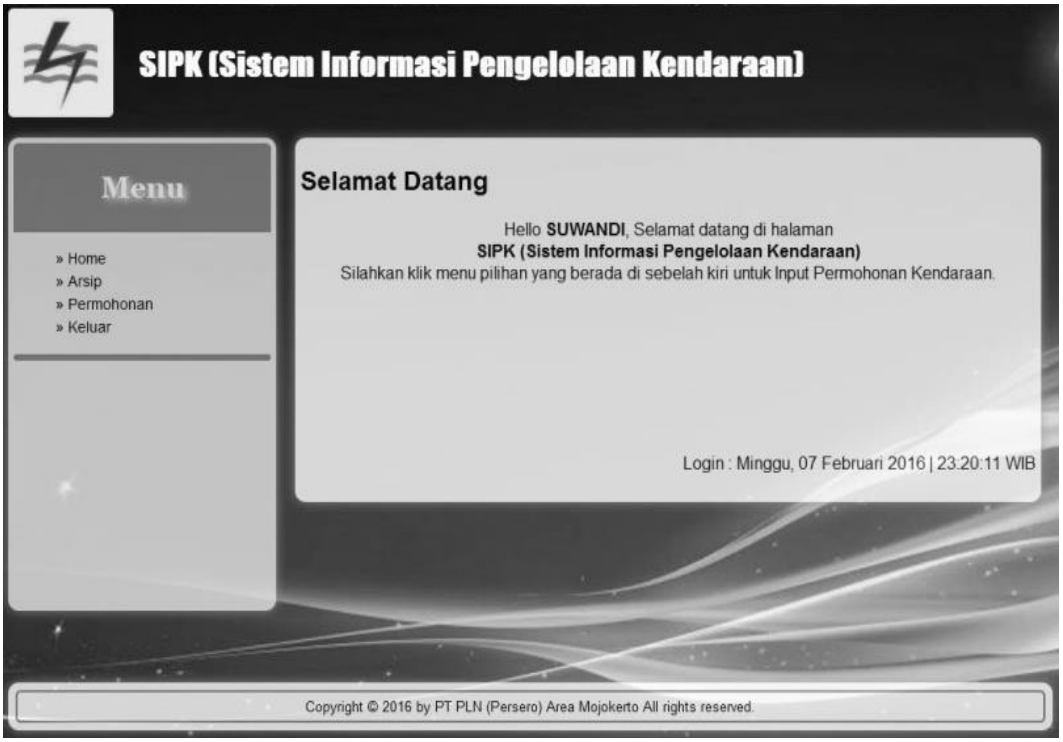
BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Ide dan proses pencarian solusi permasalahan tidak lepas dari penelitian sebelumnya yang digunakan sebagai acuan dan perbandingan. Pada penelitian sebelumnya terdapat materi yang hampir sama dengan tugas akhir ini. Agar lebih jelas dapat dilihat perbandingan penelitian terdahulu sebagai berikut.

Pengembangan Sistem Inventarisasi Barang pada Universitas Sahid Surakarta. Aritonang (2015), menjelaskan bahwa dari hasil penelitian yang dilakukan menghasilkan sistem inventaris barang di Universitas Sahid Surakarta yang bertujuan untuk dapat mempermudah dalam memasukan data barang, data suplier, spk dan serah terima barang, laporan data barang yang berhubungan dengan data barang ke *database* barang, suplier, spk dan serah terima barang. Pengujian sistem menggunakan metode *McCall* dengan sampel data yang digunakan berjumlah 30 orang responden yang terdiri 5 orang karyawan Universitas Sahid Surakarta, dosen sebanyak 5 orang, mahasiswa sebanyak 18 orang dan 2 bagian umum dari kalangan mahasiswa. Selanjutnya dari data sampel didapatkan nilai kualitas sistem inventarisasi barang adalah 77,288%. Diharapkan dengan adanya sistem inventaris barang di Universitas Sahid Surakarta, dapat membantu dalam proses penginputan barang secara cepat, tepat dan akuran tidak memakan waktu yang lama.

Rancang Bangun Sistem Informasi Pengelolaan Kendaraan di PT PLN (Persero) Area Mojokerto. Masrur, dkk. (2015) menjelaskan bahwa dari penelitian yang dilakukan, sistem informasi pengelolaan kendaraan di PT PLN (Persero) Area Mojokerto sudah bisa mencatat laporan penggunaan kendaraan dinas, mengetahui keluar masukknya kendaraan dinas, mampu mempercepat proses cetak laporan kendaraan dinas dan menghemat waktu dalam pengerjaan. Tampilan sistem informasi pengelolaan kendaraan tersebut disajikan pada Gambar 2.1.



