BAB II

KAJIAN PUSTAKA DAN LANDASAN TEORI

2.1 Kajian Pustaka

Maureen Pauline Kereh (2014) dalam penelitiannya yang berjudul "Aplikasi *E-Tourism* Kuliner Kota Manado Dengan Platform Android" mengatakan pada saat ini, dengan berbagai aplikasi mobile yang ada, belum bisa menemukan aplikasi yang membahas tentang wisata kuliner yang terdapat di kota Manado. Untuk itu, penulis melihat perlunya dibangun sebuah aplikasi yang akan membantu wisatawan dalam memperoleh berbagai informasi mengenai wisata kuliner kota Manado (Kereh, 2014).

Kharisma Nara As'ad (2017) dalam penelitiannya yang berjudul "Aplikasi Pencarian Terdekat Wisata Kuliner Kota Kediri Menggunakan Location-Based Services Pada Platform Android" mengatakan dalam banyaknya keanekaragaman jenis makanan dan tempat kuliner menarik di kota Kediri bisa membuat sebagian besar orang bingung untuk memilih makanan dan tempat-tempat kuliner yang ingin mereka kunjungi. Salah satu solusi yang bisa digunakan untuk pencarian adalah dengan menggunakan Sistem LBS (*location-based services).* Dengan LBS diharapkan lokasi kuliner tersebut dapat diakses dengan mudah oleh pengguna yang berasal dari luar kota maupun dalam kota (As'ad, 2017).

Acip Budi Hendratno (2012) dalam penelitiannya yang berjudul "Aplikasi Location Based Service Wisata Kuliner Yogyakarta Berbasis Android" mengatakan kebanyakan para wisatawan mencari informasi wisata kuliner dari internet, televisi, media cetak (koran, majalah) atau rekomendasi dari kerabatnya. Bila dikaji lebih dalam, hal tersebut tidaklah efektif karena pada saat berada di Yogyakarta mereka akan merasa kebingungan untuk mencari lokasi-lokasi tersebut. Berdasarkan latar belakang diatas maka akan dibuat aplikasi yang akan membantu para wisatawan

untuk menemukan lokasi serta informasi wisata kuliner di Yogyakarta yang akan dibangun pada smartphone dengan sistem operasi Android (Hendratno, 2012)

Hana Andiana Maulida (2014) dalam penelitiannya yang berjudul "Aplikasi Pencarian Lokasi Wisata Kuliner Berbasis Android" mengatakan kurangnya informasi tentang daftar lokasi wisata kuliner membuat wisatawan mengalami kesulitan untuk menentukan kuliner pilihan mereka. Berdasarkan uraian diatas, maka dibuatlah aplikasi yang diharapkan dapat membantu wisatawan yang berkunjung ke suatu kota dalam pencarian lokasi wisata kuliner. Dengan memanfaatkan fitur GPS yang disediakan pada *smartphone android,* sehingga wisatawan cukup mengaksesnya melalui *smartphone* yang dimiliki (Maulida, 2014).

Pilipus Kurniawan (2016) dalam penelitiannya yang berjudul "Perancangan Sistem Informasi Wisata Kuliner di Kota Salatiga Menggunakan Aplikasi *Hybrid* Berbasis Android" mengatakan seiring dengan perkembangan teknologi informasi, semakin bertambah pula kemampuan komputer dalam membantu menyelesaikan permasalahan-permasalahan di berbagai bidang. Di antaranya sistem informasi berbasis Android yang berfungsi sebagai media informasi tempat-tempat wisata dan kuliner di kota Salatiga. Mempertimbangkan kondisi tersebut maka penulis akan merancang sistem informasi dengan menggunakan aplikasi *hybrid* berbasis *Android*. dengan aplikasi ini akan membantu para pengunjung situs ataupun wisatawan lebih mudah mencari lokasi dan informasi tempat-tempat kuliner di kota Salatiga (Kurniawan, 2016)

Dari beberapa penelitian yang telah dilakukan, penelitian yang sudah ada hanya membahas aplikasi wisata kuliner berbasis android yang menampilkan informasi seputar kuliner yang umum. Dengan begitu pada penelitian ini akan dibuat aplikasi Hidden Kuliner di Yogyakarta berbasis *android*, yaitu aplikasi yang akan menyediakan informasi mengenai tempat kuliner yang tersembunyi dan jarang diketahui kebanyakan orang serta fitur *action call* atau telepon untuk menghubungi tempat wisata kuliner.

