BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Salah satu penerapan dalam metode pengumpulan data adalah studi pustaka, studi pustaka bermanfaat agar menghindari pembuatan ulang, mengidentifikasi metode yang pernah dilakukan serta untuk mengetahui peneliti lain yang mempunyai area yang sama dalam bidang ini. *Literature review* merupakan suatu survey *literature* tentang penemuan-penemuan yang telah dilakukan oleh penelitian sebelumnya yang berhubungan dengan topik penelitian.

Penelitian yang dilakukan oleh (Versianita & dkk., 2012) yang berjudul "Perancangan Sistem Antrian Pelayanan Rawat Jalan pada Rumah Sakit Islam Ibnu Sina Yarsi Sumber Padang Panjang Menggunakan PHP dan *MySql*", sistem antrian pelayanan ini dibuat dengan teknologi pemrograman PHP dan teknologi penyimpanan data *MYSQL*. Untuk dapat mengantri pasien diharuskan datang untuk registrasi, setelah registrasi pasien dipersilahkan menunggu untuk mendapatkan pelayanan. Kekurangan sistem antrian ini terletak pada proses pengambilan nomor antrian, dimana untuk dapat mengantri pasien diharuskan mengambil nomor antrian dengan datang terlebih dulu ketempat pengambilan nomor.

Penelitian yang dilakukan (Khumaesi, 2011) dari Universitas Islam Negeri Syarif Hidayatullah Jakarta berjudul "Aplikasi Sistem Antrian Kapal di Berlian Jasa Terminal Indonesia (PT. BJTI) Dermaga Surabaya Berbasis *Website*", Sistem aplikasi yang dibuat menggunakan bahasa pemrograman PHP dan bahasa

pengoperasian basis data *MYSQL*. Fitur-fitur pada sistem aplikasi kapal ini dapat digunakan untuk pengabilan data, penyimpanan data, pengaturan jadwal.

Penelitian yang dilakukan oleh (Nurwanto, 2009) dari Universitas Esa Unggul berjudul "Analisis dan Perancangan Sistem Informasi Antrian Pengecatan Barang pada CV. Bangkit Bersama", Sistem aplikasi yang dibuat menggunakan bahasa pemrograman *VB.NET* dan bahasa pengoperasian basis data *Microsoft Access* 2007. Sistem ini dapat memberikan pemberitahuan data barang yang lebih akurat dan evisien kepada para karyawan CV. Bangkit Bersama.

Penelitian yang dilakukan oleh (Nurhidayah, 2008) dari Universitas Esa Unggul berjudul "Analisis dan Perancangan Sistem Informasi Antrian pada PT. Bank Rakyat Indonesia (Persero) Tbk. Unit Pasar Timbul", Sistem aplikasi yang dibuat menggunakan bahasa pemrograman *Visual Basic* dan bahasa pengoperasian basis data *Microsoft Access* serta *Active Report* untuk menangani hasil *report.*

Dari beberapa sumber *literature review* di atas, dapat diketahui bahwa penelitian tentang pemecahan masalah antrian dengan merancang sebuah sistem terkomputerisasi sudah banyak dibahas. Meski demikian masih terdapat kekurangan pada masing-masing penelitian. Tujuan dilakukan penelitian ini adalah untuk menutupi kekurangan umum yang ada pada penelitian sebelumnya, yaitu kekurangan saat ingin mengantri, pengantri diharuskan datang untuk mengambil nomor antrian. Kurangnya kebebasan pengantri untuk mengambil nomor antrian dari manapun dinilai menjadi sebuah kekurangan.

2.2. Landasan Teori

Untuk mendukung pembuatan laporan ini, maka perlu dikemukakan hal atau teori-teori yang berkaitan dengan permasalahan dan ruang lingkup pembahasan sebagai landasan membangun laporan ini.

2.2.1 Antrian

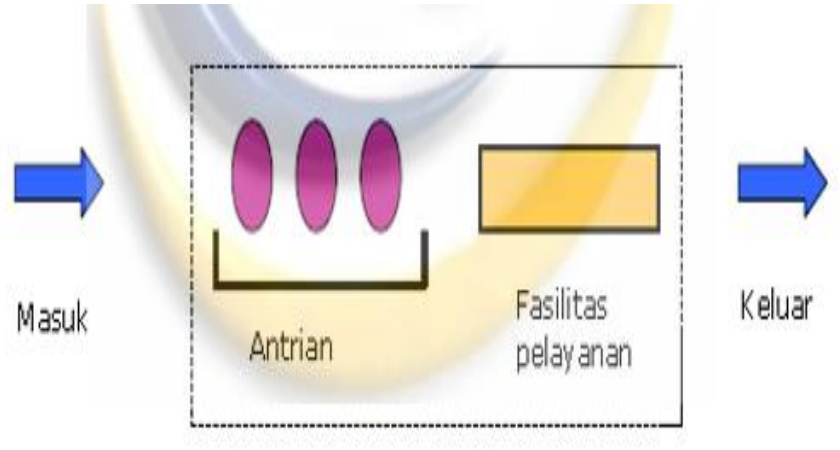
Antrian ialah suatu garis tunggu dari pelanggan (satuan) yang memerlukan layanan dari satu atau lebih pelayan (fasilitas layanan). Pada umumnya, sistem antrian dapat diklasifikasikan menjadi sistem yang berbeda – beda di mana teori antrian dan simulasi sering diterapkan secara luas (Siagian, 1987).

A. Teori Antrian

Teori tentang antrian diketemukan dan dikembangkan oleh A. K. Erlang, seorang insinyur dari *Denmark* yang bekerja pada perusahaan telepon di *Kopenhagen* pada tahun 1910. Erlang melakukan eksperimen tentang fluktuasi permintaan fasilitas telepon yang berhubungan dengan automatic dialing equipment, yaitu peralatan penyambungan telepon secara otomatis.

B. Komponen Sistem Antrian

Struktur umum dari mode antrian yang memiliki dua komponen utama yaitu: (1) Garis tunggu atau sering disebut antrian, dan (2) Fasilitas pelayanan. Pelanggan atau konsumen menunggu untuk memasuki fasilitas pelayanan, menerima pelayanan, dan akhirnya keluar dari sistem pelayanan. Selain Komponen utama struktur dari model antrian memiliki komponen lain. Adapun struktur antrian dapat dilihat pada gambar 2.1.



Gambar 2.1 Struktur Umum Model Antrian

2.2.2 Sistem Operasi *Android*

Android menyediakan kerangka kerja aplikasi yang kaya yang memungkinkan Anda untuk membangun aplikasi *inovatif* dan *game* untuk perangkat *mobile* di lingkungan bahasa *Java*. Dokumen yang tercantum di *navigasi* sebelah kiri memberikan rincian tentang bagaimana membangun aplikasi menggunakan berbagai API *android*. (https://developer.android.com/guide/index.html).

Pada awalnya, *Google inc*. Membeli *android Inc,* pendatang baru yang membuat perangkat lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuk lah *Open Handset Aliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google*, *HTC*, *Inter*, *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 lisensei perangkat lunak dan standar terbuka perangkat seluler.

Berikut adalah kelebihan dan kekurangan dari sistem operasi *Android* yaitu:

A. Kelebihan *Android*

1. User Friendly

Kata *User Friendly* sangat melekat pada sistem operasi *Windows* milik *Microsoft*, ibarat pengguna dengan sangat mudah mengoperasikan komputernya hanya dengan belajar beberapa hari bahkan beberapa jam saja, dan ini juga melekat pada *Android* yang berjalan pada *Smartphone*.

2. Notifications

Pengguna dapat dengan sangat mudah mendapatkan *notifikasi* dari *samartphone android* dengan mengatur beberapa akun pada aplikasi.

3. Tampilan

Android tidak kalah bagus dari *IOS* milik *Apple*, karena memang dari awal *Android* hampir mengusung teknologi *IOS*, hanya saja dapat dibilang ini versi murahnya.

4. Open Source

Operating system ini memang dibuat *open source* oleh penciptanya karena memang berbasis *kernel Linux* dan sangan banyak *Custom Room* yang dibuat untuk masing-masing perangkat *android*.

