & Testing*

Seluruh unit yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing unit. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

e. *Operation & Maintenance*

Tahap akhir dalam model waterfall. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi unit sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

2.2.13 *Black Box Testing*

Menurut Rex Black, *black box testing* adalah suatu metode pengujian dimana *tester* hanya fokus pada apa yang seharusnya dilakukan oleh sistem (Black, 2009, p. 3). Sebuah tes dapat dikatakan berhasil ketika sebuah sistem dapat memproses data dan hasil yang ada sesuai dengan apa yang diharapkan. Ketika menggunakan metode *black box, tester* tidak perlu mengetahui bagaimana struktur dan desain data yang ada di dalam sistem. Mereka hanya melihat apakah sistem terjadi *bugs* atau tidak (Tim Riley, 2010, p. 270).

2.2.14 Skala Likert

Menurut (Djaali, 2008) skala likert ialah skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang suatu gejala atau fenomena pendidikan. Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei.