BAB II
KAJIAN LITERATUR

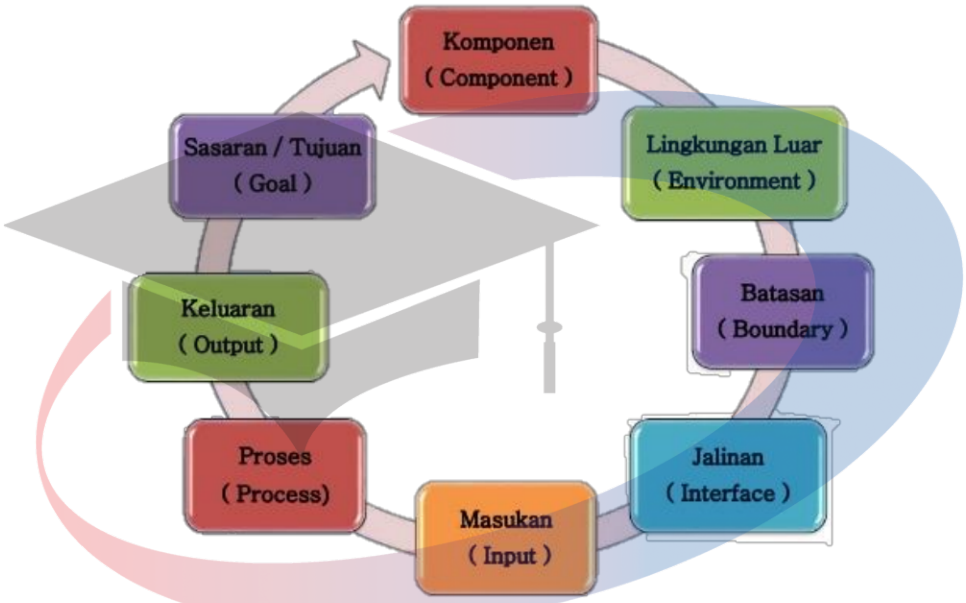
2.1 Konsep Sistem Informasi

2.1.1 Sistem

Sistem adalah gabungan dari sekumpulan elemen, komponen atau variabel yang dihubungkan untuk mencapai suatu tujuan tertentu [6]. Sistem adalah kumpulan orang, bahan, fasilitas, dan peralatan yang bekerja sama menurut aturan-aturan yang sistematis dan terstruktur untuk mengubah masukan (*input*) menjadi keluaran (*output*) yang penting dan diperlukan, serta membentuk satu kesatuan yang menjalankan suatu fungsi untuk mencapai suatu tujuan [7], [8], [9]. Sistem juga bisa diartikan sebagai sebuah gabungan atau kesatuan data yang terhubung dan terorganisir secara prosedural untuk mencapai suatu sasaran [10]. Sistem yang baik mempunyai sejumlah karakteristik antara lain [11], [12]:

1. Komponen sistem (*Component*) yang dimiliki suatu sistem saling berinteraksi dan bekerja sama membentuk satu kesatuan yang terdiri dari berbagai cabang sistem.
2. Batasan sistem (*Boundary*) merupakan sebuah batasan antara sistem dengan sistem lain atau lingkup di luar sistem.
3. Lingkungan luar sistem (*Environment*) merupakan hal-hal di luar sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat meguntungkan dan merugikan. Apabila lingkungan luar berifat menguntungkan, maka hal itu harus dijaga. Namun, yang sifatnya merugikan harus dikendalikan.
4. Penghubung sistem (*Interface*) merupakan media yang menghubungkan satu subsistem dengan subsistem lain. Penghubung sistem ini memungkinkan sumber daya mengalir dari satu subsistem ke subsistem lain. Keluaran dari subsistem ini akan menjadi masukan untuk subsistem lain melalui alat penghubung ini
5. Masukan sistem (*Input*) adalah sumber daya seperti masukan pemeliharaan (*maintenance input*) dan masukan sinyal (*signal input*) yang dimasukkan ke dalam sistem. Maintenance adalah sumber daya yang dimasukkan agar sistem dapat beroperasi, sedangkan *signal input* adalah sumber daya yang diproses untuk menghasilkan keluaran.
6. Keluaran sistem (*Output*) adalah hasil dari pemrosesan masukan dan telah diklasifikasikan sebagai keluaran yang berguna dan yang akan dibuang. Keluaran ini merupakan masukan bagi subsistem lain.

7. Pengolahan sistem (*Process*) merupakan suatu proses yang mengubah masukan menjadi keluaran. Suatu sistem dapat memiliki bagian pengolah atau sistem itu sendiri sebagai pengolahnya.
8. Sasaran sistem (*Objective*) merupakan tujuan yang ingin dicapai oleh sistem. Apabila sistem tidak memiliki sasaran, maka operasi sistem tidak akan berguna. Sasaran sistem sangat menentukan masukan yang dibutuhkan dan keluaran yang dihasilkan oleh sistem. Sistem dianggap berhasil jika sudah mencapai sasaran atau tujuannya



Gambar 2. 1 Karakteristik Sistem

2.1.2 Informasi

Informasi adalah sekumpulan data yang diambil dan dikelola untuk memberikan suatu arti yang akan mempengaruhi proses pengambilan keputusan [10]. Informasi juga bisa dikatakan sebagai data yang diolah menjadi bentuk yang lebih berguna dan informatif sehingga dapat bermanfaat bagi penerimanya [13]. Menurut Gordon B. Davis, informasi adalah data yang telah diolah menjadi bentuk yang penting bagi penerima dan memiliki nilai nyata atau yang dapat dirasakan dalam pengambilan keputusan saat ini atau yang akan datang [14]. Informasi memiliki beberapa fungsi, yaitu sebagai sumber pengetahuan baru, menghapus ketidakpastian, sebagai media hiburan, sumber berita, sosialisasi kebijakan, mempengaruhi khalayak, dan menyatukan pendapat. Informasi yang berkualitas memiliki ciri-ciri sebagai berikut [12]:

1. Akurat, artinya informasi tersebut mencerminkan keadaan sebenarnya.
2. Tepat waktu, artinya informasi harus tersedia segera bila dibutuhkan.
3. Relevan, artinya informasi yang diberikan harus sesuai dengan kebutuhan.

4. Lengkap, artinya informasi harus utuh.

Informasi dapat mengurangi ketidakpastian dan memiliki nilai dalam proses pengambilan keputusan karena melalui informasi, kita dapat memilih tindakan dengan risiko yang paling minimal. Untuk mencapai kebijaksanaan dan pengambilan keputusan yang efektif, diperlukan pengolahan data menjadi informasi yang relevan dengan masalah yang dihadapi oleh perusahaan. Oleh karena itu, data dapat dianggap sebagai bahan mentah yang perlu diproses terlebih dahulu sebelum dapat digunakan dengan maksimal [12].

2.1.3 Sistem Informasi

Sistem informasi merupakan suatu struktur di dalam suatu organisasi yang mengakomodasi kebutuhan pengolahan transaksi sehari-hari, yang secara khusus mendukung fungsi manajerial dalam strategi organisasi. Sistem ini dirancang untuk menyediakan laporan yang dibutuhkan oleh pihak eksternal, dan dapat dijelaskan sebagai kombinasi orang, fasilitas, teknologi, media, prosedur, dan pengendalian. Tujuan utama sistem informasi adalah untuk memfasilitasi komunikasi yang vital, memproses transaksi rutin, memberikan informasi kepada manajemen tentang peristiwa penting internal dan eksternal, serta menyediakan dasar informasi untuk mendukung pengambilan keputusan yang efektif [10].

Sistem informasi melibatkan tiga kegiatan pokok di dalamnya, yakni kegiatan *input* (masukan), proses (pemrosesan), dan *output* (keluaran). Ketiga kegiatan ini bertujuan untuk menghasilkan informasi yang diperlukan oleh organisasi, baik untuk pengambilan keputusan, pengendalian operasi, analisis masalah, maupun pengembangan produk atau jasa baru. Peran masukan adalah dalam pengumpulan bahan mentah atau data mentah, baik yang berasal dari internal organisasi maupun lingkungan sekitarnya. Pemrosesan berfungsi untuk mengubah bahan mentah menjadi bentuk yang lebih bermakna. Sementara itu, keluaran bertujuan untuk mentransfer informasi yang telah diproses kepada pihak atau aktivitas yang membutuhkannya. Sistem informasi juga memerlukan umpan balik (*feedback*) sebagai dasar evaluasi dan perbaikan pada tahap input berikutnya [15].