5. Aplikasi

   Sangat banyak aplikasi yang disajikan bahkan jutaan pilihan aplikasi yang menarik dari yang gratis hingga berbayar.

B. Kekurangan *Android*

1. *Update System*

   Untuk melakukan *update system* dapat dibilang tidak mudah Pengguan harus menunggu dari masing-masing *vendor* untuk merilis *Update* Versi yang terbaru.

2. Baterai Cepat Habis

   Masalah ini sering terjadi jika pengguna menyalakan paket data dan menggunakan *widget* serta apilkasi yang berjalan secara berlebihan.

3. *Lag*

   Sebenarnya ada kaitannya dengan spesifikasi dari masing-masing perangkat, namun ada kalanya *Android* ini tidak bersahabat dengan beberapa aplikasi dikarenakan Ram ataupun prosessor yang kurang memadai.

2.2.3 Bahasa Pemograman *Java*

(Wikipedia) Pemrograman *Java* adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer atau *smartphone*. Bahasa pemrograman ini banyak mengadopsi sintaks yang terdapat pada bahasa *C* dan *C++* namun dengan sintaks model yang lebih. Aplikasi berbasis *Java* pada umumnya dikompile ke dalam p-*code* (*bytecode*) dan dapat dijalankan pada berbagai Mesin *Virtual Java* (JVM).

Java berdiri di atas sebuah mesin penterjemah (*interpreter*) yang diberi nama *Java Virtual Machine* (JVM). JVM inilah yang akan membaca kode bit (*bytecode*) dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu bahasa *Java* disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada *system* operasi tersebut terdapat JVM. Alasan utama pembentukan bahasa *Java* adalah untuk membuat aplikasi-aplikasi yang dapat diletakkan di berbagai macam perangkat elektronik, sehingga *Java* harus bersifat tidak bergantung pada *platform* (*platform independent*). Itulah yang menyebabkan dalam dunia pemrograman *Java* dikenal adanya istilah "*write once, run everywhere*", yang berarti kode program hanya ditulis sekali, namun dapat dijalankan di bawah kumpulan pustaka (*platform*) manapun, tanpa harus melakukan perubahan kode program.

Berikut adalah beberapa penjelasan tentang Kelebihan yang dimiliki oleh *java* yaitu:

1. *Multiplatform*

   Java dapat dijalankan dalam beberapa *platform* komputer dan sistem operasi yang berbeda.

2. OOP atau *Object Oriented Programming*

   Java memiliki *library* yang lengkap. *Library* di sini adalah sebuah kumpulan dari program yang disertakan dalam *Java*. Hal ini akan memudahkan pemograman menjadi lebih mudah karena *library* semakin beragam jika ditambah dengan karya komunitas *Java*.

3. *Multithread*

   Kemampuan suatu program komputer untuk mengerjakan beberapa proses dalam suatu waktu. *Thread* dalam *Java* memiliki kemampuan untuk memanfaatkan kelebihan *multi prosessor* apabila sistem operasi yang digunakan mendukung prosessor.

4. Dapat di *distribusi* dengan mudah

   Java memiliki *library* rutin yang lengkap untuk dirangkai pada *protocol* TCP/IP (seperti HTTP dan FTP) dengan mudah. Kemampuan *networking Java* lebih kuat dan lebih mudah digunakan. *Java* memudahkan tugas pemograman jaringan yang sulit seperti membuka dan sebuah soket koneksi. *Java* juga memudahkan pembuatan CGI (*Common Gateway Interface*).

5. Bersifat dinamis

   Java dirancang untuk beradaptasi dengan lingkungan yang sedang berkembang, *Java* bersifat dinamis dalam tahap *linking*. *Class* yang ada dapat di *link* sebatas yang diperlukan, apabila diperlukan modul kode yang baru dapat di *link* dari beberapa sumber, bahkan dari sumber dalam jaringan Internet.

   Berikut adalah beberapa penjelasan tentang kekurangan yang dimiliki oleh *java* yaitu:

1. Pada satu slogannya

   "*Tulis sekali dan jalankan dimana saja*" ternyata tidak sepenuhnya benar. Beberapa hal harus disesuaikan jika dijalankan pada platform yang berbeda.