Gambar 2.1. Tampilan Sistem Informasi Pengelolaan Kendaraan di PT PLN (Persero) Area Mojokerto (Masrur, 2015)

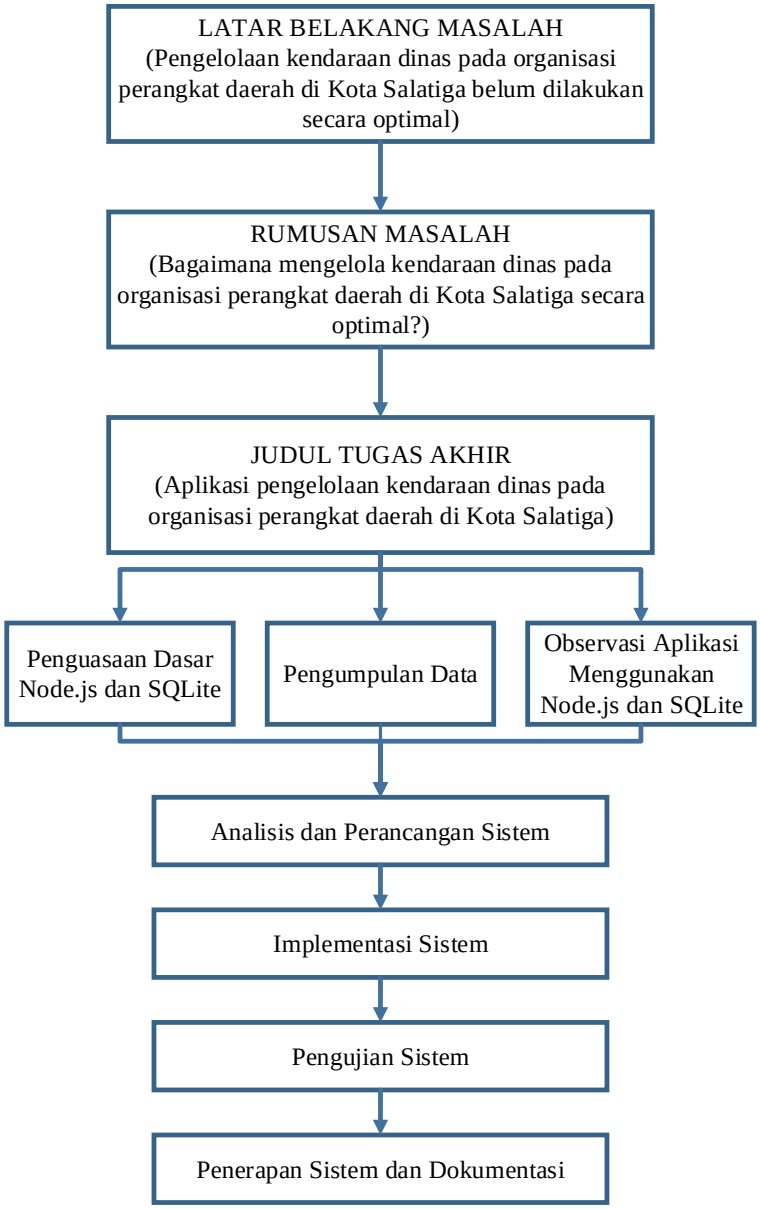
Rancang Bangun Aplikasi Pengelolaan Kendaraan Dinas di Sub Bagian Sarana dan Prasarana (SUBAGSARPRAS) Polres Sidoarjo. Cahyono, dkk. (2016) menjelaskan dalam penelitian yang dilakukan bahwa dengan adanya aplikasi pengelolaan kendaraan dinas di sub bagian sarana dan prasarana Polres Sidoarjo dapat memproses pendataan kendaraan dinas, pendataan pegawai (pemegang kendaraan dinas), pencarian kendaraan dinas dan pemegang kendaraan dinas, pengelolaan ijin pinjam pakai kendaraan dinas, memproses pembagian jatah BBM serta pencataan perawatan kendaraan dinas. Aplikasi pengelolaan kendaraan dinas tersebut juga memberikan informasi ke petugas apabila ada ijin pinjam pakai kendaraan dinas yang masa berlakunya hampir habis, pajak kendaraan dinas yang hampir hapis masa berlakunya dan pemegang kendaraan dinas yang harus mengembalikan kendaraan dinas. Tampilan aplikasi pengelolaan kendaraan dinas tersebut disajikan pada Gambar 2.2.

Gambar 2.2. Tampilan Aplikasi Pengelolaan Kendaraan Dinas di Sub Bagian Sarana dan Prasarana Polres Sidoarjo (Cahyono, 2016)

Aplikasi Monitoring Kendaraan Dinas dan Bahan Bakar Minyak pada PT. PLN (Persero) Penyaluran dan Pusat Pengatur Beban Sumatera Unit Pelayanan Transmisi Palembang Berbasis *Website*. Ramadhan (2017) melalui penelitian yang dilakukan telah membangun aplikasi Monitoring Kendaraan Dinas dan Bahan Bakar Minyak pada PT. PLN (Persero) Penyaluran dan Pusat Pengatur Beban Sumatera Unit Pelayanan Transmisi Palembang Berbasis Website, terdiri dari fungsi untuk petugas, admin, dan manajer yang berupa *form* data pegawai, data kendaraan, data supir, data pemesanan, dan data pengembalian. Penelitian tersebut menjelaskan bahwa dengan dibangunnya aplikasi monitoring kendaraan dinas dan bahan bakar minyak pada PT. PLN (Persero) Penyaluran dan Pusat Pengaturan Beban Sumatera unit Pelayanan Transmisi Palembang berbasis website, dapat mempermudah dalam pemakaian kendaraan dinas di PT. PLN (Persero) P3B Sumatera UPT. Palembang.

2.2 Kerangka Pemikiran

Tugas akhir ini disusun melalui beberapa tahapan dalam suatu kerangka pemikiran. Kerangka pemikiran tersebut dapat digambarkan pada Gambar 2.3.



