BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem adalah kumpulan orang yang saling bekerja sama dengan ketentuan-ketentuan aturan yang sistematis dan terstruktur untuk membentuk satu kesatuan yang melaksanakan suatu fungsi untuk mencapai tujuan. Sistem memiliki beberapa karakteristik atau sifat yang terdiri dari komponen sistem, batasan sistem, lingkungan luar sistem, penghubung sistem, masukan sistem, keluaran sistem, pengolahan sistem, dan sasaran sistem.

Informasi adalah data yang diolah menjadi lebih berguna dan berarti bagi penerimanya, serta untuk mengurangi ketidakpastian dalam proses pengambilan keputusan mengenai suatu keadaan.

Sistem informasi merupakan suatu kombinasi teratur dari orang-orang, *hardware*, *software*, jaringan komunikasi, dan sumber daya data yang mengumpulkan, mengubah, dan menyebarkan informasi dalam sebuah organisasi (Yunaeti & Irviani, 2017).

Fungsi dari sistem informasi adalah:

1. Untuk meningkatkan aksebilitas data yang ada secara efektif dan efisien kepada pengguna, tanpa perantara sistem informasi.
2. Memperbaiki produktivitas pengembangan aplikasi dan *maintenance* sistem.
3. Menjamin tersedianya kualitas dan keterampilan dalam memanfaatkan sistem informasi secara kritis.
4. Mengidentifikasi kebutuhan mengenai keterampilan pendukung sistem informasi.
5. Mengantisipasi dan memahami akan konsekuensi ekonomi.
6. Menetapkan investasi yang akan diarahkan pada sistem informasi.
7. Mengembangakan proses perencanaan yang efektif.

Komponen-komponen dari sistem informasi adalah sebagai berikut:

1. *Input*, adalah data yang masuk ke dalam sistem informasi.

2. Model, adalah kombinasi prosedur, logika, dan model matematika yang memproses data yang tersimpan di basis data dengan cara yang sudah ditentukan untuk menghasilkan *output* yang diinginkan.
3. *Output*, adalah hasil informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.
4. Teknologi, adalah alat dalam sistem informasi, teknologi digunakan dalam menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan *output* dan memantau pengendalian sistem.
5. *Database*, adalah kumpulan data yang saling berhubungan yang tersimpan di dalam komputer.
6. *Control*, adalah komponen yang mengendalikan gangguan terhadap sistem informasi.

2.2 System *Development Life Cycle* (SDLC)

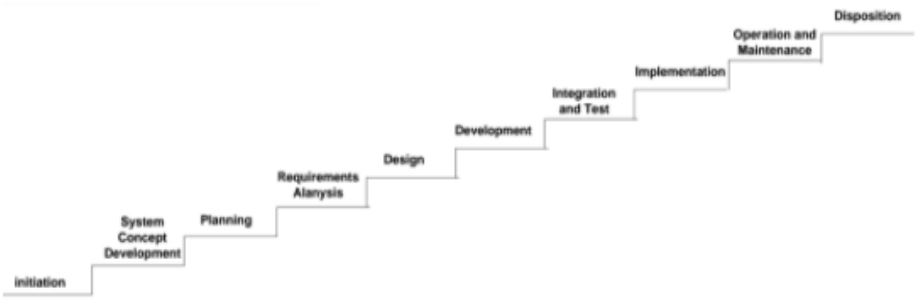
System Development Life Cycle atau yang biasa disebut dengan SDLC, merupakan sebuah metode yang digunakan untuk mengembangkan sebuah sistem. SDLC adalah sebuah proses logika yang digunakan oleh seorang *system analist* untuk mengembangkan sebuah sistem informasi yang melibatkan *requirements*, *validation*, *training*, dan pemilik sistem (Mulyani, 2016).

Merujuk pada sistem komputer atau informasi, SDLC adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Meskipun SDLC atau *life cycle models* sudah digunakan sejak lama, tidak ada perjanjian atau aturan khusus untuk setiap tahapannya dalam proses pengembangan. Akan tetapi tahapan tersebut merujuk pada *requirements*, *analysis*, *design*, *implementation*, dan *installation* (Britton & Doake, 2005).