Misalnya untuk J2SE dengan platform SWT-AWT *bridge* tidak berfungsi di *Mac OS X*.

2. Kemudahan aplikasi *Java* di dekompilasi.

   Dekompilasi adalah suatu proses membalikkan sebuah aplikasi menjadi kode sumbernya. Hal ini memungkinkan terjadi pada *java* karena berupa *bytecode* yang menyimpan bahasa tingkat tinggi. Hal ini terjadi pula pada *platform .NET* dari *Microsoft* sehingga program yang dihasilkan mudah dibajak kodenya karena sulit untuk disembunyikan.

3. Penggunaan memori yang banyak.

   Penggunaan memori untuk program berbasis *Java* jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti *C/C++* dan *Pascal*. Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena tren memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun

2.2.4 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi *Android*, berdasarkan *IntelliJ IDEA* Selain merupakan editor kode *IntelliJ* dan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi *Android*, misalnya:

- Sistem versi berbasis *Gradle* yang fleksibel.
- *Emulator* yang cepat dan kaya fitur.

- Lingkungan yang menyatu untuk pengembangan bagi semua perangkat *Android*.
- *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
- *Template* kode dan integrasi *GitHub* untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
- Alat pengujian dan kerangka kerja yang ekstensif.
- Alat *Lint* untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
- Dukungan *C++* dan *NDK*.
- Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

2.2.5 MySQL

Mysql (Kadir, 2008) adalah salah satu jenis *database server* yang menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*nya. Dengan menggunakan *script* PHP dan PERL *Software database* ini dapat berfungsi atau berjalan pada semua platform sistem operasi yang biasa digunakan *(Windows, Linux, OS/2,* berbagai varian *Unix*).

Mysql merupakan salah satu jenis program yang berfungsi untuk mengolah, menyimpan data dan manipulasi data di *server*. Di dalam suatu program terdapat suatu penggunaan *database*. *Mysql* merupakan sebuah *database* bahasa yang di kembangkan dari *Structured Query Language* (*SQL*) yang digunakan untuk metode berkomunikasi antara *script* program dengan *database server* dalam memasukan

atau mengambil data. *Mysql* termasuk dalam kelompok RDBMS(*Relational Database Management Sistem*).

2.2.7 *Web Service*

Web Service merupakan kumpulan aplikasi logika yang menyediakan data dan *service* bagi aplikasi-aplikasi yang lain (Danny Ryan dan Tommy Ryan, 2002). Adapun aplikasi terdistibusi tersebut dapat diakses oleh aplikasi-aplikasi *client* tanpa memperhatikan sistem operasi maupun bahasa pemrograman.

Service yang disediakan oleh komponen *Web Service* umumnya berupa operasi-operasi logika maupun operasi *query* yang dimanfaatkan oleh banyak *client* (orang/program lain). *Service* tersebut dapat dimanfaatkan secara langsung dan juga dapat dimanfaatkan oleh *web service* lain.

2.2.8 GCM (*Google Cloud Messaging*)

Google Cloud Messaging untuk *Android* (GCM) adalah layanan yang membantu pengembang mengirim data dari *server* untuk aplikasi mereka *Android* pada perangkat *Android*. Ini bisa menjadi pesan ringan memberitahu aplikasi *Android* bahwa ada data baru yang akan diambil dari *server*. Layanan *GCM* menangani semua aspek antrian pesan dan pengiriman ke aplikasi target *Android* berjalan pada perangkat target. GCM memungkinkan aplikasi *android* untuk mngirimkan pesan kepada *server* untuk melakukan *broadcast* sebuah notifikasi kepada seluruh *client* yang ada. Hanya membutuhkan sebuah *account Gmail* maka akan langsung dapat menggunakan fasilitas GCM ini (Santoso, 2014).