Gambar 2.3. Kerangka Pemikiran Penelitian

Adapaun penjelasan dari kerangka pemikiran yang digunakan dalam menyusun tugas akhir ini, dapat diuraikan sebagai berikut :

1) Latar belakang masalah

Pengurus barang yang bertugas mengelola proses pemakaian barang milik organisasi perangkat daerah, belum secara optimal melaksanakan pengelolaan kendaraan dinas pada organisasi perangkat daerah mereka. Hal ini disebabkan belum adanya alat bantu yang mempermudah pengurus barang dalam mengelola kendaraan dinas.

2) Rumusan masalah

Bagaimana pengelolaan kendaraan dinas pada organisasi perangkat daerah di Pemerintah Kota Salatiga berjalan secara optimal?

3) Judul Tugas Akhir

Judul yang sekiranya tepat untuk menangani permasalahan yang ada pada organisasi perangkat daerah di Pemerintah Kota Salatiga.

4) Pengumpulan Data

Semua data yang dibutuhkan dikumpulkan, baik melalui wawancara dengan pengurus barang organisasi perangkat daerah maupun dengan observasi langsung ke beberapa organisasi perangkat daerah di Pemerintah Kota Salatiga.

5) Penguasaan Dasar (Node.js dan SQLite)

Membua aplikasi sederhana sebagai dasar percobaan dengan tujuan agar dapat lebih menguasai bahasa pemrograman Node.js dan *database* SQLite, sehingga hasil yang diharapkan akan lebih maksimal.

6) Observasi Aplikasi

Dilakukan pengamatan terhadap aplikasi yang sudah ada, baik dari karya ilmiah, buku atau internet, sehingga dapat dijadikan bahan referensi untuk membangun aplikasi.

7) Analisis dan Perancangan Sistem

Terhadap aplikasi yang akan dibangun dilakukan analisa dan perancangan seperti apa dan bagaimana desainnya serta apa saja *content* yang diinginkan, sehingga aplikasi dapat membantu memecahkan permasalahan pengelolaan kendaraan dinas pada organisasi perangkat daerah di Pemerintah Kota Salatiga.

Analisis dan perancangan sistem pada Tugas Akhir ini menggunakan metode terstruktur yang terdiri dari teknik normalisasi, *Data Flow Diagram* (DFD), *Entity Relationship Diagram* (ERD) dan *flowchart*.

8) Implementasi Sistem

Implementasi sistem merupakan tahan penerapan sistem berdasarkan hasil analisis dan perancangan sistem.

a. Perancangan *database* SQLite

Membuat *database* sesuai dengan kebutuhan sistem dari data-data yang diperoleh.

b. Perancangan aplikasi

Merancang aplikasi yang mudah digunakan dan sesuai dengan kebutuhan.

9) Pengujian Sistem

Pengujian sistem dilakukan bertujuan untuk mengetahui apabila ternyata masih terdapat kesalahan ataupun kekurangan pada aplikasi yang telah dikembangkan.

10) Penerapan Sistem dan Dokumentasi

Aplikasi telah siap untuk digunakan setelah melewati tahap pengujian dan kemudian membuat dokumentasi dari keseluruhann kegiatan penyusunan Tugas Akhir.

2.3 Landasan Teori

2.3.1 Aplikasi

Menurut Safaat (2012) perangkat lunak aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut sebagai suatu paket atau *suite* aplikasi (*application suite*). Contohnya adalah Microsoft Office dan Open Office.org, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan

setiap aplikasi. Sering kali, aplikasi ini memiliki kemampuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna.

2.3.2 Pengertian Pengelolaan

Kata "Pengelolaan" dapat disamakan dengan manajemen, yang berarti pula pengaturan atau pengurusan (Arikunto, 2013). Banyak orang yang mengartikan manajemen sebagai pengaturan, pengelolaan, dan pengadministrasian, dan memang itulah pengertian yang populer saat ini. Pengelolaan diartikan sebagai suatu rangkaian pekerjaan atau usaha yang dilakukan oleh sekelompok orang untuk melakukan serangkaian kerja dalam mencapai tujan tertentu.

Manajemen adalah suatu proses perencanaan dan pengambilan keputusan, pengorganisasian, memimpin dan pengendalian organisasi manusia, keuangan, fisik dan informasi sumber daya untuk mencapai tujuan organisasi secara efisiensi dan efektif. Fattah (2019) berpendapat bahwa dalam proses manajemen terlibat fungsi-fungsi pokok yang ditampilkan oleh seorang manajer atau pimpinan, yaitu perencanaan (*planning*), pengorganisasian (*organising*), pemimpin (*leading*), dan pengawasan (*controlling*). Oleh karena itu, manajemen diartikan sebagai proses merencanakan, mengorganising, memimpin, dan mengendalikan upaya organisasi dengan segala aspeknya agar tujuan organisasi tercapai secara efektif dan efisien.