2.2 Sistem Persetujuan

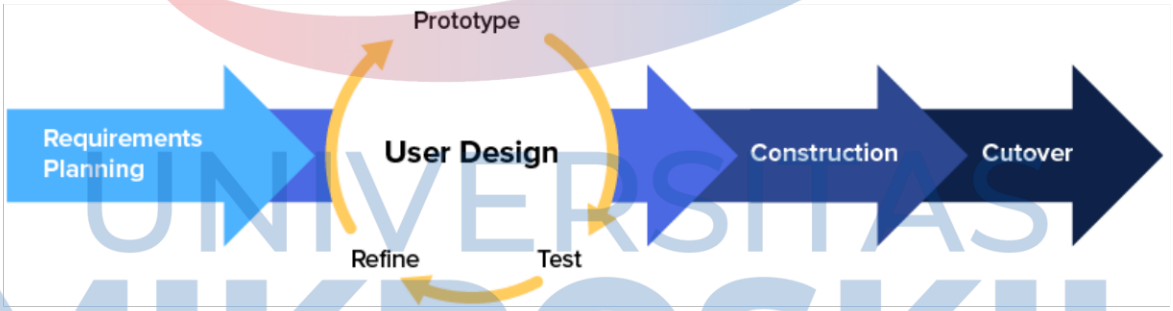
Persetujuan, yang juga dikenal sebagai *approval*, merujuk pada suatu pernyataan yang menunjukkan kesepakatan atau persetujuan. Persetujuan juga dapat diartikan sebagai pembenaran atau validasi. Dalam konteks ini, *approval* menjadi bagian dari *workflow* atau alur kerja, suatu proses bisnis di mana informasi atau dokumen yang telah dibuat dialirkan

dari satu pihak ke pihak lainnya untuk memperoleh klarifikasi lebih lanjut sesuai dengan ketentuan dan prosedur yang telah disepakati oleh perusahaan sebelumnya. Penerapan aplikasi *approval request* memiliki tujuan untuk mengurangi potensi kesalahan dalam proses pengajuan persetujuan, memudahkan karyawan dalam mengajukan permohonan, dan mempercepat proses pemeriksaan serta validasi pengajuan tanpa memerlukan pertemuan langsung dengan pihak terkait [16].

2.3 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah kumpulan metodologi yang muncul sebagai respons terhadap kelemahan metode pengembangan *waterfall* dan variasinya. RAD menggabungkan teknik dan alat komputer khusus untuk mempercepat tahap analisis, desain, dan implementasi, sehingga sebagian sistem dapat dikembangkan dengan cepat dan tersedia untuk evaluasi dan umpan balik pengguna [17]. RAD adalah salah satu model pengembangan sistem yang menawarkan fleksibilitas tinggi dan dirancang untuk mengakomodasi perubahan yang mungkin terjadi dalam kebutuhan pembuatan sistem [18].

2.3.1 Tahapan Metode RAD



Gambar 2. 2 Tahapan Metode *Rapid Application Development*

Rapid Application Development (RAD) terdiri dari empat fase utama, yaitu perencanaan kebutuhan, desain pengguna, konstruksi, dan implementasi, seperti yang ditunjukkan pada Gambar 2.2. Berikut adalah uraian singkat tentang setiap tahap dalam metode RAD [18], [19]:

1. Fase Perencanaan Kebutuhan (*Requirements Planning*) juga dikenal sebagai tahap analisis, di mana analis berinteraksi dengan pengguna dan para *stakeholder* untuk mengidentifikasi tujuan sistem dan kebutuhan sistem berdasarkan proses bisnis yang ada untuk mencapai tujuan yang diharapkan. Fase ini adalah tahap paling penting karena melibatkan kedua belah pihak.

2. Fase Desain Pengguna (*User Design*) juga disebut sebagai tahap desain. Pada tahap ini aktivitas pengguna yang terlibat menentukan tercapai atau tidaknya tujuan. Hal ini dikarenakan pada fase ini akan menjalankan proses desain dan melakukan perbaikan jika terdapat ketidaksesuaian desain antara pengguna dan analis. Pengguna dapat langsung memberikan komentar jika terdapat ketidaksesuaian dalam rancangan dan perancangan sistem akan mengacu pada dokumen kebutuhan pengguna yang dibuat pada tahap sebelumnya.
3. Fase Konstruksi (*Construction*) merupakan tahapan di mana sistem akan dibangun sesuai dengan rancangan sistem yang sudah disetujui oleh pengguna dan analis pada tahap sebelumnya. Sebelum diterapkan pada suatu organisasi akan dilakukan pengujian terhadap sistem terlebih dahulu untuk memastikan apakah ada kesalahan atau tidak. Biasanya pada tahap ini pengguna akan memberikan umpan balik terhadap sistem yang sudah dibangun dan mendapat persetujuan mengenai sistem tersebut.
4. Fase Implementasi (*Implementation*) juga dapat disebut sebagai tahap *cutover*, tahap akhir dalam pengembangan sistem *Rapid Application Development* (RAD). Sistem yang telah dibangun dan diperkenalkan kepada pengguna. Spesifikasi *server*, perangkat lunak yang dibutuhkan untuk menjalankan sistem, pengujian, peralihan sistem, dan pelatihan pengguna akan dijelaskan pada tahap ini.

2.3.2 Kelebihan dan Kekurangan Metode RAD

Rapid Application Development (RAD) memungkinkan pengembang membuat prototipe perangkat lunak dengan cepat, mempercepat siklus pengembangan, dan meningkatkan keterlibatan para pemangku kepentingan. Namun seperti metode pengembangan lainnya, *Rapid Application Development* (RAD) juga memiliki beberapa kelebihan dan kekurangan yang perlu diperhatikan sebagai berikut [20], [21].

Tabel 2. 1 Kelebihan dan Kekurangan Metode RAD

Kelebihan	Kekurangan
Lebih efektif dibandingkan metode pengembangan *Waterfall / Sequential linear* dalam pembuatan sistem yang memenuhi kebutuhan langsung pengguna.	Metode RAD tidak akan berhasil jika pengembang dan pelanggan tidak terlibat dalam aktivitas *rapid-fire* yang diperlukan untuk melengkapi sistem dalam waktu yang sangat singkat.
Ideal untuk proyek dengan waktu singkat. Tim pengembangan dapat membuat prototipe atau versi awal aplikasi dengan cepat menggunakan RAD. Ini memungkinkan untuk merespon perubahan kebutuhan pelanggan atau kebutuhan pasar.	Pendekatan RAD cenderung tidak cocok untuk proyek yang besar atau kompleks karena fokus pada pengembangan cepat mungkin mengorbankan elemen seperti keamanan, skalabilitas, dan kinerja. Selain itu, jika sistem tidak dapat dimodulkan dengan teratur, pembangunan komponen penting pada RAD akan menjadi sangat sulit.
Model RAD memiliki kemampuan untuk menggunakan kembali komponen yang sudah ada sehingga pengembang tidak perlu membuatnya dari awal. Ini mengurangi waktu pengembangan dan meningkatkan efisiensi. Pendekatan iteratif dapat mengurangi risiko proyek secara keseluruhan karena setiap iterasi memberikan kesempatan untuk mengevaluasi dan mengubah rencana serta kebutuhan.	RAD tidak cocok digunakan untuk sistem yang mempunyai resiko teknik yang tinggi.
RAD memberi pengembang lebih banyak waktu untuk berkonsentrasi pada kualitas perangkat lunak karena memungkinkan mereka untuk menemukan dan mengatasi masalah pada tahap awal siklus pengembangan.	Membutuhkan tim pengembang yang sangat terampil dan berpengalaman untuk mengimplementasikan RAD dengan efektif.

2.4 Alat Bantu Pengembangan Sistem

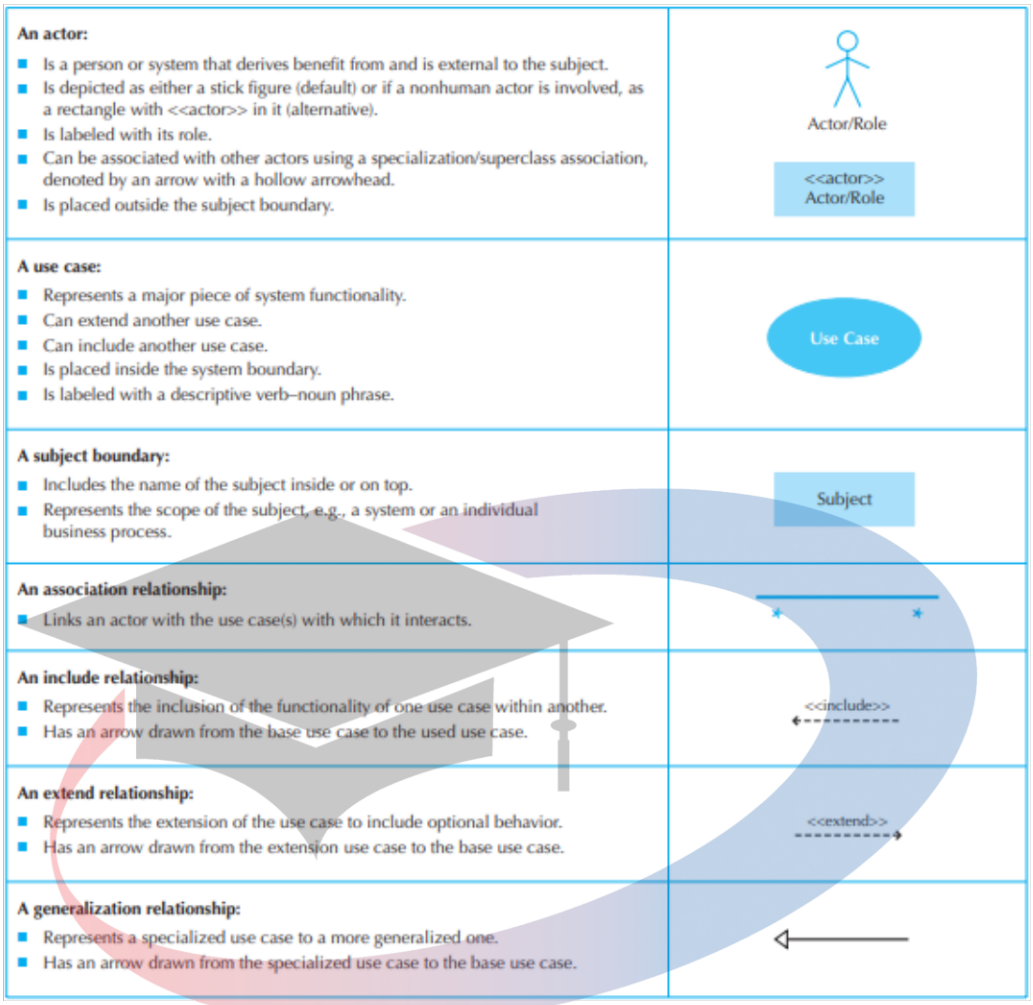
2.4.1 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan sebuah bahasa standar yang digunakan dalam rekayasa perangkat lunak untuk mendokumentasikan, mendesain, dan memodelkan sistem perangkat lunak berbasis objek. UML terdiri dari beberapa jenis diagram yang berbeda, masing-masing berfokus pada aspek yang berbeda dari sistem perangkat lunak yang dimodelkan. Berikut adalah beberapa diagram UML yang banyak digunakan [22]:

1. Diagram *use case* (*Use Case Diagram*)
2. Diagram aktivitas (*Activity Diagram*)
3. Diagram kelas (*Class Diagram*)
4. Diagram sekuensi (*Sequence Diagram*)

2.4.1.1 *Use Case Diagram*

Use Case Diagram merupakan diagram yang menggambarkan hubungan antara sistem dengan sistem eksternal dan pengguna, mencakup identifikasi siapa yang menggunakan sistem serta cara pengguna berinteraksi dengan sistem tersebut [22]. Diagram *use case* adalah diagram fungsional yang menggambarkan sesuatu yang harus atau dapat dilakukan oleh sistem dalam berbagai situasi saat merespon permintaan aktor utama. Diagram ini biasanya digunakan untuk menjelaskan fungsi setiap sistem dan mengetahui hak akses untuk menggunakannya [23].

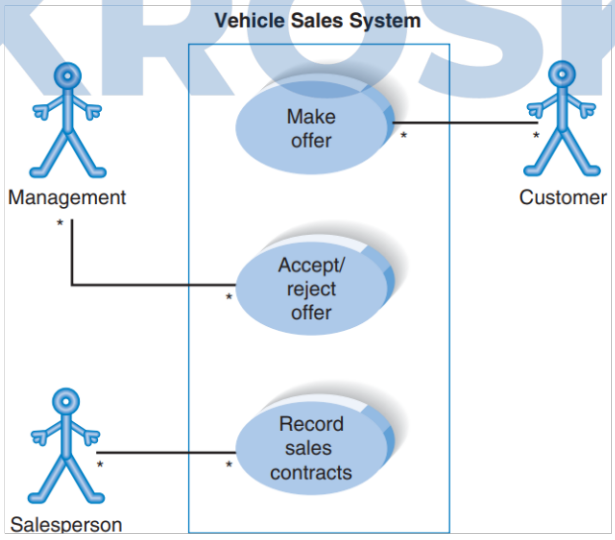


Gambar 2. 3 Elemen pada *Use Case Diagram*

Use Case Diagram memiliki tujuh elemen yang dapat digunakan untuk menunjukkan alur proses aplikasi yang dirancang, yaitu [23], [24]:

1. *Actor* merupakan seseorang atau sistem eksternal yang berinteraksi dan memperoleh nilai dari sistem. Aktor bukanlah individu, namun peran yang dapat dimainkan pengguna saat berinteraksi dengan sistem. Satu individu tertentu dapat memainkan banyak peran secara bersamaan. Aktor adalah sistem eksternal yang mengawali *use case*. Aktor biasanya digambarkan sebagai gambar *stick figure* atau, apabila melibatkan aktor non-manusia, maka digambarkan sebagai gambar persegi panjang dengan tulisan <<*actor*>> di dalamnya. Aktor diberi nama sesuai dengan perannya dan dapat berhubungan dengan aktor lain melalui hubungan spesialisasi/*superclass* yang ditandai dengan panah berujung kosong. Aktor berada di luar batas sistem.
2. *Use Case* direpresentasikan seperti elips, yang merupakan proses yang signifikan yang akan dilakukan oleh sistem yang memberikan manfaat bagi aktor dalam beberapa hal. Setiap *use case* mewakili bagian utama dari satu fungsi sistem. *Use case* diberi label dengan frasa kata kerja deskriptif.

3. *Subject Boundary* di mana *use case* diapit dalam batas sistem, yaitu kotak yang mewakili sistem dan dengan jelas menggambarkan bagian mana dari diagram yang berada di luar atau di dalam sistem. Nama sistem dapat muncul di dalam atau di atas kotak.
4. *Association Relationship* biasanya mewakili komunikasi dua arah antara *use case* dan aktor. Hubungan ini menunjukkan *use case* mana yang berinteraksi dengan para aktor. Jika komunikasi hanya bersifat satu arah, maka kepala panah padat dapat digunakan untuk menunjukkan arah aliran informasi.
5. *Include Relationship* merupakan jenis hubungan lain antara *use case* yang muncul ketika satu *use case* menggunakan *use case* lain. *Include relationship* digambarkan sebagai panah garis putus-putus yang mengarah ke *use case* yang sedang digunakan dan diberi label <<*include*>>. Panah garis putus-putus tidak menunjukkan jenis data atau aliran proses apa pun di antara *use case*. *Include relationship* menyiratkan bahwa *use case* tempat panah berasal menggunakan *use case* tempat panah berakhir saat dieksekusi. Biasanya, *use case* yang "*included*" mewakili fungsi generik yang umum untuk banyak fungsi bisnis. Fungsionalitas tersebut diperhitungkan ke dalam *use case* terpisah yang kemudian dapat digunakan oleh *use case* lain.
6. *Extend Relationship* memperluas *use case* dengan menambahkan perilaku atau tindakan baru. Ini digambarkan sebagai panah putus-putus yang menunjuk ke *use case* yang diperluas dan diberi label dengan simbol <<*extend*>>. Panah garis putus-putus tidak menunjukkan jenis data atau aliran proses apa pun di antara *use case.*
7. *Generalization relationship* mewakili *use case* khusus ke *use case* yang lebih umum. Hubungan ini memiliki panah yang ditarik dari *use case* khusus ke *use case* dasar.

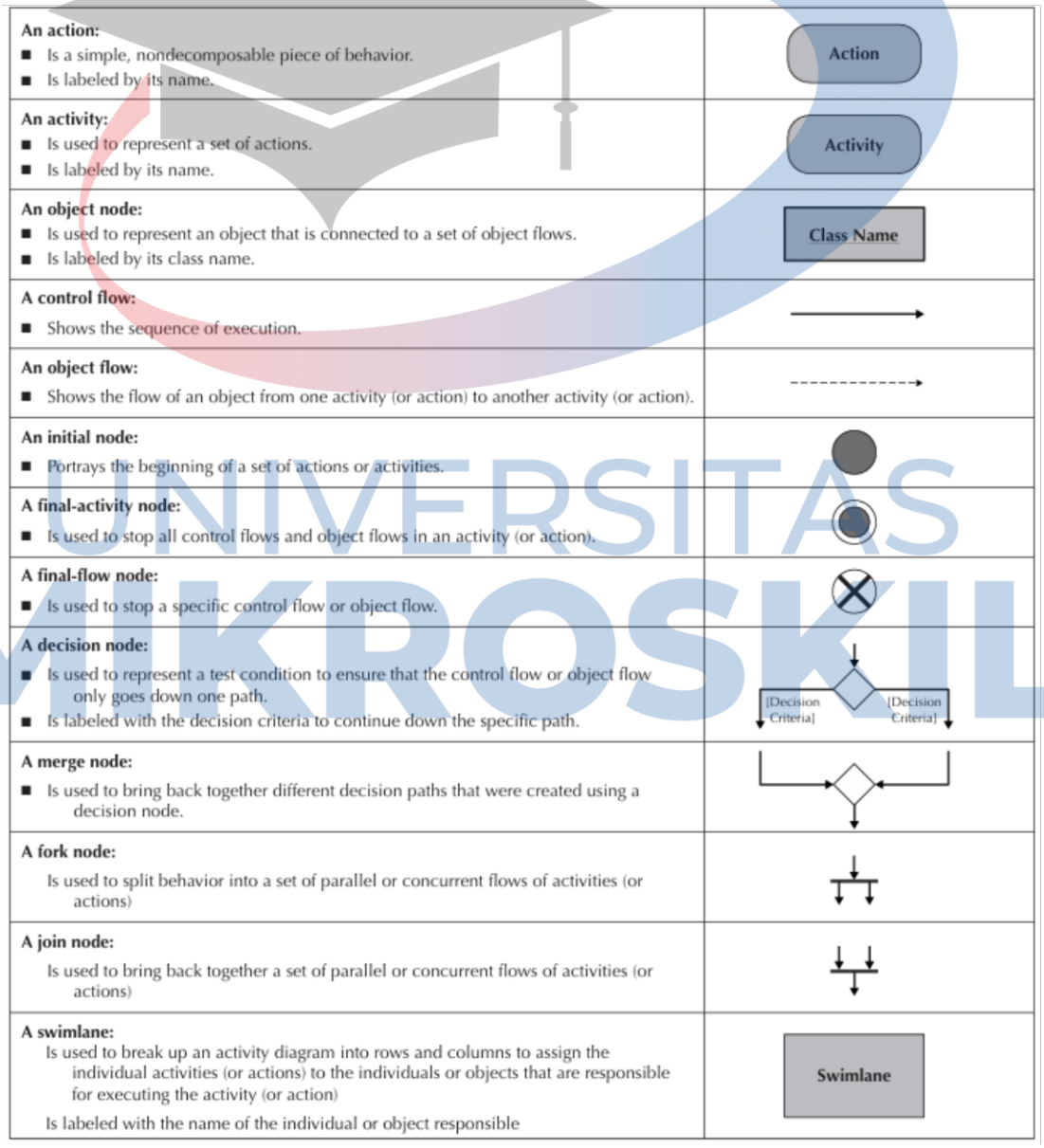


Gambar 2. 4 Contoh *Use Case Diagram*

Gambar 2.4 merupakan gambar diagram *use case* untuk sistem penjualan kendaraan di mana aktornya adalah pelanggan, personel manajemen, dan tenaga penjualan. Dapat dilihat dari diagram bahwa masing-masing aktor akan menggunakan sistem penjualan dengan *use case*, seperti membuat penawaran, menerima/menolak penawaran, dan mencatat kontrak penjualan [23].