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*

Menurut Abelson (2009) android adalah sebuah aplikasi *platform mobile* yang *open source*. Android utamanya adalah produk Google, tetapi lebih tepatnya bagian dari *Open Handset Alliance*. *Open Handset Alliance* merupakan aliansi dari 30 organisasi yang berkomitmen untuk membawa sebuah perangkat seluler yang lebih baik dan terbuka untuk pasar. Android adalah *platform* terbuka pertama untuk perangkat *mobile*, Android adalah sebuah lingkungan perangkat lunak yang dibangun untuk perangkat-perangkat berbasis *mobile*. Android termasuk *kernel* berbasis *Linux*, aplikasi *end-user*, dan *framework* aplikasi. *User application* dibangun berbasiskan bahasa pemrograman Java.

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. (Abelson, 2009)

2.2.3. Arsitektur *Android*

Menurut Nazruddin Safaat H (2012) *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 tempat untuk 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. (H, 2012)

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 memperintahkan halaman web tapi java telah menjadi bahasa untuk pengembangan aplikasi skala *interprise* berbasis jaringan besar (Haryanto, 2011, p. 2).

2.2.5. Android Studio

Android Studio adalah sebuah *IDE* untuk Android *Development* yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangan dari *Eclipse IDE*, dan dibuat berdasarkan *IDE Java* populer, yaitu *IntelliJ IDEA*. Android Studio merupakan *IDE* resmi untuk pengembangan aplikasi Android.

Sebagai pengembangan dari *Eclipse*, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan *Eclipse IDE*. Berbeda dengan *Eclipse* yang menggunakan *Ant*, Android Studio menggunakan *Gradle* sebagai *build environment* (Syaputra, 2015).

2.2.6. JSON (*JavaScript Object Notation*)

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 (Saat ini sudah terbit edisi ke-5 – Juni 2011). JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk *C, C++, C#, Java*, *JavaScript, Perl*, *Python* dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. (Safaat, 2011)

2.2.7. MySQL (My Structured Query Language)

MySQL atau My Structured Query Language adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengelolaan datanya. Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses *database*, sehingga mudah digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan menengah kecil. MySQL merupakan *database* pertama kali didukung oleh bahasa pemprograman script untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan software pengembang aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembang aplikasinya menggunakan bahasa pemprograman script PHP. (Arief, 2011)

2.2.8. CPanel (Control Panel)

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.

Fitur-fitur *control panel* antara lain:

- Mengelola *hosting account (* melihat detail account, mengganti *password*, dll)
- Mengelola *domain (* menambah *domain, subdomain*)
- mengelola *email* (*forwarding, email account, filter, dll*)
- Mengelola database
- Mengelola *File* (*FTP, Net2FTP, dll*)
- Cron Jobs (menjadwalkan *script, scheduler*)
- Mengatur DNS

2.2.9. Flowchart

Menurut Mujono (2012) *flowchart* adalah gambaran atau bagan yanng memerlihatkan urutan dan hubungan antar proses beserta instruksinya. (Mujono, 2012). Simbol-simbol dalam *flowchart* Dipakai sebagai alat Bantu menggambarkan proses di dalam program. Dan dibagi menjadi tiga kelompok, antara lain:

a. Flow Direction Symbols

Simbol ini digunakan untuk menggabungkan antara simbol yang satu dengan simbol lainnya. Simbol-simbol *Flow Direction* dapat dilihat pada Tabel 2.1.

Tabel 2. 1 Simbol-simbol *Flow Direction*

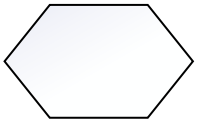
No.	Gambar	Nama	Keterangan
1.		*Off-line Connector*	untuk keluaratau masuk prosedure atau proses dalam lembar/halaman yang lain

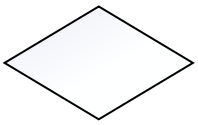
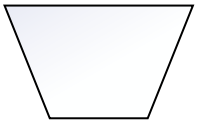
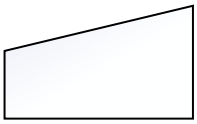
No.	Gambar	Nama	Keterangan
2.		*Connector*	untuk keluar atau masuk prosedur atau proses dalam lembar atau halaman yang sama

b. Processing symbols

Processing symbols menunjukkan jenis operasi pengolahan dalam suatu prosedur. Simbol-simbol *Processing* dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Simbol-simbol *Processing*

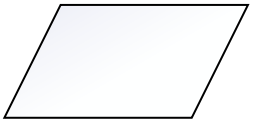
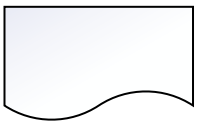
No.	Gambar	Nama	Keterangan
1.		*Terminal*	untuk permulaan atau akhir dari suatu program
2.		*Predefined Process*	untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storageelemen yang bergantung pada elemen yang tidak mandiri (*independent*).
3.		*Process*	yang menunjukkan pengolahan yang dilakukan oleh komputer)