Pengertian manajemen telah banyak dibahas para ahli yang antara satu dengan yang lain saling melengkapi. Stoner yang dikutip oleh Handoko menyatakan bahwa manajemen merupakan proses perencanan, pengorganisasian, pengarahan, dan pengawasan, usaha-usaha para anggota organisasi dan pengguna sumber daya organisasi lainya untuk mencapai tujuan organisasi yang telah ditetapkan. Stoner menekanan bahwa manajemen dititik beratkan pada proses dan sistem. Oleh karena itu, apabila dalam sistem dan proses perencanaan, pengorganisasian, pengarahan, penganggaran, dan sistem pengawasan tidak baik, proses manajemen secara keseluruhan tidak lancar sehingga proses pencapaian tujuan akan terganggu atau mengalami kegagalan.

Bedasarkan definisi manajemen tersebut secara garis besar tahap-tahap dalam melakukan manajemen meliputi melakukan perencanaan, pengorganisasian, pelaksanaan, dan pengawasan. Perencanaan merupakan proses

dasar dari suatu kegiatan pengelolaan dan merupakan syarat mutlak dalam suatu kegiatan pengelolaan. Kemudian pengorganisasian berkaitan dengan pelaksanaan perencanaan yang telah ditetapkan. Sementara itu pengarahan diperlukan agar menghasilkan sesuatu yang diharapkan dan pengawasan yang dekat. Dengan evaluasi, dapat menjadi proses monitoring aktivitas untuk menentukan apakah individu atau kelompok memperolah dan mempergunakan sumber-sumbernya secara efektif dan efisien untuk mencapai tujuan.

2.3.3 Kendaraan Dinas

Berdasarakan Peraturan Walikota Kota Salatiga Nomor 53 Tahun 2012, kendaraan dinas adalah kendaraan milik pemerintah daerah yang dipergunakan hanya untuk kepentingan dinas, terdiri atas kendaraan perorangan dinas, kendaraan dinas operasional/kendaraan dinas jabatan dan kendaraan dinas khusus/lapangan.

1) Kendaraan Perorangan Dinas

   Kendaraan perorangan dinas merupakan kendaraan dinas yang disediakan dan dipergunakan untuk Walikota dan Wakil Walikota. Jenis kendaraan perorangan dinas adalah sedan dan jeep.

2) Kendaraan Dinas Operasional/Jabatan

   Kendaraan dinas operasional/jabatan disediakan dan dipergunakan untuk kegiatan operasional perkantoran. Kendaraan dinas operasional/jabatan diperuntukkan bagi Ketua dan Wakil Ketua DPRD, eselon II, eselon III dan eselon IV. Jenis kendaraan dinas operasional/jabatan adalah sedan, minibus dan sepeda motor.

3) Kendaraan Dinas Operasional Khusus/Lapangan

   Kendaraan dinas operasional khusus/lapangan disediakan dan diperuntukkan untuk pelayanan operasional khusus/lapangan dan pelayanan umum. Kendaraan dinas operasional khusus/lapangan diperuntukkan bagi pegawai yang menjalankan tugas-tugas khusus/lapangan dan pelayanan umum. Jenis kendaraan dinas operasional khusus/lapangan adalah truk, pick up/box, station wagon, motor roda tiga, jeep, minibus, sepeda motor trail dan lain sebagainya.

2.3.4 Javascript

Javascript adalah salah satu bahasa pemrograman populer yang mampu membuat halaman web dapat berinteraksi dengan penggunanya. Javascript pertama kali dikembangkan pada pertengahan dekade 90an. Meskipun memiliki nama yang hampir serupa, javascript berbeda dengan bahasa pemrograman Java. Penulisan javascript dapat disisipkan di dalam dokumen HTML ataupun dijadikan dokumen tersendiri yang kemudian diasosiasikan dengan dokumen lain yang dituju.

Javascript mengimplementasikan fitur yang dirancang untuk mengendalikan bagaimana sebuah halaman web berinteraksi dengan penggunanya. Seperti tampilan pada *window* atau kendali pada *menu* dan *button*. Javascript juga dapat digunakan untuk memvalidasi sebuah *webform* pada *browser* sebelum informasi pada *form* tersebut dikirim ke *server*.

2.3.5 Node.js

Node.js pertama kali dikembangkan oleh Ryan Lienhart Dahl pada tahun 2009. Platform ini dibuat untuk memperoleh *performance* aplikasi yang tinggi dan mengoptimalisasikan lingkungan *concurrent* yang tinggi pula. Platform Node.js memungkinkan bekerja dengan keandalan tinggi dan dengan I/O *non-blocking*.