SDLC identik dengan teknik pengembangan sistem *waterfall*, karena tahapannya menurun dari atas ke bawah. Berikut ini merupakan tahapan dari SDLC (Mulyani, 2016):

1. *Planning*
2. *Analysis*
3. *Design*
4. *Implementation*
5. *Use*

Seiring dengan perkembangan dunia teknologi dan pemikiran manusia, kelima tahap di atas juga mengalami pengembangan. Tahapan tersebut tidak lagi kaku atau mengharuskan pengembangan agar selalu mengikuti langkah dengan prioritas tersebut. Interupsi bisa saja dilakukan diantaran tahap-tahap pengembangan, misalnya sebagian tim sudah melakukan desain untuk membuat *prototype* dari sistem, atau sudah melakukan pengembangan sistem, baik dalam bentuk pembuatan program maupun rancangan sistem baru. Berikut ini ilustrasi SDLC:



Gambar 2.1 Tahapan Pengembangan Sistem SDLC *Waterfall* (Mulyani, 2016)

Pada Gambar 2.1 diberikan gambaran sebanyak 10 tahapan dalam melakukan pengembangan sistem. Namun pada kondisi-kondisi tertentu tahapan-tahapan tersebut tidak perlu dilakukan keseluruhannya. Hanya tahapan yang perlu dilakukan saja.

Initiation / Planning, merupakan tahap di mana sistem digambarkan secara *global* beserta tujuan yang akan direncanakan terhadap sistem yang akan dikembangkan. Tahap ini identik dengan tahap analisis.

Requirement Gathering and Analysis, pada tahap ini analis mencoba untuk menguraikan permasalahan sistem dan menggambarkannya ke dalam beberapa diagram untuk menggambarkan situasi yang sedang berjalan, kemudian pada tahap ini juga analis mencoba mendesain sebuah solusi yang akan diberikan kepada *user*.

Design, pada tahap ini solusi-solusi yang sudah digambarkan secara *global* pada tahap *requirement gathering* dan *analysis* diuraikan secara *detail* baik dalam bentuk *diagram, layouts*, *bussiness rules*, dan dokumentasi-dokumentasi lain yang dibutuhkan.

Build or Coding, pada tahap ini sistem mulai dibangun atau dikembangkan. Tahap ini identik dengan pembuatan program aplikasi untuk mendukung sistem.

Testing, pada tahap ini sistem yang sudah dibangun atau dikembangkan dicoba oleh tim *tester* ataupun oleh *user*.

2.3 *Rapid Application Development* (RAD)

RAD atau *Rapid Application Development* merupakan sebuah metodologi atau *development lifecycle* yang dapat membantu proses pengembangan menjadi lebih cepat dan berkualitas lebih baik dibandingkan dengan *traditional lifecycle*. RAD didesain agar dapat mengoptimalkan pengembangan perangkat lunak sehingga dapat menyesuaikan perkembangan teknologi yang berevolusi cepat. Karena itulah istilah metodologi RAD dikenal menekankan tiga hal vital, yaitu *high speed*, *high quality*, dan *low cost* (Martin, 1991).

RAD dapat membantu proses pengembangan sistem menjadi lebih cepat karena model kerjanya yang bersifat iteratif atau berulang. Dimana tahap yang dilakukan dimulai dari *requirement, design*, dan *implementation* (Kendall & Kendall, 2002). Tahapan tersebut dapat dilihat pada Gambar 2.2 berikut ini.

Gambar 2.2 Tahapan Metode RAD (Kendall & Kendall, 2002)

Requirement Planning merupakan tahap awal yang dilakukan, di mana pada tahap ini dilakukan perencanaan kebutuhan sesuai dengan hasil diskusi yang dilakukan pada pertemuan antara *user* dan *analyst*. Keterlibatan kedua belah pihak sangat diperlukan di sini agar kebutuh informasi dan tujuan sistem dapat terdefinisikan dengan jelas.

Design System merupakan tahap yang menghasilkan sebuah desain yang dibuat sesuai dengan perencanaan awal. Pada tahap ini *user* memiliki keterlibatan

dalam proses menentukan desain seperti apa yang diinginkan. Biasanya pada tahap ini terjadi perbaikan-perbaikan atau masukan yang diberikan *user* agar desain dapat sesuai dengan keinginan dan terutama kebutuhan sistem.