2.4.1.2 *Activity Diagram*

Activity diagram merupakan diagram yang menggambarkan alur kerja atau proses bisnis dalam sistem, langkah-langkah dalam sebuah *use case*, serta metode dari sebuah objek [22]. Diagram aktivitas menggambarkan aktivitas utama dan hubungan antar aktivitas dalam suatu proses [24]. *Activity diagram* digambarkan dengan sebuah alur secara terstruktur [25].



Gambar 2. 5 Elemen pada *Activity Diagram*

Elemen-elemen utama yang terdapat dalam sebuah diagram aktivitas, adalah sebagai berikut [24]:

1. Tindakan dan aktivitas (*Action and Activities*) menggambarkan tindakan atau perilaku yang dilakukan dalam proses. Dalam diagram aktivitas elemen ini digambarkan sebagai persegi Panjang bulat. Action dan activities dapat mewakili perilaku manual maupun terkomputerisasi. Kedua elemen memiliki nama yang dimulai dengan kata kerja dan diakhiri dengan kata benda. Satu-satunya perbedaan antara tindakan dan aktivitas adalah bahwa aktivitas dapat diuraikan menjadi serangkaian aktivitas dan/atau perilaku, sedangkan tindakan mewakili bagian sederhana yang tidak dapat diurai dari keseluruhan perilaku yang dapat diurai. Biasanya, hanya aktivitas yang digunakan untuk memodelkan proses bisnis atau alur kerja.
2. Node objek (*Object nodes*) digunakan untuk mewakili suatu objek yang terhubung ke sekumpulan aliran objek. Aktivitas dan tindakan sering kali memodifikasi atau mengubah objek. Node objek memodelkan objek-objek ini dalam diagram aktivitas. Dalam diagram aktivitas, node objek direpresentasikan sebagai persegi Panjang.
3. Aliran Kontrol dan Aliran Objek (*Control Flow* dan *Object Flow*) merupakan dua jenis aliran berbeda dalam diagram aktivitas. Aliran kontrol memodelkan aliran yang diambil melalui proses bisnis. Aliran kontrol digambarkan sebagai garis padat dengan mata panah yang menunjukkan arah aliran. Aliran kontrol hanya dapat digunakan pada *action* atau *activities*. Sedangkan, Aliran objek memodelkan aliran objek melalui proses bisnis. Karena *action* dan *activities* memodifikasi atau mengubah objek, aliran objek diperlukan untuk menunjukkan objek sebenarnya yang mengalir masuk dan keluar dari *action* atau *activities*. Aliran objek digambarkan sebagai garis putus-putus dengan mata panah yang menunjukkan arah aliran. Sebuah Aliran objek harus dikaitkan ke suatu *action* atau *activities* di satu ujung dan *object node* di ujung lainnya.
4. Node kontrol (*Control Node*) dalam diagram aktivitas dibagi menjadi tujuh jenis, yaitu *initial, final-activity, final-flow, decision, merge, fork,* dan *join.*
    a. *Initial Node* atau node awal menggambarkan awal dari serangkaian *action* atau *activities.* Initial node digambarkan sebagai lingkaran kecil berisi.
    b. *Final-activity node* digunakan untuk menyelesaikan proses yang sedang dimodelkan. Setiap *final-activity node* tercapai, semua action dan activities akan segera berakhir, terlepas dari apakah aktivitas tersebut selesai atau tidak. *Final-activity node* digambarkan sebagai lingkaran yang mengelilingi lingkaran kecil berisi, sehingga tampak seperti target.

c. *Final-flow node* mirip dengan *Final-activity node,* hanya saja node tersebut menghentikan jalur eksekusi tertentu dalam proses bisnis namun mengizinkan jalur paralel atau jalur bersamaan lainnya untuk melanjutkan. *Final-flow node* digambarkan sebagai lingkaran kecil dengan tanda X di dalamnya.

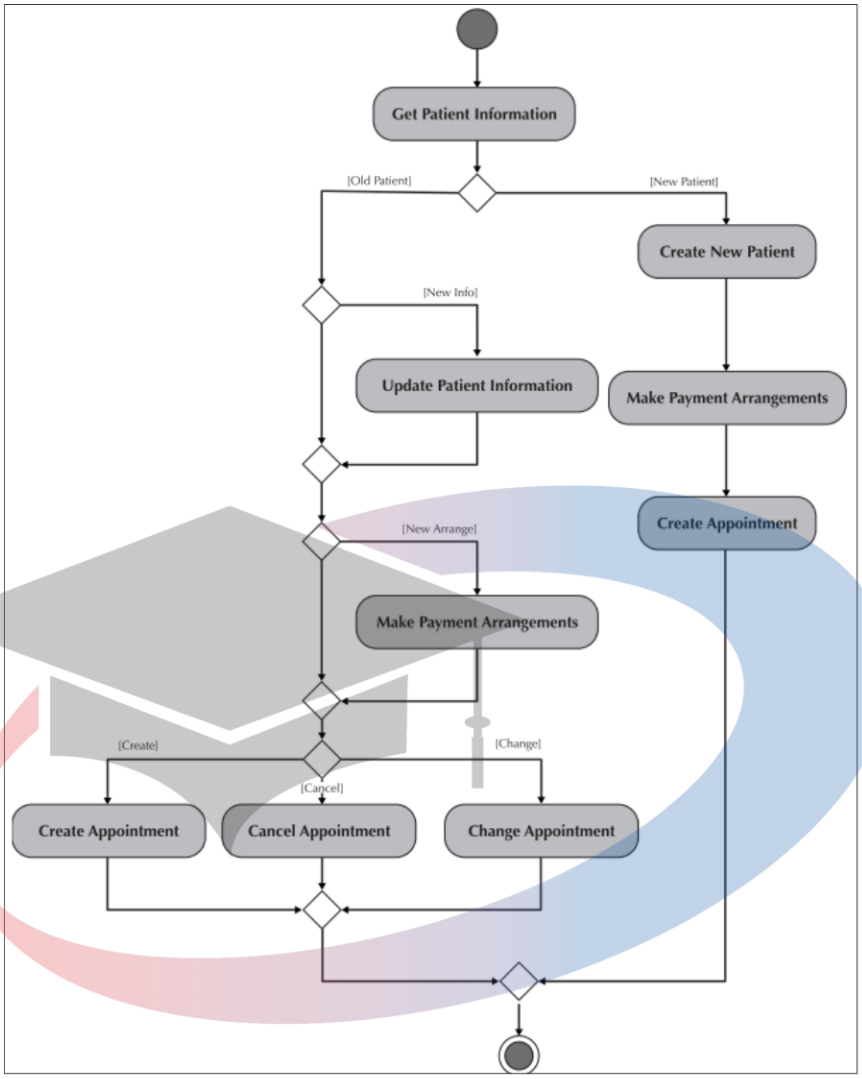
d. *Decision Node* digunakan untuk mewakili kondisi pengujian aktual yang menentukan jalur keluar dari *decision node* mana yang akan diambil. Setiap jalur keluaran harus diberi label kondisi proteksi.

e. *Merge node* digunakan untuk menggabungkan beberapa jalur yang saling eksklusif yang telah dipisahkan berdasarkan keputusan sebelumnya. Namun agar lebih jelas, lebih baik tidak menggunakan *merge node*.

f. *Fork node* digunakan untuk membagi perilaku proses bisnis menjadi beberapa aliran pararel atau bersamaan.

g. *Join node* digunakan untuk menyatukan kembali aliran pararel atau bersamaan yang terpisah dalam proses bisnis menjadi satu aliran.

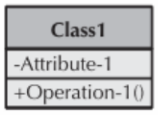
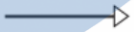
5. *Swimlanes* digunakan untuk memecah diagram aktivitas menjadi baris dan kolom untuk menetapkan aktivitas individu (atau *actions*) kepada individu atau objek yang bertanggung jawab untuk melaksanakan aktivitas (atau tindakan).



Gambar 2. 6 Contoh *Activity Diagram*

2.4.1.3 *Class Diagram*

Class diagram merupakan diagram yang menggambarkan struktur objek dari perangkat lunak dengan menunjukkan kelas-kelas, atribut-atribut, dan hubungan antara kelas-kelas tersebut [22]. Diagram kelas adalah model statis yang menunjukkan kelas-kelas dan hubungan antar kelas yang tetap konstan dalam sistem sepanjang waktu. Diagram kelas menggambarkan kelas-kelas, yang mencakup perilaku dan keadaan, dengan hubungan antar kelas [26].