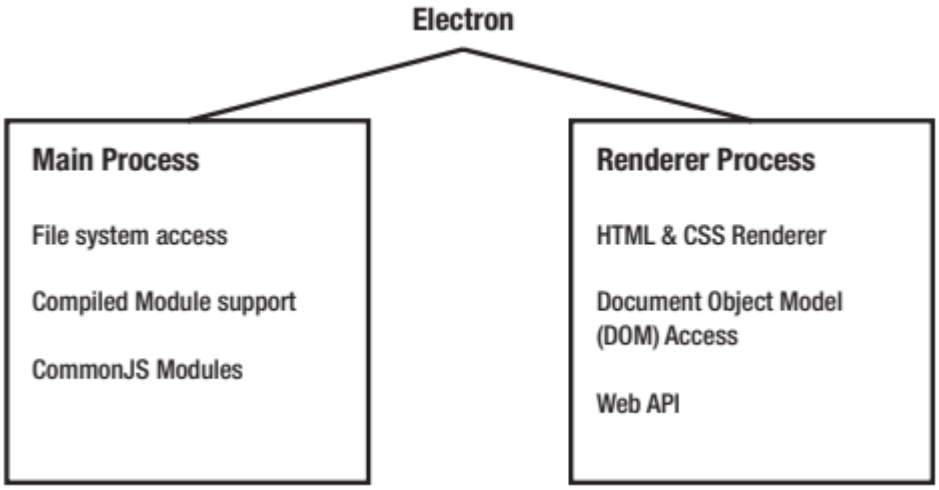
Node.js adalah sebuah platform yang dirancang untuk *webserver*. Aplikasi ini ditulis dalam bahasa javascript dan berbasis pada *event* (*event driven*). Tidak seperti kebanyakan bahasa javascript yang dijalankan di *client*, Node.js dieksekusi sebagai aplikasi *server*. Node.js memiliki efisiensi memori yang lebih baik apabila bekerja dalam keadaan muatan yang banyak. Pengguna tidak perlu khawatir dengan terjadinya *deadlock* karena tidak ada *lock* dalam Node.js. Aplikasi ini terdiri dari V8 Javascript *engine* buatan google dan beberapa modul bawaan yang terintegrasi (Nodejs.org). Di dalam Node.js terdapat operasi-operasi yang sifatnya *synchronous* dan *asynchronous*, tetapi keunggulan Node.js ini terdapat pada *asynchronous*nya atau bisa juga disebut dengan *non-blocking* I/O. Node.js menggunakan pendekatan *event-driven* berbasis *infinite event loop* dalam satu *thread* untuk mengurangi jumlah memori. Kiessling (2012) berpendapat bahwa Node.js merupakan model yang efisien untuk membangun sebuah aplikasi yang

harus bersinggungan dengan *concurrency* (dua hal yang bekerja secara bersamaan).

2.3.6 Electron

Electron merupakan sebuah *framework* yang memudahkan *developer* untuk membuat aplikasi *dekstop* lintas-perangkat (*cross-platform*) seperti pada Linux, Windows dan MacOS dengan menggunakan teknologi Javascript. Electron pada dasarnya menggunakan Node.js dan Chromium, dengan menggabungkan kemudahan membuat *website* dan kekuatan *native desktop app* yang dapat berinterkasi secara langsung dengan sistem operasi.

Aplikasi berbasis electron berjalan pada 2 proses berbeda, yaitu *main process* dan *render process*. Masing-masing dari kedua proses tersebut memilik tanggung jawab yang spesifik di dalam aplikasi. Sebagai contoh, akses terhadap API pada sistem operasi hanya dibatasi untuk *main process*, sedangkan akses ke *clipboard* pada sistem tersedia untuk *main process* dan *render process* (Chris & Leif : 2017). Gambaran terhadap 2 proses tersebut, dapat dilihat pada Gambar 2.4.



Gambar 2.4. Proses-Proses yang Mendukung Aplikasi Berbasis Electron (Chris & Leif, 2017)

2.3.7 Database

Database adalah kumpulan-kumpulan data-data yang saling terkait. Definisi *database* yang cukup umum, misalnya, kita dapat mempertimbangkan koleksi kata-kata yang membentuk halaman teks menjadi data terkait dan karenanya

untuk terbentuklah *database*. Namun, penggunaan umum dari istilah *database* lebih terbatas.

2.3.7.1 Database Management System

Database Management System (DBMS) adalah kumpulan program yang memungkinkan pengguna untuk membuat dan memelihara *database*. DBMS adalah tujuan umum dari sistem perangkat lunak yang memfasilitasi proses-proses mendefinisikan, membangun, memanipulasi, dan berbagi *database* antara berbagai pengguna dan aplikasi. Mendefinisikan *database* meliputi menentukan jenis data, struktur, dan keterbatasan dari data yang akan disimpan dalam *database*. Definisi *database* atau informasi deskriptif juga disimpan oleh DBMS dalam bentuk katalog *database* atau kamus yang disebut meta-data. Membangun *database* adalah proses menyimpan data pada beberapa media penyimpanan yang dikendalikan oleh DBMS. Manipulasi *database* termasuk fungsi seperti *query database* untuk mengambil data tertentu, memperbarui *database* mencerminkan perubahan dalam *miniworld*, dan menghasilkan laporan dari data. Berbagi *database* memungkinkan beberapa pengguna dan program untuk mengakses *database* secara bersamaan.

2.3.7.2 SQL

SQL adalah bahasa *database* yang komprehensif yaitu memiliki pernyataan untuk mendefinisikan data, *query*, dan *update*. Awalnya, SQL atau *Structured Query Language* disebut SEQUEL (*Structured Query Language English*) dan dirancang dan diterapkan di IBM Research sebagai antarmuka untuk eksperimental sistem database relasional yang disebut SISTEM R. SQL sekarang merupakan bahasa standar untuk DBMS relasional komersial.

2.3.7.3 SQLite

SQLite adalah sebuah *open source* dengan *database* relasional. Awalnya dirilis pada tahun 2000, SQLite dirancang untuk menyediakan cara yang nyaman untuk aplikasi untuk mengelola data tanpa *overhead* yang sering muncul dengan sistem manajemen database relasional khusus. SQLite memiliki reputasi baik karena sangat portabel, mudah digunakan, kompak, efisien, dan dapat diandalkan.