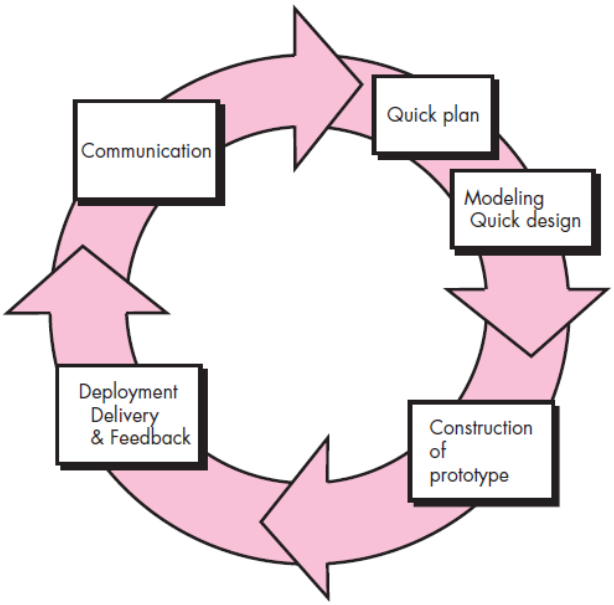
Implementation merupakan tahap yang paling banyak dilakukan oleh *programmer*. Di mana pada tahap ini akan dilakukan pengimplementasian desain - sistem dari tahap sebelumnya - dengan *coding*. Hasil dari implementasi tersebut kemudian akan diberikan kepada *user* agar dapat memberikan *feedback* atau persetujuan terkait sistem telah dibuat tersebut.

2.4 *Prototype*

Prototype merupakan metode pengembangan perangkat lunak, di mana versi awal dari perangkat lunak tersebut digunakan untuk mendemonstrasikan konsep yang telah direncanakan, mencoba berbagai pilihan desain, dan menggali lebih banyak permasalahan serta menemukan solusinya (Maulana, 2020).

Seringkali, *customer* hanya mendefinisikan satu set tujuan secara umum untuk perangkat lunak tanpa mengidentifikasikan persyaratan rinci untuk fungsi serta fitur. Dalam kasus lain, *developer* mungkin tidak yakin dengan efisiensi algoritma, kemampuan adaptasi dari sistem operasi, atau bentuk interaksi *human-machine* yang harus diambil. Dalam situasi tersebut, maupun situasi lainnya *prototyping paradigm* mungkin dapat menawarkan pendekatan terbaik.

Meskipun *prototyping* dapat digunakan sebagai *stand-alone process model*, *prototyping* lebih umum digunakan sebagai teknik yang dapat diimplementasikan dalam *process models*. Bagaimana pun cara *prototyping* diaplikasikan, metode ini dapat membantu *developers* dan *stakeholders* lainnya untuk lebih memahami apa yang akan dibangun ketika *requirements* tidak jelas (Pressman, 2010).



Gambar 2.3 *Prototyping* oleh (Pressman, 2010)

Berdasarkan Gambar 2.3, tahap-tahap pengembangan *Prototype* adalah sebagai berikut (Pressman, 2012):

1. *Communication*

   Dimulai dengan komunikasi, *developer* akan bertemu dengan *stakeholders* lainnya untuk menjelaskan tujuan secara keseluruhan untuk perangkat lunak, mengidentifikasikan *requirements* apa saja yang diketahui, dan menguraikan bagian mana saja yang perlu dijelaskan lebih lanjut.

2. *Quick Plan, Modelling, and Quick Design*

   Setelah tujuan perangkat lunak terdefinisikan secara umum maka tahap perancangan akan dilakukan. Tahap ini berfokus ada perancangan antarmuka *user* akhir dalam bentuk *mockup* atau desain tampilan. *Quick design* mengarah kepada *construction of prototype*.

3. *Modelling Quick Design*

   Pada tahap ini dilakukan pemodelan dengan UML *Diagram* seperti *Use Case Diagram, Activity Diagram, Sequence Diagram,* dan *Class Diagram.*

4. *Construction of Prototype*

   Pada tahap ini dibuat *prototype* yang mewakili sistem yang akan dibangun sesuai dengan hasil perencanaan.

5. *Deployment Delivery and Feedback*

   Setelah *prototype* selesai dibuat maka *prototype* tersebuat diserahkan kepada *client* untuk dilakukan evaluasi. Pada tahap ini *client* dapat memberikan *feedback* apakah perangkat lunak sudah sesuai dengan kebutuhan. Dari hasil *feedback* tersebut *developer* dapat mengetahui apa saja yang harus diperbaiki dari *prototype*.