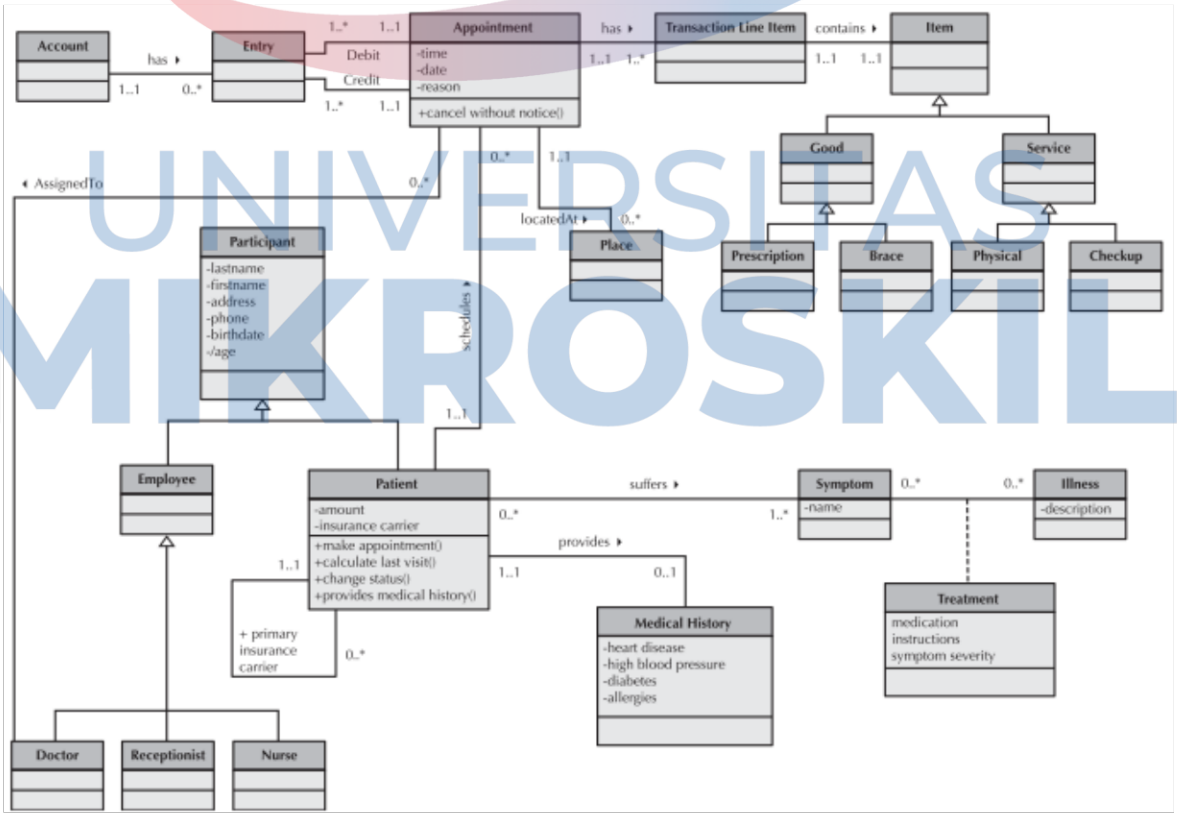
A class: • Represents a kind of person, place, or thing about which the system will need to capture and store information. • Has a name typed in bold and centered in its top compartment. • Has a list of attributes in its middle compartment. • Has a list of operations in its bottom compartment. • Does not explicitly show operations that are available to all classes.	
An attribute: • Represents properties that describe the state of an object. • Can be derived from other attributes, shown by placing a slash before the attribute's name.	attribute name /derived attribute name
An operation: • Represents the actions or functions that a class can perform. • Can be classified as a constructor, query, or update operation. • Includes parentheses that may contain parameters or information needed to perform the operation.	operation name ()
An association: • Represents a relationship between multiple classes or a class and itself. • Is labeled using a verb phrase or a role name, whichever better represents the relationship. • Can exist between one or more classes. • Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance.	
A generalization: • Represents a-kind-of relationship between multiple classes.	
An aggregation: • Represents a logical a-part-of relationship between multiple classes or a class and itself. • Is a special form of an association.	
A composition: • Represents a physical a-part-of relationship between multiple classes or a class and itself • Is a special form of an association.	

Gambar 2. 7 Elemen pada *Class Diagram*

Elemen-elemen utama yang terdapat dalam sebuah diagram kelas, adalah sebagai berikut [26]:

1. Kelas (*Class*) merupakan elemen utama dalam diagram kelas yang merepresentasikan entitas atau objek seperti jenis orang, tempat, atau benda yang dibutuhkan sistem untuk menangkap dan menyimpan informasi. Kelas Digambar menggunakan persegi Panjang tiga bagian, dengan nama kelas di atas, atribut di tengah, dan operasi di bawah.
2. Atribut (*Attribute*) merupakan variable yang terkeait dengan kelas, yang mendeskripsikan keadaan atau property dari objek kelas tersebut. Atribut dapat memiliki tipe data interger, string, boolean, atau tipe data objek lainnya. Atribut dapat diturunkan dari atribut lain, dan ditunjukkan dengan memberi tanda garis miring di depan nama atribut.

3. Operasi (*Operation*) merupakan tindakan atau fungsi yang dapat di lakukan suatu kelas. Operasi dapat diklasifikasikan sebagai konstruktor, kueri, atau pembaruan operasi. Fungsi-fungsi yang tersedia untuk semua kelas (misalnya, membuat *instance* baru, mengembalikan nilai untuk atribut tertentu, menetapkan nilai untuk atribut tertentu, menghapus sebuah *instance*) tidak secara eksplisit ditampilkan dalam persegi panjang kelas.
4. Relasi (*Relationship*) menggambarkan hubungan antara kelas-kelas dalam sistem. Beberapa jenis relasi antara kelas, yaitu:
   a. Asosiasi (A*ssociation*) merupakan hubungan di mana objek dari satu kelas merupakan bagian dari objek lain, tetapi objek tersebut dapat berdiri sendiri dan diberi label menggunakan frasa kata kerja atau nama peran, atau mana saja yang lebih mewakili hubungan tersebut. Asosiasi dapat berada di antara satu atau lebih kelas.
   b. Generalisasi (*Generalization*), mewakili semacam hubungan antara beberapa kelas.
   c. Agregasi (*Aggregation*) merupakan bagian logis dari hubungan antara beberapa kelas atau satu kelas dan kelas itu sendiri.
   d. Komposisi (*Composition*) merupakan bagian fisik dari hubungan antara beberapa kelas atau satu kelas dan kelas itu sendiri dan tidak dapat ada secara independen.



Gambar 2. 8 Contoh *Class Diagram*

2.4.1.4 *Sequence Diagram*

Sequence diagram merupakan representasi interaksi antar objek yang digunakan untuk menunjukkan komunikasi atau pesan yang ada antar objek tersebut [27]. *Sequence diagram* adalah model dinamis yang menunjukkan urutan pesan secara eksplisit yang diteruskan antar objek dalam interaksi tertentu. Karena *Sequence diagram* fokus pada urutan aktivitas yang terjadi antara sekumpulan objek sepanjang waktu, diagram ini berguna untuk memahami spesifikasi waktu nyata dan kasus penggunaan yang kompleks [28].

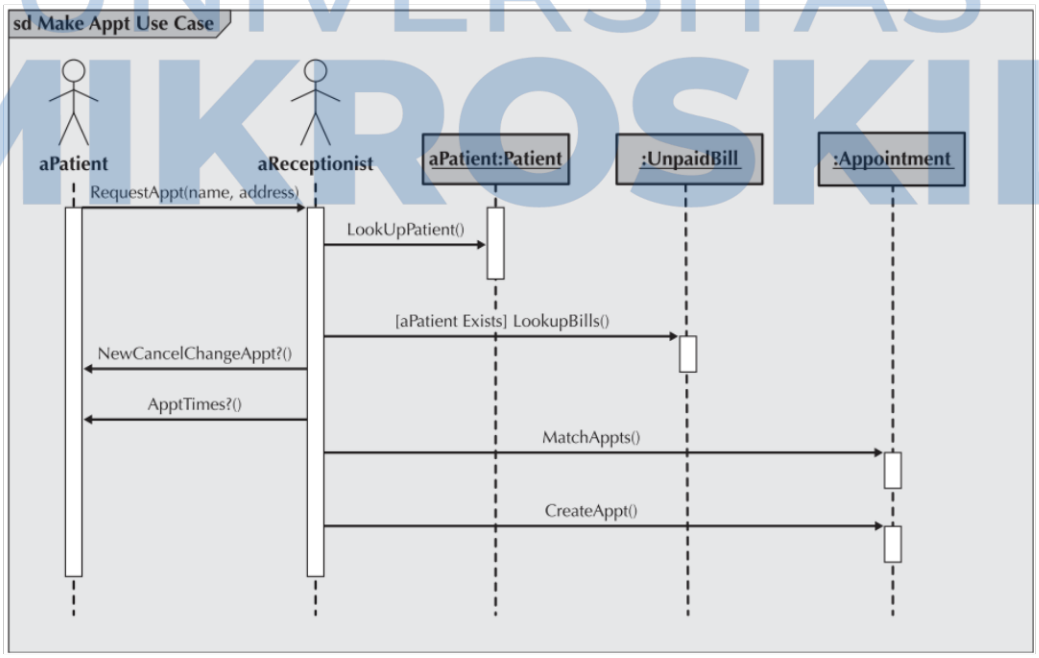
Gambar 2. 9 Elemen pada *Sequence Diagram*

Elemen-elemen utama yang terdapat dalam sebuah diagram kelas, adalah sebagai berikut [28]:

1. *Actor* adalah seseorang atau sistem yang memperoleh manfaat dari sistem dan di luar sistem. Aktor dapat berupa pengguna, sistem lain, atau perangkat keras. Aktor yang

berpartisipasi dalam urutan ditempatkan di bagian atas diagram menggunakan simbol aktor dari diagram *use case* / figur tongkat (*default*). Jika aktor bukan manusia, maka akan digambarkan sebagai persegi panjang dengan <<actor>> di dalamnya (alternatif).