No.	Gambar	Nama	Keterangan
4.		*Decision*	untuk kondisi yang akan menghasilkan beberapa kemungkinan jawaban atau aksi
5.		*Manual Operation*	yang menunjukkan pengolahan yang tidak dilakukanoleh komputer
6.		*Keying Operation*	operasi dengan menggunakan mesin yang mempunyai keyboard

c. Input-output symbols

Input-output symbols untuk menyatakan jenis peralatan yang digunakan sebagai media input atau output. Simbol-simbol *Input-output* dapat dilihat pada Tabel 2.3.

Tabel 2. 3 Simbol-simbol *Input-output*

No.	Gambar	Nama	Keterangan
1.		*input-output*	yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya
2.		*document*	yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas

No.	Gambar	Nama	Keterangan
3.		*disk and on-line storage*	untuk menyatakan input berasal dari disk atau output disimpan ke disk
4.		*punched card*	yang menyatakan input berasal dari kartu atau output ditulis ke kartu

2.2.10. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) 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.

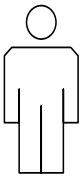
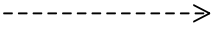
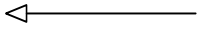
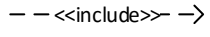
UML yang digunakan dalam pengembangan aplikasi Hidden Kuliner berbasis android, antara lain:

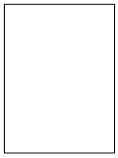
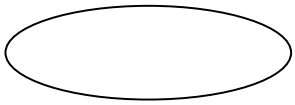
a. *Use Case Diagram*

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 2.4.

Tabel 2. 4 Simbol-simbol dalam *use case diagram*

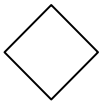
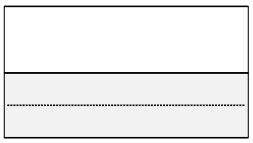
No.	Gambar	Nama	Keterangan
1.		*Actor*	menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2.		*Dependency*	hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung pada elemen yang tidak mandiri (*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.

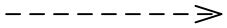
No.	Gambar	Nama	Keterangan
5.		*Extend*	menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6.		*Association*	yang menghubungkan antara objek satu dengan objek lainnya.
7.		*System*	menspesifikasikan bahwa *use case* target memperluas 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, attribut, operasi, dan relasi. Simbol-simbol dalam class diagram dapat dilihat pada Tabel 2.5.

Tabel 2. 5 Simbol-simbol dalam *class diagram*

No.	Gambar	Nama	Keterangan
1.		*Generalization*	hubungan dimana anak (*descendent*) berbagi 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.

No.	Gambar	Nama	Keterangan
4.		*Collaboration*	deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
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*	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

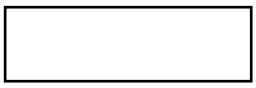
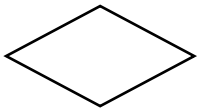
Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar atribut diwakili oleh simbol elips.

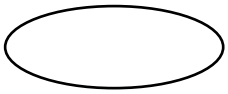
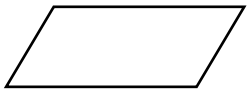
3. Hubungan atau Relasi

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

Simbol-simbol dalam *entity relationship diagram* diagram dapat dilihat pada Tabel 2.6.

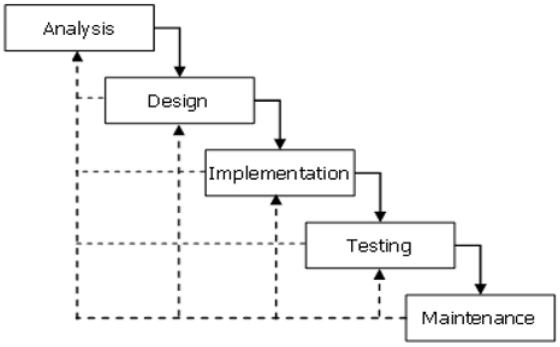
Tabel 2. 6 *Simbol-simbol dalam entity relationship diagram*

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.

Notasi	Keterangan
	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.

2.2.11. 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. *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. *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. 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. 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.

2.2.12. Black Box Testing

Menurut Rex Black (2009), *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).

2.2.13. Skala Likert

Teknik pengolahan data untuk variabel bebas dapat menggunakan pengukuran dengan 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 (Djaali, 2008).