Setelah evaluasi dilakukan tahapan *communication* kembali terulang dan dilanjutkan dengan tahap-tahap selanjutnya hingga kepuasaan *client* terhadap perangkat lunak yang dibutuhkan tercapai.

Beberapa manfaat *prototype* adalah:

1. Mewujudkan sistem yang direncanakan dalam sebuah replika sistem yang akan berjalan serta menampung masukan dari pengguna agar dapat mencapai sistem yang sempurna.
2. *User* dapat menerima perubahan sistem dengan lebih baik sesuai dengan berjalannya *prototype* sampai dengan hasil akhir sistem yang dikembangkan.
3. *Prototype* dapat ditambah maupun dikurangi ketika proses *development* sedang berjalan. *Progress* yang terjadi disetiap tahapannya dapat diikuti langsung oleh *user*.
4. Dapat menghemat sumber daya dan waktu dalam menghasilkan produk yang lebih baik dan tepat guna untuk *user*.

2.5 Alat Bantu Perancangan Sistem

Dalam perancangan sistem, dibutuhkan alat bantu atau *tools* untuk membantu proses perancangan tersebut. Alat-alat yang digunakan dalam suatu perancangan sistem umumnya berupa gambaran yang terjadi atau sedang dilakukan dalam sebuah penelitian.

2.5.1 *Unified Model Language* (UML)

Unified Modeling Language selanjutnya disebut UML adalah sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem (Mulyani, 2016).

UML biasanya diaplikasikan untuk tujuan tertentu, antara lain (Muslihudin, 2016):

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisis dan mencari apa saja yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Pembangun utama UML adalah diagram. Beberapa diagram bersifat rinci (*timing diagram*) dan ada pula yang bersifat umum (misalnya *class diagram*).

Pada UML ada sembilan jenis diagram yaitu:

1. *Class Diagram*
2. *Package Diagram*
3. *Use Case Diagram*
4. *Sequence Diagram*
5. *Communication Diagram*
6. *Statechart Diagram*
7. *Activity Diagram*
8. *Component Diagram*, dan
9. *Deployment Diagram*

Kesembilan diagram ini tidak harus mutlak digunakan dalam pengembangan perangkat lunak, cukup dibuat sesuai dengan kebutuhan. Pada UML, penggunaan diagram-diagram lainnya juga sangat memungkinkan seperti *Data Flow Diagram*, *Entity Diagram*, dan sebagainya.

2.5.2 *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk *behavior* sistem informasi yang akan dibuat dan mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Sukamto & Shalahuddin, 2015).

Use case diagram digunakan untuk menunjukkan sifat dinamis dari suatu sistem, yang mana terdiri dari *use cases*, aktor, dan hubungan mereka satu sama

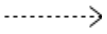
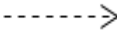
lain. *Use case diagram* digunakan pada desain tingkat tinggi yang menunjukkan kebutuhan dari sistem. Jadi, *use case diagram* mewakili fungsionalitas sistem dan alirnya (Singh, 2019).

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case* menurut (Sukamto & Shalahuddin, 2013):

Tabel 2.1 Simbol *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*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

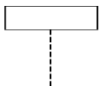
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan 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

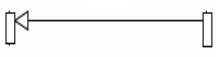
2.5.3 *Sequence Diagram*

Sequence diagram atau diagram sekuen menggambarkan *behavior* objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirim dan diterima antar objek (Sukamto & Shalahuddin, 2013). Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen menurut (Sukamto & Shalahuddin, 2013):

Tabel 2.2 Simbol *Sequence Diagram*

No.	GAMBAR	NAMA	KETERANGAN
1		*LifeLine*	Objek *entity*, antarmuka yang saling berinteraksi.
2		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

3		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi

2.5.4 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak (Sukamto & Shalahuddin, 2013).

Berikut adalah simbol-simbol yang ada pada diagram aktivitas menurut (Sukamto & Shalahuddin, 2013):

Tabel 2.3 Simbol Activity Diagram

No.	Gambar	Nama	Keterangan
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Actifity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

2.5.5 *Class Diagram*

Class diagram adalah diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas pada *class diagram* memiliki atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas, sedangkan operasi atau metode merupakan fungsi-fungsi yang dimiliki oleh suatu kelas. *Class diagram* dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai dengan rancangan di dalam *class diagram* agar antar dokumentasi perancangan dan perangkat lunak sinkron (Sukamto & Shalahuddin, 2013).