2.3 Perancangan Sistem

2.3.1 *UML* (*Unified Modeling Language*)

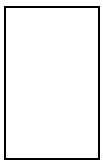
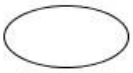
Menurut Nugroho (2010:6) "*UML* (*Unified Modeling Language*) adalah bahasa pemodelan (*modeling*) untuk sistem atau perangkat lunak yang berorientasi objek. Pemodelan sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan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

a. *Use Case* Diagram

Use case merupakan teknik menangkap kebutuhan-kebutuhan fungsional dari sistem baru atau sistem yang diubah. Setiap *use case* terdiri dari satu atau lebih skenario yang menerangkan bagaimana sistem berinteraksi dengan pengguna atau sistem yang lain untuk mencapai suatu sasaran bisnis tertentu. Dalam teknik ini tidak diterangkan cara kerja sistem secara *internal* maupun implementasinya. Yang ditunjukkan adalah langkah-langkah yang dilakukan pengguna dalam menggunakan perangkat lunak. Pada dasarnya ada dua jenis *use case* yaitu diagram *use case* dan naratif *use case* (Nyimas Artina:2006). Berikut adalah keterangan tentang tabel *Use Case* Diagram.

Tabel 2.1 *Use Case Diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan 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 padanya 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 *eksplisit*.

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 paket yang menampilkan sistem secara terbatas.
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 prilaku 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. *Entity Relationship* Diagram

Menurut Sutanta (2011:91), "*Entity Relationship* Diagram (ERD) merupakan suatu model data yang dikembangkan berdasarkan objek." *Entity Relationship Diagram* (ERD) digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pengguna secara logis.

Bagi perancang atau analis sistem, *Entity Relationship Diagram* (ERD) berguna untuk memodelkan sistem basis data yang nantinya akan dikembangkan. Model ini juga membantu perancang atau analis sistem pada saat melakukan analisis dan perancangan basis data karena model ini dapat menunjukkan macam data yang dibutuhkan dan kerelasian antardata didalamnya.

c. *Activity* Diagram

Activity Diagram merupakan diagram yang digunakan untuk menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.

Tabel 2.2 *Activity Diagram*

No	Gambar	Nama	Keterangan
1		*Actifity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.

No	Gambar	Nama	Keterangan
3		*Initial Node*	Node Awal untuk memulai sebuah proses *activity*.
4		*Actifity Final Node*	Node akhir untuk mengakhiri sebuah proses *activity.*
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

d. *Class* Diagram

Class diagram dibuat untuk menjelaskan hubungan antar kelas yang satu dengan kelas yang lainya. Di dalam beberapa kelas terdapat atribut yang menjelaskan komponen apa saja yang terdapat di dalam kelas tersebut dan terdapat operation yang berfungsi sebagai penjelas kegiatan apa saja yang dapat dilakukan oleh suatu aktor ketika berada didalam kelas tersebut. Terdapat beberapa simbol dari bagian *Class diagram*, simbol-simbol tersebut antara lain terlihat pada Tabel 2.3 berikut :

Tabel 2.3 Relasi *Class diagram*

Nama Relasi	Gambar	Keterangan
Association		Relasi antar kelas dengan makna umum. Biasanya disertai dengan *multiplicity* (keterangan banyak).
Directed Association		Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
Agregation		Relasi antar kelas dengan makna semua bagian (*whole-part*).
Dependency		Relasi antar kelas dengan makna kebergantungan antar kelas.
Generalisation		Relasi antar kelas dengan makna kebergantungan antar kelas.
Compotition		Relasi antar kelas dengan makna ada beberapa kelas yang merupakan bagian dari kelas utama.

2.3.2 Database

Raghu Ramaksrishan dan Johannes Gehrke (2004:3) menyatakan bahwa *database* adalah kumpulan data, umumnya mendeskripsikan aktivitas satu organisasi yang berhubungan atau lebih. Setiap *database* dapat berisi atau memiliki sejumlah objek *database* seperti *field, table, indeks*, dan lain-lain.

Menurut Connoly dan Begg (2002:14), pengertian basis data yaitu kumpulan koleksi data-data yang saling berhubungan secara logika yang isinya didesain untuk memenuhi kebutuhan informasi dari suatu perusahaan.