2. *Object* berpartisipasi secara berurutan dengan mengirim dan/atau menerima pesan. Objek yang berpartisipasi dalam urutan ditempatkan di bagian atas diagram menggunakan simbol objek pada diagram objek.
3. *Lifeline* mewakili keberadaan objek selama interaksi. *Lifeline* berisi tanda X pada titik di mana kelas tidak lagi berinteraksi. Elemen ini digambarkan dengan garis putus-putus.
4. *Execution Occurrence* adalah persegi panjang sempit yang ditempatkan di atas garis hidup untuk menunjukkan kapan kelas mengirim dan menerima pesan.
5. Pesan (*message*) adalah komunikasi antar objek yang menyampaikan informasi dengan harapan akan terjadi aktivitas. Banyak jenis pesan yang berbeda dapat digambarkan pada diagram urutan.
6. *Guard Condition* merupakan tes yang harus dipenuhi agar pesan dapat dikirim. *Guard condition* ditempatkan di depan nama pesan (*message name*). Namun, ketika menggunakan sequence diagram untuk memodelkan skenario tertentu, *guard condition* biasanya tidak ditampilkan pada *sequence diagram* tunggal. Sebaliknya, *guard condition* hanya tersirat melalui adanya diagram *sequence* yang berbeda.
7. *Object Destruction* direpresentasikan sebagai tanda X yang ditempatkan di akhir garis hidup (*lifeline*) suatu objek untuk menunjukkan bahwa objek tersebut akan lenyap.
8. *Frame* menunjukkan konteks diagram urutan.



Gambar 2. 10 Contoh *Sequence Diagram*

2.5 Platform

Platform dalam konteks digital merupakan suatu wadah yang digunakan oleh banyak orang untuk berbagai keperluan. Secara sederhana, platform dapat diartikan sebagai alat yang digunakan untuk menjalankan suatu sistem sesuai dengan rencana program yang telah dirancang. Sebagai contoh, dalam kegiatan pembelajaran daring, platform yang digunakan biasanya didasarkan pada digitalisasi.

Di sisi lain, platform digital merujuk pada kumpulan perangkat lunak yang membentuk suatu sistem tertentu. Perangkat lunak ini dapat diakses melalui komputer pribadi (PC) atau sistem operasi *android*. Apabila digunakan pada sistem *android*, platform digital bisa berbentuk aplikasi. Platform digital saat ini sangat diminati, terutama karena peningkatan jumlah pengguna *smartphone* yang secara otomatis meningkatkan aktivitas *online* di dunia maya [29].

2.5.1 Pengembangan dengan Platform *No Code*

Platform pengembangan tanpa kode yang disebut NCDP, memberikan kemungkinan untuk mengembangkan perangkat lunak tanpa memerlukan pemrograman komputer konvensional, melainkan program dapat dikembangkan dengan menggunakan konfigurasi yang telah dipreparasi dan antarmuka pengguna. NCDP secara signifikan mengurangi waktu siklus pengembangan produk dan memungkinkan pengembang yang kurang terampil untuk berpartisipasi secara setara dalam proses pengembangan. Umumnya, lingkungan pengembangan NCDP mencakup fitur-fitur berikut [30]:

1. Antarmuka pengguna yang menawarkan berbagai modul atau blok yang dapat ditarik dan dilepas.
2. Alat visual untuk memodelkan antarmuka pengguna, konfigurasi data, dan berbagai kemampuan. Selain itu, juga menawarkan kemungkinan untuk menyertakan kode yang dikembangkan secara manual.
3. Koneksi untuk mengelola *cache* dan mengambil data.

Platfom pengembangan tanpa kode ini dirancang untuk melayani tujuan tertentu. Meskipun memiliki fitur-fitur yang disebutkan di atas, semua NCDP berbeda karena fungsionalitas yang dimaksudkan. Perbedaan muncul dalam mode operasi, integrasi, dan kebutuhan di pasar. Aplikasi yang dikembangkan dari NCDP dapat difokuskan pada satu segmen bisnis, seperti visualisasi alur kerja, atau bisa sangat luas dan mungkin memiliki alat perencanaan sumber daya perusahaan yang terintegrasi. NCDP mirip dengan bahasa pemrograman visual. Platfom tanpa kode populer di kalangan perusahaan yang

menginginkan transformasi cepat dari sistem tradisional ke sistem terdigitalisasi dengan menggunakan solusi berbasis *cloud* untuk aplikasi *mobile*. Lingkungan tanpa kode dan aksesorisnya disesuaikan untuk pengguna individu dan kebutuhan mereka, tidak seperti solusi TI tradisional. Hal ini membantu mengatasi hambatan pengembangan perangkat lunak seperti waktu, keuangan, dan Keahlian [30].

NCDP menawarkan terminal yang nyaman dan mudah digunakan untuk terhubung ke antarmuka pemrograman aplikasi perangkat lunak perusahaan. Selain itu, mengintegrasikan sistem perangkat lunak saat ini dan semua komponennya dengan solusi NCDP yang baru sangat cepat dan mudah. Penggunaan NCDP memberikan beberapa keuntungan, antara lain [30]:

1. Akses : Adopsi pengembangan aplikasi mobile tanpa kode yang meningkat pesat telah menghasilkan akses mudah ke platform NCDP bagi para pembuatnya. Kemudahan akses ini telah mengakibatkan pergeseran paradigma dalam industri pengembangan aplikasi. Sekarang, siapa pun dengan keterampilan dasar internet dan pengetahuan bisnis dapat mengembangkan aplikasi.
2. Kecepatan : NCDP menawarkan tingkat fleksibilitas yang mencolok dalam fungsionalitas antarmuka pengguna dan dengan demikian dapat memberikan pengalaman pengguna yang lebih baik. Tampilan data dan bentuk yang berbeda mengurangi waktu pengembangan dengan fleksibel untuk bagian yang berbeda dari aplikasi yang sama maupun aplikasi lain.
3. Kekayaan : Berbeda dengan pandangan umum, NCDP kaya fitur dan fungsionalitas. Selain itu, mereka menawarkan berbagai kemungkinan integrasi yang luas, memungkinkan pengembang mencapai persyaratan bisnis yang diinginkan. Platfom pengembangan tanpa kode memberikan peluang untuk mengurangi beban pada pekerja umum. Dengan menggunakan platform seperti ini, karyawan yang kurang terampil dapat mengambil tanggung jawab selama proses pengembangan aplikasi dan kemudian pengguna dapat menikmati peningkatan cepat pada aplikasi sesuai dengan kebutuhan yang berubah.

Popularitas yang meningkat dari NCDP disumbangkan oleh kemudahan akses bagi pengguna akhir yang mencoba memanfaatkan teknologi informasi untuk bisnis. Salah satu alasan dominan adalah bahwa, berbeda dengan platform tradisional dan *low code*, NCDP tidak memerlukan keterampilan pemrograman komputer apa pun, sehingga memungkinkan setiap anggota bisnis untuk berpartisipasi secara aktif dalam proses pengembangan aplikasi mereka. Di atas itu, NCDP memberikan kontrol kepada pengguna yang mendapatkan

manfaat dari proses pengembangan aplikasi seret dan lepas berkat konsep interpretasi yang didorong oleh model dari platform tersebut [30].

Salah satu kekhawatiran mencolok tentang platform pengembangan tanpa kode adalah keamanan. Popularitas yang meningkat telah memicu debat tentang kemampuan NCDP untuk melindungi data sensitif dari para peretas. Konsep ini muncul ketika perusahaan keamanan memberikan peringatan bahwa aplikasi yang dikembangkan di NCDP dibuat oleh personil non-teknis. Studi yang hati-hati dan sadar menunjukkan bahwa aplikasi yang dikembangkan dengan kode tradisional jauh lebih rentan dibandingkan dengan aplikasi yang dikembangkan menggunakan NCDP. Penjelasan sederhana untuk hal ini adalah bahwa aplikasi NCDP hanya memungkinkan pengguna untuk mengubah atau memanipulasi yang terlihat sehingga membatasi pengembang aplikasi untuk mengakses pemrograman sebenarnya dari platform. Hal ini memastikan bahwa fungsionalitas aplikasi tetap tidak dapat ditembus, dan keamanan tetap teguh. Platform pengembangan tanpa kode yang populer dan berkembang pesat termasuk Microsoft Power Apps, Airtable, Quickbase, dan Google AppSheet [30].