2.6 *Database*

Database adalah struktur yang dispesialisasikan khusus untuk memungkinkan sistem berbasis komputer agar dapat menyimpan, mengelola, dan mengambil data dengan sangat cepat (Coronel & Moris, 2016).

Sekumpulan informasi yang saling terkait pada media penyimpanan *database* dapat diakses oleh satu aplikasi, tanpa duplikasi, dan terstruktur secara independen dari aplikasi apa pun dengan tujuan memenuhi kebutuhan pengguna yang bisa berbeda dengan pengguna yang lain (Ouahab & Adel, 2018).

Keseluruhan bagian-bagian dari informasi yang terintegrasi dan terhubung secara logis - dengan sistem yang terkoneksi dengan datanya - disimpan sejajar pada *database*. Untuk dapat menggunakan *database* secara efektif maka desain *database* merupakan bagian yang penting. Konsep dari sistem *database* terdiri dari *databases*, *resources* komputer, dan dalam artian yang lebih luas, *database-administrators*, yang merancang dan memprogram *database* tersebut (Ayyavaraiah & Gopi, 2017).

Database didesain agar dapat menyimpan data, yang mana berisikan fakta yang masih mentah. Maksud dari kata "mentah" tersebut mengindikasikan bahwa fakta yang ada pada *database* belum diproses agar dapat menunjukkan arti data yang sesungguhnya. Misalnya ketika data dimasukkan melalui *form* dan tersimpan, data tersebut ditempatkan ke dalam *database* sebagai data mentah yang akan diproses agar menjadi informasi. Informasi adalah hasil dari proses data mentah dan kemudian menunjukkan arti sebenarnya dari data itu sendiri. Untuk menunjukkan maksud dari data tersebut, informasi membutuhkan konteks (Coronel & Moris, 2016). Misalkan di dalam *database* tersimpan data mentah berupa angka 80. Angka tersebut dapat berarti banyak hal seperti suhu, umur, atau jumlah barang. Tanpa konteks akan sulit untuk mengerti apa maksud dari data mentah tersebut. Jika konteksnya adalah suhu maka dapat dirincikan lagi apakah suhu tersebut dalam satuan *celcius* atau *fahrenheit*.

Database Management System (DBMS) adalah kumpulan program yang mengelola struktur *database* dan mengontrol akses ke data yang disimpan dalam *database*. Dengan arti lain, *database* menyediakan lemari arsip elektronik yang terorganisir dengan sangat baik di mana perangkat lunak (DBMS) membantu

mengelola isi lemari tersebut (Coronel & Moris, 2016).

Berdasarkan model ANSI/SPARC yang menunjukkan hubungan antara pengguna dan data yang disimpan secara fisik pada *mass storage* komputer terdapat tiga *level* abstraksi data, yaitu (Ayyavaraiah & Gopi, 2017):

1. *Outer level*, atau *user view*, yang memeriksa data dari sudut pandang pengguna.

   Level ini merupakan *level* abstraksi data yang tertinggi dan menggambarkan sebagian dari apa saja tentang data yang dapat dilihat dan dipakai dari keseluruhan *database*. Pandangan para *user database* di sini memiliki cara pandang yang berbeda-beda tergantung pada macam data apa saja yang tersedia atau dapat diakses oleh *user* tersebut. *User* tidak perlu tahu bagaimana sebenarnya dat tersebut disimpan.
2. *Conceptual level*, yang mencakup semua *user views*. Pada *level* ini *database* diberikan dengan skema logis.

   Level ini merupakan *level* abstraksi yang lebih rendah dari *outer level* dan memberikan gambaran tentang data apa (*what*) yang sebenarnya perlu disimpan ke dalam *database*, serta hubungan atau relasi yang terjadi di antara data dari keseluruhan *database*. Pada c*onceptual level*/*global logical data*, *user* tidak memperdulikan kerumitan dalam struktur level fisik lagi, penggambaran cukup dengan memakai kotak, garis, dan hubungan secukupnya.
3. *Inner level*, atau *physical level*, artinya presentasi data sebenarnya di komputer yang digunakan.