Ada beberapa istilah umum yang sering dipakai pada *database*, yaitu sebagai berikut :

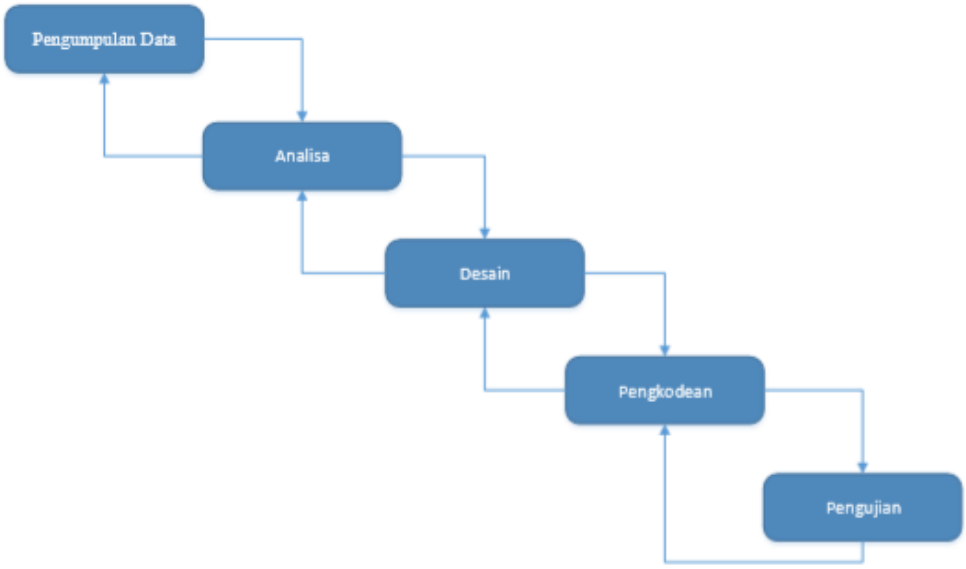
- *Field*, yaitu sekumpulan kecil dari kata atau sebuah deretan angka-angka.
- *Record*, yaitu kumpulan dari *field* yang berelasi secara logis.
- *File*, yaitu kumpulan dari *record* yang berelasi secara logis.

- *Entity*, yaitu orang, tempat, benda, atau kejadian yang berkaitan dengan informasi yang disimpan.
- *Attribute*, yaitu setiap karakteristik yang menjelaskan suatu *entity*.
- *Primary key*, yaitu sebuah *field* yang nilainya unik yang tidak sama antara satu *record* dengan *record* yang lain.

2.4 Metode Pengembangan Sistem

2.4.1 *Waterfall*

Menurut Sommerville (2011:30-31), tahapan utama dari *waterfall* model langsung mencerminkan aktifitas pengembangan dasar. Terdapat 5 tahapan pada *waterfall* model, yaitu *requirement* and *definition*, *system* and *software design*, *implementation* and *unit testing*, *integration and system testing*, dan *operation and maintenance*. *Waterfall* model atau *Classic Life Cycle* merupakan model yang paling banyak dipakai dalam *Software Engineering* (SE). Proses metode *waterfall* dapat dilihat pada gambar 2.1.



Gambar 2. 2 *Waterfall* model

Berikut adalah penjelasan dari tahapan-tahapan tersebut :

1 Tahap Pengumpulan Data

Pengumpulan data merupakan usaha yang dilakukan untuk memperoleh informasi dalam bentuk data yang dibutuhkan dalam penilitian. Metode pengumpulan data dapat dilakukan dengan: Metode Wawancara (*interview*), Metode Pengamatan (*observasi*), Dokumentasi (*documentation*).

2 Tahap Analisis Kebutuhan

Dalam langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dalam tahap ini bisa melakukan sebuah penelitian, wawancara atau *study literatur*.

3 Tahap Desain

Proses desain akan menterjemahkan syarat kebutuhan kesebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat koding. Proses ini berfokus pada : struktur data, arsitektur perangkat lunak, representasi *interface*, dan detail (algoritma) prosedural..

4 Tahap Pengkodean atau Penulisan *Code* Program

Coding merupakan penerjemahan *design* dalam bahasa yang bisa dikenali oleh komputer.Dilakukan oleh *programmer* yang akan meterjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem. Dalam artian penggunaan *computer* akan dimaksimalkan dalam tahapan ini.

5 Tahap Pengujian Program

Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Setelah melakukan analisa, *design* dan pengkodean maka sistem yang sudah jadikan digunakan oleh *user*.