SQLite mendukung fitur *database* relasional standar seperti sintaks SQL, *transactions* dan *prepared statements*. Selain itu hanya memerlukan sedikit memori pada saat *runtime* (sekitar 250 KByte). SQLite mendukung tipe data *TEXT* (mirip dengan *String* di Java), *INTEGER* (mirip dengan *long* di Java) dan *REAL* (mirip dengan *double* di Java). Semua tipe lain harus dikonversi sebelum menyimpannya dalam *database*. SQLite sendiri tidak memvalidasi apakah tipe yang ditulis ke kolom sebenarnya sesuai dengan tipe yang sudah didefinisikan, Sebagai contoh, sebuah *integer* dapat ditulis ke dalam kolom *string*.

2.3.8 Normalisasi

Menurut Connolly, dkk. (2015), normalisasi merupakan suatu teknik untuk menghasilkan sekumpulan hubungan dengan *property* yang diinginkan, yang memberikan kebutuhan data terhadap suatu perusahaan. Tujuan dari normalisasi adalah sebagai berikut :

1) Meminimalkan jumlah atribut yang diperlukan untuk mendukung kebutuhan data dari suatu perusahaan.
2) Untuk memperoleh atribut yang bersifat *functional dependencies*.
3) Untuk menghilangkan data yang bersifat *redundancy* pada tiap atribut.

Menurut Connolly, dkk. (2015), terhadap beberapa bentuk normalisasi pada *database* yaitu :

1) *Unnormalized Normal Form* (UNF)

   Unnormalized Normal Form (UNF) merupakan sebuah tabel yang mengandung satu atau lebih *repeating group*.

2) *First Normal Form* (1NF)

   First Normal Form (1NF) merupakan sebuah relasi dimana setiap potongan baris dan kolom mengandung satu dan mungkin hanya satu nilai, dan proses untuk mengubah tabel UNF ke dalam *First Normal Form* (1NF) adalah dengan cara harus diidentifikasi dan menghilangkan bagian yang mengandung *repeating group* pada tabel.

   Ada dua pendekatan untuk menghilangkan perulangan kelompok (*repeating group*) dari tabel yang belum dinormalisasikan, yaitu :

   a. Memasukan data kedalam kolom kosong dari baris yang berisi

perulangan data. Maksudnya adalah mengisi yang kosong dengan duplikat data yang tidak diulang, yang diinginkan. Pendekatan ini umumnya ditunjuk sebgai '*flattening'* pada tabel.

b. Menempatkan perulangan data, sepanjang dengan sebuah salinan atribut kunci yang asli ke dalam sebuah relasi yang terpisah. Terkadang tabel yang belum dinormalisasi mungkin berisi lebih dari satu perulangan kelompok. Pada kasus seperti ini, pendekatan dapat diulang sampai tidak ada lagi perulangan yang terjadi. Sebuah set relasi dapat berada pada 1NF jika tidak terdapat perulangan kelompok (*repeating group*).

3) *Second Normal Form* (2NF)

Second Normal Form (2NF) dapat dihasilkan dengan cara melihat apakah ada atribut yang bukan merupakan *primary key* dapat merupakan fungsi dari sebagian *primary key* (*partial dependence*).

Dalam bentuk normal kedua setiap atribut yang bergantung secara parsial harus dipisahkan. Bentuk normal akan diperoleh bila setiap atribut yang bukan merupakan *primary key* dari suatu tabel secara penuh yang merupakan *functional dependence* dari *primary key* itu.

4) *Third Normal Form* (3NF)

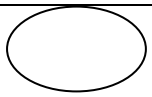
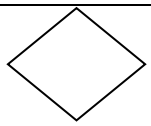
Third Normal Form (3NF) akan secara langsung dilakukan pengujian dengan cara melihat apakah terdapat atribut bukan *key* yang bergantung fungsional terhadap atribut yang bukan *key* yang lain atau disebut (*transitive dependence*). Menggunakan cara yang sama, maka setiap *transitive dependence* harus dipisahkan. *Third Normal Form* (3NF) dapat dikatakan sudah normal apabila anomali yang ada di dalamnya sudah tidak ada, pada kasus tertentu normalisasi dilakukan sampai BCNF.

2.3.9 Entity Relationship Diagram (ERD)

Yakub (2012) mengemukakan bahwa, *Entity Relationship Diagram* (ERD) untuk mendokumentasikan data perusahaan dengan mengidentifikasikan jenis entitas (*entity*) dan hubungannya ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. Rosa &

Shalahuddin (2013) mengemukakan bahwa, *Entity Relationship Diagram* digunakan untuk permodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan *Object Oriented Database Management System* (OODBMS) maka perancangan basis data tidak perlu mengguakan ERD. Adapun simbol-simbol *Entity Relationship Diagram* (ERD) tercantum dalam Tabel 2.1.

Tabel 2.1. Simbol-Simbol *Entity Relationship Diagram* (ERD)

Simbol	Nama	Keterangan
	Entitas	Berupa orang, kejadian, atau benda di mana data akan dikumpulkan.
	Atribut	Merupakan properti dari entitas. Nama atribut harus merupakan kata benda.
	*Relationship*	Menunjukkan hubungan antar 2 entitas. Dideskripsikan dengan kata kerja.
	Link	Sebagai penghubung antara entitas dan *relationship* serta entitas dan atribut.