   Level ini merupakan *level* abstraksi yang paling rendah dan menjelaskan secara detail tentang bagaimana (*how*) data disimpan dalam kondisi sebenarnya atau diorganisasikan secara fisik atau aktual. Pada *level* ini struktur data digambarankan secara rinci dibutuhkan oleh seorang *system engineer*. *Level* ini pada umumnya digunakan oleh para pakar *software* dan *Physical Level* sering disebut sebagai *level eksternal* merupakan bentuk implementasi *conceptual*, yaitu suatu pandangan perancangan yang berkaitan

dengan permasalahan teknik penyimpanan data dalam *database* ke dalam media penyimpanan yang digunakan. Misalnya, *hardisk*, pita *magnetic*, dan sebagainya. Pandangan ini bersifat teknis dan lebih berorientasi pada mesin, yaitu berkaitan dengan organisasi berkas *database.*

2.7 Web

Website merupakan layanan atau alat tukar menukar data, informasi yang menggunakan konsep *client-server* di mana antara *user* dan *administrator* dapat saling memberikan data atau informasi yang dapat memudahkan keduanya (Azis *et al.*, 2019).

Informasi dan data yang disajikan dalam halaman *website* mempunyai teknologi layanan informasi, *multimedia*, (gambar, suara, animasi, dan video). *Website* berkembang sangat cepat dipengaruhi oleh faktor jaringan internet, di mana *website* tidak akan berfungsi tanpa adanya jaringan internet. Kemudahan dalam melakukan pencarian informasi dan pertukaran data yang menyebabkan pengguna sangat menyukai *website*.

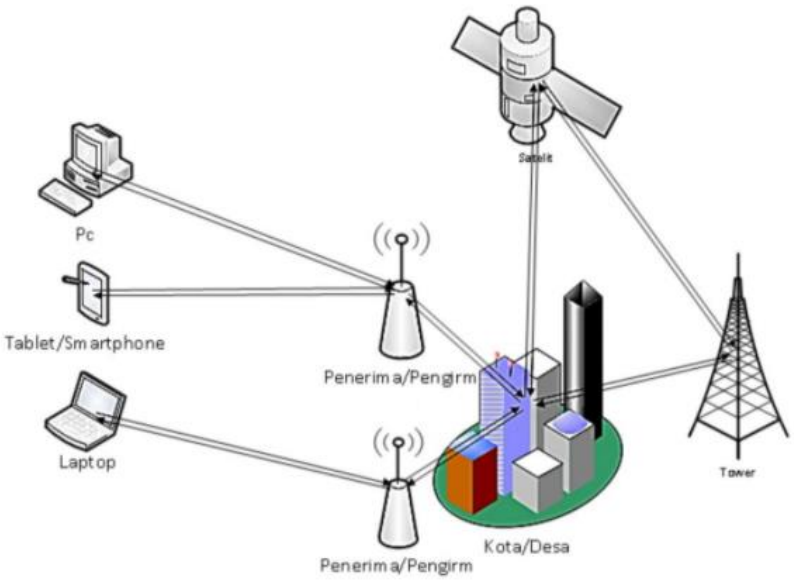
Dalam pengelompokan jenis *web*, lebih diarahkan berdasarkan pada fungsi, dan sifat bahasa pemrograman yang digunakan. Jenis-jenis *web* berdasarkan sifatnya adalah sebagai berikut (Sa'ad, 2020):

1. *Web* statis adalah *web* yang berisi konten yang tidak berubah-ubah. Maksudnya adalah isis dari dokumen *web* tersebut tidak dapat diubah secara cepat dan mudah. Ini karena teknologi yang digunakan untuk membuat dokumen *web* tidak memungkinkan dilakukan perubahan isi atau data. Teknologi yang digunakan *web* statis adalah jenis *clientside scripting*, seperti *HTML*, *Cascading Style Sheet* (*CSS*). Perubahan isi atau data halaman *web* statis hanya dapat dlakukan dengan cara mengubah langsung isinya pada *file* mentah tersebut atau mengubah *script*.
2. *Web* dinamis adalah jenis *web* yang konten atau isinya dapat diubah setiap waktu melalui halaman *admin* tanpa harus mengubah *file* mentah atau dikenal dengan istilah bongkar *script*/*coding*. Suatu *web*

yang banyak menampilkan animasi *flash* belum tentu termasuk *web* dinamis karena *web* dinamis dibuat dengan penyimpanan data pada *database*, seperti *MySQL*.