2.5.2 Google AppSheet

AppSheet merupakan platform *online* yang diakuisisi oleh Google pada tahun 2020. AppSheet adalah platform pengembangan tanpa kode yang memungkinkan pembuatan dan distribusi aplikasi *mobile*, tablet, dan *web* dengan mudah. Pembuatan aplikasi dengan AppSheet dapat terintegrasi ke berbagai sumber data, misalnya Google Spreadsheet, Excel, Cloud SQL, Salesforce, dan konektor serupa lainnya [1]. Appsheet bersifat dinamis dan dapat digunakan pada perangkat seluler atau *browser*. Desain antarmuka aplikasi menggunakan pola UX untuk membuat peta, kalender, dasbor, dan lainnya. Alur kerja otomatis juga dapat diintegrasikan ke dalam aplikasi untuk melakukan tindakan seperti mengirim pemberitahuan, menulis *e-mail*, membuat laporan khusus, dan mengedit data pada sumber yang terhubung [31].

Platform ini utamanya ditujukan untuk penggunaan bisnis, seperti *CRM*, manajemen proyek, dan laporan yang dipersonalisasi. AppSheet menganalisis struktur sumber data yang disediakan dan secara otomatis menghasilkan tampilan yang dapat ditampilkan dalam aplikasi. Pengguna dapat menyesuaikan tampilan yang dihasilkan dengan menampilkan atau menyembunyikan kolom tertentu atau menulis formula untuk pemrosesan dan agregasi data. AppSheet gratis untuk *prototyping* dan penggunaan pribadi, sementara biaya bulanan harus dibayarkan untuk aplikasi komersial. Selain itu, AppSheet menawarkan fitur *machine*

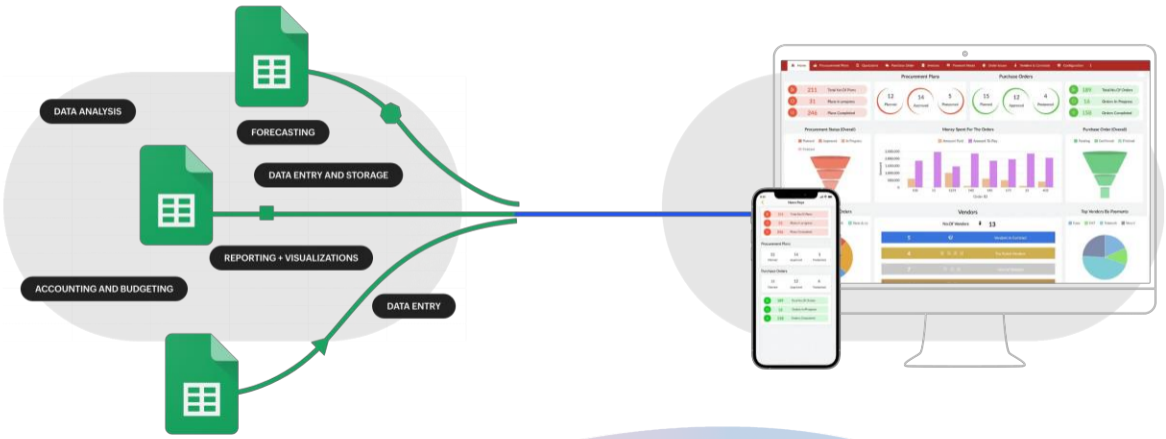
learning dan kecerdasan buatan canggih seperti prediksi nilai, pengenalan karakter optik (OCR), analisis sentimen, dan deteksi anomali. Namun, koneksi internet aktif dan aplikasi klien diperlukan untuk mengakses aplikasi AppSheet dan fiturnya karena aplikasi tersebut diterapkan di *cloud* [32].

2.5.2.1 Fitur pada Google AppSheet

AppSheet menawarkan fitur pengembangan aplikasi yang canggih dan modern. Fitur-fitur paling khas dan nyaman termasuk fasilitas untuk mengotomatisasi pemrosesan dan tampilan data, sehingga memudahkan alur informasi. Selain itu, AppSheet menampilkan sinkronisasi data secara *offline*. Aspek paling khas dan menguntungkan dari AppSheet adalah ketersediaan platform kecerdasan buatan Google yang canggih dan matang, yang dapat memungkinkan pembelajaran mesin. Penggunaan AppSheet tidak hanya memberikan kenyamanan dalam mengembangkan aplikasi tanpa kode, tetapi juga mengintegrasikan aplikasi dengan algoritma pembelajaran mesin yang dikembangkan oleh Google, sehingga menjadikannya salah satu platform pengembangan aplikasi yang paling canggih namun mudah digunakan. Penawaran yang ditingkatkan ini tidak hanya menghemat waktu dan sumber daya, tetapi juga memungkinkan aplikasi untuk memiliki otonomi dalam menjalankan beberapa aspek perangkat lunak [30].

Dari sudut pandang industri, perangkat lunak yang dikembangkan dengan AppSheet dapat menyimpan data secara lokal dan menyinkronkannya begitu internet tersedia. Hal yang membuat data lebih menarik adalah informasi tambahan yang muncul karena kecerdasan buatan AppSheet. Aplikasi menyimpan lokasi melalui sistem penentuan lokasi global perangkat dan dapat mengaitkan gambar dan informasi yang dipindai dengan data. Selain itu, semua ini dapat dibagikan dengan cepat sebagai tautan melalui email, sehingga membuat proses pengumpulan data, kolaborasi, dan pameran menjadi nyaman dan mudah. Dengan kemungkinan penyesuaian internal, AppSheet menawarkan kendali yang lebih baik dan kemampuan pengiriman kepada organisasi. Ini memberikan kemungkinan untuk dengan cepat mengubah aplikasi dengan perubahan ekosistem. Selain itu, kebijakan, fitur, dan informasi dapat diperbarui tanpa memerlukan personel IT yang ahli [30].

2.5.2.2 Proses pengembangan Google AppSheet



Gambar 2. 11 Proses Pengembangan Menggunakan Platform AppSheet

Pengembangan aplikasi menggunakan AppSheet mengikuti pendekatan yang sederhana namun efektif. Proses pengembangan biasanya mencakup perencanaan aplikasi, manajemen, dan keamanan aplikasi. Komponen dasar dari perangkat lunak yang berasal dari AppSheet adalah pembuatan Google spreadsheet yang berfungsi sebagai sumber data, yang diikuti oleh pembuatan aplikasi dengan cara menarik dan melepaskan *add-on* Google. Pada dasarnya, pada titik ini aplikasi sudah siap digunakan tetapi masih dapat disesuaikan, dan antarmuka dapat diperbarui. Aplikasi ini digunakan untuk memonitor ketersediaan di rak, dikembangkan menggunakan pembuat aplikasi seluler AppSheet. Empat persyaratan pengembangan direncanakan diikuti oleh mode manajemen aplikasi, dan pada akhirnya, mekanisme keamanan aplikasi diatur [30].

2.6 Basis Data

Basis data adalah kumpulan data yang terstruktur yang disimpan secara teratur dalam komputer sehingga dapat dengan mudah diakses, dikelola, dan diperbarui [2], [3]. Basis data biasanya disimpan dalam tabel, dan tabel berisi data atau entitas terkait, seperti orang, produk, pesanan, dan lain-lain. Tujuan utama dalam membuat basis data atau *database* adalah untuk memenuhi kebutuhan informasi pengguna dan aplikasi dengan lebih efisien, serta menyediakan mekanisme untuk mengambil Informasi yang dibutuhkan dengan cepat dan akurat [3].

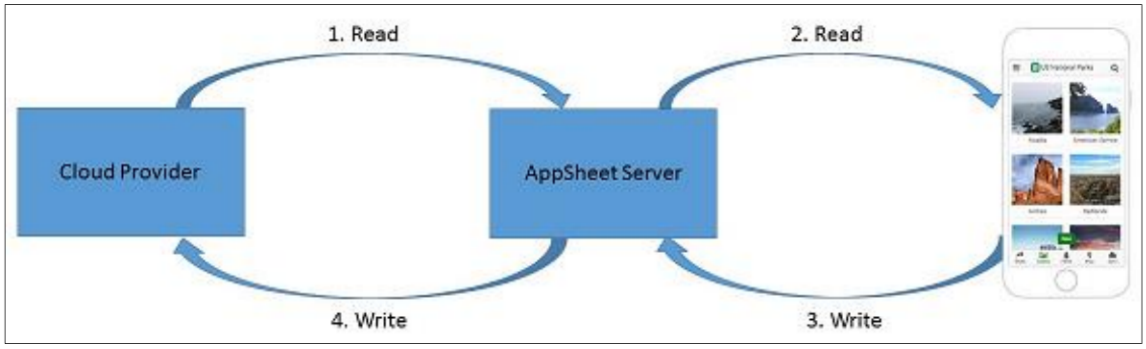
Salah satu peran utama *database* adalah menyimpan dan mengelola data secara efektif. Untuk mengelola data dalam jumlah besar dan terus berkembang, basis data sangatlah penting karena menyediakan landasan yang kuat dengan menyediakan platform terpusat dan terstruktur untuk menampung semua informasi. Dengan cara ini, Basis data tidak hanya berfungsi sebagi tempat penyimpanan data dan informasi tetapi juga sebagai alat

bagi perusahaan untuk membangun interaksi yang lebih personal dan tertarget dengan menciptakan pengalaman yang memuaskan dan disesuaikan dengan kebutuhan individu. Oleh karena itu, peran basis data dalam menyimpan data secara terstruktur dan mudah diakses berdampak langsung pada kemampuan perusahaan dalam memberikan layanan yang lebih baik dan terpersonalisasi. Selain itu, basis data juga memaikan peran sentral dalam mendukung pengambilan keputusan dan analisis data. Di era yang serba terintegrasi, analisis data menjadi salah satu faktor penentu keberhasilan bisnis. Basis data memungkinkan perusahaan tidak hanya menyimpan informasi tetapi juga menganalisis informasi tersebut dengan cermat untuk mengidentifikasikan tren, menghasilkan wawasan, dan mendukung pengambilan keputusan yang lebih tepat berdasarkan fakta [33].