2.3.10 Data Flow Diagram (DFD)

Ladjamudin (2013) mengemukakan bahwa, diagram aliran data/ *Data Flow Diagram* (DFD) merupakan model dari sistem untuk menggambarkan pembagian sistem ke modul yang lebih kecil. Rosa & Shalahuddin (2013) mengemukakan bahwa, DFD dapat digunakan untuk mempresentasikan sebuah sistem atau perangkat lunak pada beberapa level yang lebih detail untuk merepresentasikan aliran informasi atau fungsi yang lebih detail. Berikut ini adalah tahapan-tahapan perancangan dengan menggunakan DFD :

a. Membuat DFD Level 0 atau sering disebut juga *Context Diagram*.

   DFD Level 0 menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang beriteraksi dengan orang maupun sistem lain. DFD Level 0 digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.

b. Membuat DFD Level 1

DFD Level 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. DFD Level 1 merupakan hasil *breakdown* DFD Level 0 yang sebelumnya sudah dibuat.

c. Membuat DFD Level 2

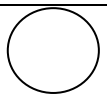
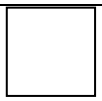
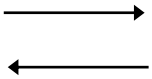
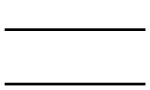
Modul-modul pada DFD Level 1 dapat di-breakdown menjadi DFD Level Modul mana saja yang harus di-*breakdown* lebih detail tergantung pada tingkat kedetailan modul tersebut.

d. Membuat1 DFD Level 3 dan seterusnya

DFD Level 3,4,5 dan seterusnya merupakan *breakdown* dari modul pada DFD Level di-atasnya. *Breakdwon* pada level 3,4,5 dan seterusnya aturannya sama persis dengan DFD Level 1 atau Level 2.

Adapun simbol-simbol yang digunakan dapat dilihat pada Tabel 2.2.

Tabel 2.2. Simbol-Simbol *Data Flow Diagram* (DFD)

Simbol	Keterangan
	Proses atau fungsi atau prosedur, simbol ini digunakan untuk proses pengolahan atau transformasi data.
	*External Entity*, simbol ini digunakan untuk menggambarkan asal atau tujuan data.
	Data Flow, Simbol ini merupakan data yang dikirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukan (*input*) atau keluaran (output).
	Simpanan Data, simbol ini digunakan untuk menggambarkan data flow yang sudah disimpan atau diarsipkan

2.3.11 Flowchart

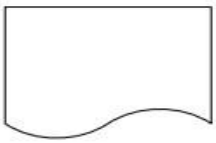
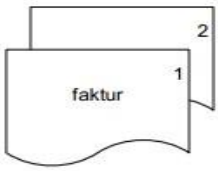
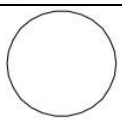
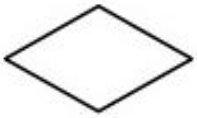
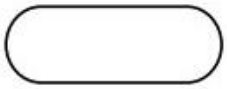
Flowchart atau bagan alir digunakan untuk menjelaskan prosedur jelas dan ringkas terutama untuk alat bantu komunikasi dan untuk dokumentasi pada waktu akan menggambarkan suatu bagan alir dan analisis sistem. Surjawan dan Susanto (2015) menyatakan *flowchart* adalah gambar simbol-simbol yang digunakan

untuk menggambarkan urutan proses atau instruksi-instruksi yang terjadi di dalam suatu program komputer secara sistematis dan logis.

2.3.11.1 Flowchart Dokumen

Kegunaan utama *flowchart* atau bagan alir dokumen adalah untuk menelusuri alur *form* dan laporan sistem dari satu bagian ke bagian lain baik bagaimana alur *form* dan laporan diproses, dicatat dan disimpan. Rizki Ahmad Fauzi (2017) memaparkan bagan alir dokumen merupakan bagan alir yang mengilustrasikan arus dokumen dan informasi di antara bidang tanggung jawab dalam suatu organisasi. Simbol-simbol *flowchart* dokumen ditampilkan Tabel 2.3.

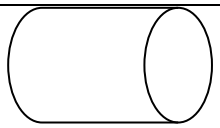
Tabel 2.3. Simbol-Simbol *Flowchart* Dokumen

Simbol	Nama	Fungsi
	Garis Alir *(flowline)*	Menunjukan arah proses pengolahan data
	Dokumen	Menggambarkan *input* dan *output* baik itu dalam proses manual, mekanik atau komputer
	Dokumen dan Tembusannya	Menunjukkan gambaran suatu dokumen asli beserta tembusannya
	*On-Page Connector*	Penghubung bagian lain *flowchart* pada halaman yang sama
	*Off-Page Connector*	Penghubung bagian lain *flowchart* pada halaman yang berbeda
	Kegitan Manual	Menunjukkan suatu kegiatan manual
	Keputusan	Merepresentasikan sebuah keputusan pada *flowchart*.
	*Terminal*	Menandakan awal dan akhir dari *flowchart*

2.3.11.2 Flowchart Sistem

Bagan alir atau *flowchart* sistem merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan alir sistem juga memiliki fungsi untuk menjelaskan setiap urutan dari berbagai prosedur terstruktur yang dilakukan sebuah sistem. Urutan sistem ini harus terdiri dari data dalam sebuah sistem dan proses transformasi data dari sistem tersebut. *Flowchart* sistem hanya bisa berproses baik secara *online* maupun *offline* dan terpisah dari komputer. Bagan alir atau *flowchart* sistem digambarkan dengan menggunakan simbol-simbol yang ditampilkan dalam Tabel 2.4.