Internet berasal dari kata *Interconnection Networking* yang mempunyai arti bahwa hubungan antara berbagai macam komputer dan berbagai macam tipe (*platform*) komputer yang membentuk sistem jaringan yang mencakup seluruh dunia dengan jalur telekomunikasi seperti telepon, *wireless* bahan satelit.

Proses kerja *website* dapat dilihat Pada Gambar 2.4 berikut:



Gambar 2.4 Proses Kerja *Website* (Azis *et al*., 2019)

Koneksi internet pada *laptop, PC,* dan *tablet* harus dipastikan sudah menyala agar dapat memperoleh informasi melalui jaringan internet di mana satelit dan *tower* mengirimkan informasi menggunakan layanan sinyal yang dikirim dan diterima oleh sinyal internet.

2.8 PHP

PHP atau *PHP Hypertext Preprocessor* merupakan bahasa pemrograman yang digunakan untuk membuat *website* dinamis dan interaktif. Dinamis artinya, *website* tersebut bisa berubah-ubah tampilan dan kontennya sesuai dengan kondisi tertentu. Sebagai contoh, PHP bisa menampilkan tanggal dan hari saat ini secara berganti-ganti di dalam sebuah *website*. Interaktif artinya, PHP dapat memberikan *feedback* bagi *user*, misalnya menampilkan hasil pencarian produk) (Agung, 2018).

PHP merupakan salah satu bahasa pemrograman *scripting client-server* di mana hasilnya berupa sebuah halaman *web*. PHP bersifat *open source* sehingga dapat digunakan dan dikembangkan secara gratis oleh siapa pun. Untuk membuat halaman berbasis PHP, cukup dengan menyimpan *file* dengan ekstensi PHP (.php) (Tan & Caroline, 2018).

Saat ini PHP sudah mencapai versi 8.1 di mana beberapa fitur baru ditambahkan seperti *enums, fibers, Array unpacking with string keys*, tipe data baru yaitu *never*, fungsi baru yaitu *array_is_list* dan *fsync*, dan berbagai *performance improvements* serta perubahan kecil lainnya. Versi ini sedang berada dalam pengembangan aktif yang rencananya akan dirilis pada tanggal 25 November 2021. Tanggal ini masih belum tetap dan dapat berubah tergantung dengan tim inti yang melakukan pengembangan. Tetapi fitur-fitur tersebut sudah pasti ada dan memiliki kemungkinan akan bertambah (Brent, 2021).

2.9 MySQL

MySQL merupakan *server* yang melayani *database*. Untuk membuat dan mengolah *database* perlu digunakan bahasa pemrograman khusus yang disebut dengan *query* (perintah) SQL. *Database* dibutuhkan agar *input* dari *user* melalui *form* HTML dalam disimpan ke dalam *database* MySQL dan diolah oleh PHP (Agung, 2018).

Sistem *database* MySQL menggunakan arsitektur *client-server* yang memiliki kendali pusat di *server*. *Server* tersebut merupakan sebuah program yang dapat memanipulasi *database*. Program klien tidak melakukannya secara langsung, tetapi ia mengkomunikasikan tujuan pengguna kepada *server* dengan cara menuliskan *query* dengan bahasa SQL (*Sructured Query Language*). Program klien di-*install* secara lokal di mesin di tempat di mana pengguna mengakses MySQL. *Server* dapat di-*install* di mana saja, selama klien dapat melakukan koneksi dengan *server* tersebut. MySQL secara inheren merupakan sistem dengan *database* jaringan, sehingga setiap klien dapat berkomunikasi dengan *server* yang dijalankan secara lokal pada mesin pengguna atau dengan *server* yang dijalankan di tempat lain, bisa saja berada pada suatu mesin di benua lain (Sianipar, 2016).

2.10 Metode Pengujian Perangkat Lunak

Dalam pengembangan perangkat lunak, beberapa pengujian penting untuk dilakukan untuk memastikan perangkat lunak tersebut layak untuk diluncurkan dan digunakan oleh target pengguna. Terdapat banyak metode yang dapat dipilih dalam pengujian. Metode dapat dipilih untuk menyesuaikan kebutuhan *developers* dalam menguji hasil implementasi perangkat lunak tersebut.

2.10.1 *Black Box Testing*

Black box atau juga dikenal sebagai *behavioral*, *functional*, *opaque-box*, dan *closed-box*, adalah metode untuk pengujian desain. *Black box testing* memperlakukan sistem seperti "*black box*", sehingga tidak secara eksplisit mengetahui bagaimana struktur internal dari sistem tersebut. Dengan kata lain *tester* tidak perlu mengetahui cara kerja *internal* dari *black box* tersebut. Metode in berfokus kepada bagian fungsionalitas dari modul (John & Done, 2018).