Dalam konteks penerapan teknologi basis data, terdapat berbagai jenis basis data yang dapat digunakan, seperti *cloud database, key-value database, document-storage database, columnar database, embedded database, graph database, object-oriented database, centralized database, distributed database, hierarchical database, end-user database, operational database, relational database*. Setiap jenis basis data memiliki karakteristik dan manfaatnya masing-masing, dan pemilihan jenis basis data yang tepat harus selaras dengan kebutuhan dan tujuan bisnis Perusahaan [34], [35]. Beberapa aplikasi seperti Google Drive, DropBox, Office 365, dan Platform Spreadsheet dapat digunakan sebagai basis data berbasis *Cloud* [36].

2.6.1 Konsep Basis Data Dalam AppSheet

Aplikasi AppSheet dapat dioperasikan di perangkat seluler atau *browser*. Seluruh data tabel yang digunakan dalam aplikasi ini disimpan sebagai salinan lokal pada perangkat atau *browser*. Adanya salinan lokal ini memungkinkan aplikasi menjadi interaktif, responsif, dan dapat berfungsi secara *offline*. Platform harus memastikan bahwa salinan data lokal tersebut tetap sinkron dengan sumber data sebenarnya yang terletak pada penyedia data *cloud* (seperti Google Drive, Office365, atau SQLServer). Namun, aplikasi AppSheet tidak dapat terhubung langsung dengan penyedia data *cloud* tersebut. Oleh karena itu, aplikasi ini berkomunikasi dengan layanan *backend* AppSheet yang kemudian bertindak sebagai perantara untuk berkomunikasi dengan penyedia *cloud*. Struktur bertingkat ini diperlukan demi keamanan dan juga memberikan manfaat kinerja [37].



Gambar 2. 12 Proses Sinkronasi Data pada AppSheet

Beberapa tahapan sinkronasi data pada AppSheet memiliki penjelasan sebagai berikut [37].

1. Ketika aplikasi di perangkat seluler memulai proses sinkronasi, aplikasi tersebut mengajukan permintaan data ke server AppSheet, kemudian meneruskan permintaan tersebut ke *cloud provider*.
2. Setelah server AppSheet menerima data dari *cloud provider*, server akan menghitung kolom virtual yang telah ditentukan, kemudian mengembalikan data tersebut ke aplikasi di perangkat seluler. Aplikasi kemudian menyimpan data ini pada perangkat seluler.
3. Aplikasi mengirim setiap baris data yang ditambahkan, diperbarui, atau dihapus ke server AppSheet. Jika pengguna mengambil foto, gambar, atau tanda tangan, semua informasi tersebut dikirimkan bersama dengan baris data tersebut.
4. Ketika server AppSheet menerima informasi ini, server akan menulis data tersebut kembali ke *cloud provider*. Jika terdapat otomatisasi yang telah ditentukan, otomatisasi tersebut akan dijalankan (biasanya mengharuskan baris yang diperbarui dibaca ulang dari *cloud provider* sebelum tindakan dijalankan).

2.6.2 Google Spreadsheet

Google Spreadsheet merupakan produk Google yang dapat digunakan untuk menyimpan data dalam bentuk table. Spreadsheet adalah layanan gratis yang fungsinya mirip dengan Microsoft Excel, hanya saja berbasis *web*. Hal ini memungkinkan Spreadsheet bisa digunakan sebagai tempat penyimpanan data dari AppSheet nantinya [38].

Spreadsheet merupakan aplikasi atau program komputer yang digunakan untuk memanipulasi, mengumpulkan, dan menampilkan data yang disusun dalam kolom dan baris. Setiap sel dapat berisi teks, angka, formula, atau fungsi matematika yang memungkinkan pengguna untuk melakukan penghitungan otomatis dan analisis data. Pengolahan data dalam lembar kerja spreadsheet disimpan dalam sel-sel yang diberi nama sesuai dengan label yang

digunakan pada baris dan kolom. Label pada baris diberikan dalam bentuk angka berurutan, seperti 1,2,3, dan seterusnya. Sementara, label pada kolom menggunakan alfabet mulai dari A hingga Z, dan seterusnya [2], [3].

2.7 Prototyping

Prototyping adalah metode yang sangat cepat untuk mengembangkan dan menguji model operasi aplikasi baru melalui proses yang berulang dan interaktif sehingga dapat dimanfaatkan dengan baik [39]. Tujuan utama dari *prototyping* adalah untuk memberikan gambaran visual atau fungsional dari produk atau sistem yang akan dibangun. Prototipe digunakan untuk menguji konsep, mengumpulkan umpan balik dari pengguna atau pemangku kepentingan, dan mengidentifikasi kesenjangan atau perubahan yang diperlukan sebelum produk atau sistem dikembangkan sepenuhnya. *Prototyping* terdiri dari beberapa tahapan, yaitu [39], [40]:

1. Pengumpulan kebutuhan merupakan langkah dimana dilakukan pengidentifikasian semua perangkat dan masalah, serta menganalisis dan menentukan persyaratan umum sistem.
2. Membangun prototipe yang berfokus pada penyajian. Contohnya membuat masukan dan keluaran hasil sistem.
3. Evaluasi Prototipe, pada tahap ini, penting untuk memeriksa langkah 1 karena ini adalah faktor penentu keberhasilan dan merupakan proses yang sangat penting. Jika langkah 1 dan 2 hilang atau salah, maka akan sulit untuk melanjutkan ke langkah berikutnya di kemudian hari.
4. Mengkodekan sistem atau biasa disebut koding.
5. Menguji sistem untuk memastikan fungsi-fungsi tampilan dan kodingan sudah benar dan sesuai dengan sistem yang diharapkan.
6. Mengevaluasi sistem dengan meninjau ulang semua langkah yang pernah dilakukan untuk memastikan sistem sudah sesuai dengan kebutuhan atau belum.
7. Implementasi sistem oleh pengguna dan melakukan *maintenance* agar sistem tetap terjaga dan berfungsi dengan baik.

Ada beberapa jenis prototipe yang dapat digunakan selama pengembangan produk. Berikut ini adalah tiga jenis prototipe yang sering dipakai [41].

1. *Low Fidelity Prototype* merupakan jenis prototipe yang paling sederhana, menggunakan kertas dan alat tulis untuk menggambar sketsa antarmuka atau alur pengguna. Biasanya

jenis prototipe ini digunakan pada tahap awal pengembangan untuk merancang dan menguji konsep awal dan ide-ide dasar,

2. *Medium Fidelity Prototype* merupakan jenis prototipe yang lebih kompleks daripada *Low Fidelity Prototype*, karena mencakup beberapa fungsi dan interaksi yang mendekati produk akhir. Biasanya dibuat menggunakan perangkat lunak *prototypin*g atau alat pengembangan cepat. *Medium Fidelity Prototype* digunakan untuk menguji fitur dan visual produk pada tingkat detail yang lebih tinggi.
3. *High Fidelity Prototype* merupakan jenis prototipe yang paling mirip atau mendekati produk akhir. Prototipe ini menggunakan teknologi dan bahan yang sama dengan produk yang akan akan dikembangkan. Jenis prototipe ini digunakan untuk pengujian rancangan, visual, dan fungsionalitas produk secara menyeluruh. Biasanya, prototipe ini digunakan pada tahap akhir pengembangan.

2.8 Insentif

Insentif merupakan salah satu bentuk kompensasi atau imbalan berupa uang yang diberikan kepada karyawan yang dapat bekerja sesuai dengan standar yang telah ditentukan dan tidak berkaitan dengan gaji atau upah [42]. Insentif terdiri dari tiga jenis utama yang sering dikenal, yaitu [43]:

1. Insentif finansial, merupakan pemberian dalam bentuk uang atau keuntungan finansial lainnya sebagai imbalan atas pencapaian tujuan tertentu. Contohnya, bonus kinerja, komisi penjualan, atau gaji yang lebih tinggi bagi karyawan yang mencapai target tertentu.
2. Insentif non-finansial, merupakan pemberian penghargaan atau keuntungan non-keuangan. Contohnya, pengakuan public, sertifikat penghargaan, kesempanan untuk pelatihan, pengembangan karir, dan sebagainya.
3. Insentif sosial, merupakan insentif yang didasarkan pada kebutuhan individu akan penerimaan sosial. Contohnya pengakuan dari rekan kerja, keluarga, teman yang dapat mendorong karyawan untuk mencapai tujuan tertentu.

Tujuan utama dari insentif adalah memberikan tanggung jawab penuh dan memotivasi karyawan untuk meningkatkan produktivitas dan mencapai tujuan organisasi/perusahaan sehingga dapat meningkatkan keuntungan [42]. Beberapa tujuan lain dari pemberian insentif, yaitu [43]:

1. Memberikan penghargaan kepada karyawan yang berprestasi.

2. Memastikan karyawan akan berusaha semaksimal mungkin utnuk mencapai tujuan perusahaan.
3. Mengukur upaya karyawan melalui prestasi kerjanya.