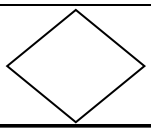
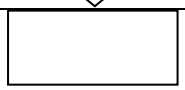
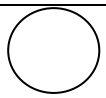
Tabel 2.4. Simbol-Simbol *Flowchart* Sistem

Simbol	Nama	Fungsi
	*Flow*	Merepresentasikan alur jalannya suatu proses.
	Proses	Merepresentasikan proses pada *flowchart*.
	Dokumen	*Input* / *output* dalam format yang dicetak.
	Manul Input	Memasukkan data secara manual *online keyboard*
	*Disk* dan *Online Storage*	Menyatakan *input* yang berasal dari *disk* atau disimpan ke *disk*

2.3.11.3 Flowchart Program

Bagan alir atau *flowchart* program digunakan untuk menjelaskan prosedur jelas dan ringkas mengenai alur algoritma suatu program. Surjawan dan Susanto (2015) mendefinisikan *flowchart* program sebagai simbol-simbol yang menggambarkan proses secara rinci dan detail antara instruksi yang satu dengan instruksi yang lainnya di dalam suatu program komputer yang bersifat secara logik. Simbol-simbol *flowchart* program ditampilkan dalam Tabel 2.3.

Tabel 2.5. Simbol-Simbol *Flowchart* Proses

Simbol	Nama	Fungsi
	*Terminal*	Menandakan awal dan akhir dari *flowchart*.
	*Input* / *Output*	Merepresentasikan *input* atau *output* data atau informasi yang diproses.
	*Flow*	Merepresentasikan alur jalannya suatu proses.
	Keputusan	Merepresentasikan sebuah keputusan pada *flowchart*.
	Proses	Merepresentasikan proses pada *flowchart*.
	*Connector*	Penghubung bagian lain *flowchart* pada halaman yang sama.
	*Offline Connector*	Penghubung bagian lain *flowchart* pada halaman yang berbeda.

2.4 Pengujian

Menurut Pressman (2015) tujuan dari pengujian adalah untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program sebelum menyerahkan program kepada *customer*. Salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas tinggi dalam menemukan kesalahan.

2.4.1 Pengujian Blackbox

Pengujian menggunakan sekumpulan aktifitas validasi, dengan pendekatan *blackbox testing*. Menurut Rosa & Salahuddin (2016), *blackbox testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian *blackbox* dilakukan dengan membuat kasus uji yang

bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai spesifikasi yang dibutuhkan.

Pressman (2010) menambahkan *blackbox testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh *set* kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program.

2.4.2 Pengujian Five View

Sebuah aplikasi dikatakan berkualitas saat banyak pengguna yang menggunakannya, sebaliknya akan dikatakan gagal apabila pengguna meninggalkannya karena beberapa kesalahan yang dimiliki aplikasi tersebut. Kesalahan pertama adalah dengan mengasumsikan bahwa kualitas berarti kebaikan atau kemewahan. Kualitas secara jelas sangat penting untuk dapat mengukur dan mengelola konsep kualitas itu sendiri. Kualitas itu kompleks dan memiliki beragam konsep yang dapat dijelaskan dari lima sudut pandang yang berbeda, di antaranya, *transcendental view*, *user view, manufacturer's view, product view*, dan *value-based view* (Pressman, 2015).

1. *Transcendental View* (Pandangan Transendental)

   Pandangan ini mempertimbangkan kualitas sebaik mungkin atau kondisi yang paling baik. Kualitas yang dapat dikenal atau dirasakan berdasarkan pengalaman atau pernah menggunakan produk yang sejenis tetapi sulit didefinisikan dan dioperasionalkan.

2. *User View* (Pandangan Pengguna)

   Pandangan ini mengganggap bahwa kualitas harus sesuai dengan maksud dan tujuan. Sebuah produk memiliki kualitas yang baik jika produk tersebut memuaskan sejumlah besar dari pengguna. Elemen yang berkaitan dalam poin ini adalah *usability, reliability,* dan *efficiency*.

3. *Manufacturing View* (Pandangan Manufaktur)

   Pengertian kualitas disini adalah mengerti dan menyesuaikan diri dengan spesifikasi tertentu. Tingkat kualitas dari sebuah produk ditentukan oleh seberapa luas produk tersebut telah memenuhi spesifikasi yang ditentukan.

4. *Product View* (Pandangan Produk)

   Berdasarkan padangan ini, kualitas dilihat sebagai ikatan terhadap karakteristik yang melekat dari sebuah produk. Karakteristik yang melekat dari sebuah produk tersebut terdiri dari kualitas internal, dan kualitas eksternal.

5. *Value-based View* (Pandangan Berbasis Nilai)

   Dalam pandangan ini, kualitas tergantung pada jumlah pengguna yang akan membayar sejumlah uang untuk aplikasi tersebut. Kualitas tidak dapat dipahami jika aplikasi tidak membuat nilai ekonomis. Pandangan *value-based* membuat *trade-off* antara harga dan kualitas.