Black box testing lebih condong terhadap pengujian fungsional. Di mana *black box* berarti *device* yang tidak dapat melihat ke dalam, dalam bahasa pengujian, yang berarti perangkat lunak harus diuji tanpa mengetahui apa yang ada di dalamnya. Pengujian fungsional dilakukan berdasarkan dari dokumen kebutuhan, karena di dalam dokumen tersebut berisi fungsi dan fitur yang diharapkan. *Project team* dan *customer* mempersiapkan *test cases* untuk dilakukan pengujian (Desai & Srivastava, 2016).

Metode *Black Box Testing* berfokus untuk menemukan kesalahan dalam beberapa kategori, diantaranya (Azis *et al*., 2019):

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database external*
4. Kesalahan performa
5. Kesalahan inisialisai dan terminasi

2.10.2 *User Acceptance Test (UAT)*

Pengujian UAT atau Uji Penerimaan Pengguna adalah suatu proses pengujian oleh pengguna yang dimaksudkan untuk menghasilkan dokumen yang

dijadikan bukti bahwa *software* yang telah dikembangkan telah dapat diterima oleh pengguna, apabila hasil pengujian (*testing*) sudah bisa dianggap memenuhi kebutuhan dari pengguna.

User Acceptance Testing merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah *staff*/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya (Perry, 2006).

Maka dari definisi tersebut, dapat dikatakan bahwa UAT merupakan pengujian yang dilakukan oleh pengguna dari sistem tersebut untuk memastikan fungsi-fungsi yang ada pada sistem tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna. Proses dalam UAT adalah pemeriksaan dan pengujian terhadap hasil pekerjaan. Diperiksa apakah *item-item* yang ada dalam dokumen *requirement* sudah ada dalam *software* yang diuji atau tidak. Diuji apakah semua item yang telah ada telah dapat memenuhi kebutuhan penggunanya.

2.11 Kajian Terkait

Berikut ini adalah penelitian serupa dengan penelitian yang akan dilakukan:

(Setioardi, 2019) telah melakukan penelitian dengan judul Perancangan Sistem Informasi Pengelolaan Barang Inventaris *Web* di SMAN 24 Kabupaten Tanggerang. Hasil dari penelitian tersebut adalah aplikasi yang dapat mengolah data barang, data pegawai, dan data peminjam. Serta dapat membuat laporan peminjaman dan laporan barang rusak. Sistem ini dapat diakses oleh *Admin* sebagai operator utama dan Kepala Sekolah yang memiliki hak akses untuk melihat data yang dikelola pada aplikasi tersebut. Aplikasi dikembangkan dengan menggunakan metode *Waterfall* dan dibangun berbasis Web.

(Agusvianto, 2017) telah melakukan penelitian dengan judul Sistem Informasi Inventori Gudang Untuk Mengontrol Persediaan Barang Pada Gudang Studi Kasus : PT.Alaisys Sidoarjo. Hasil dari penelitian ini adalah aplikasi yang mengelola data keluar-masuk barang di gudang PT.Alaisys Sidoarjo. Aplikasi ini dibuat berbasis web dan digunakan oleh *staff* bagian gudang sebagai *Admin* dan bagian penjualan, bagian pembelian, serta direktur sebagai *user* yang dapat melihat data.

(Indriani, 2015) telah melakukan penelitian dengan judul Sistem Informasi *Inventory* Alat Tulis Kantor (ATK) Menggunakan Metode *Waterfall* (Studi Kasus : Otoritas Jasa Keuangan (OJK)). Hasil dari penelitian ini adalah aplikasi yang dapat memudahkan Bagian Gudang dan Kabag. Logistik dalam memantau stok barang alat tulis kantor (ATK). Metode yang digunakan adalah *Waterfall* dan dibuat berbasis

(Suhandono, 2020) telah melakukan penelitian berjudul Sistem Informasi Pengelolaan Barang Persediaan Milik Negara Di Pusat Penilaian Pendidikan. Hasil dari penelitian ini adalah aplikasi yang dibangun agar mempermudah proses permintaan barang dan pemberian barang persediaan di Lingkungan Pusat Penilaian Pendidikan.