BAB II
TINJAUAN PUSTAKA

1.1 Konsep Sistem Informasi

1.1.1 Sistem

Suatu sistem pada dasarnya adalah sekelompok unsur yang erat hubungannya satu dengan yang lain, yang berfungsi bersama-sama untuk mencapai tujuan tertentu. Secara sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Dari definisi ini dapat dirinci lebih lanjut pengertian secara umum, yaitu [2]:

1. Setiap sistem terdiri dari unsur-unsur.
2. Unsur-unsur tersebut merupakan bagian terpadu sistem yang bersangkutan.
3. Unsur sistem tersebut bekerja sama untuk mencapai tujuan sistem.
4. Suatu sistem merupakan bagian dari sistem lain yang lebih besar.

Tujuan sistem merupakan target atau sasaran akhir yang ingin dicapai oleh suatu sistem. Sistem ada karena tujuan. Sistem dibangun agar tujuan tercapai tidak menyimpang sehingga resiko kegagalan bisa diminimalkan. Agar supaya target tersebut bisa tercapai secara efektif dan efisien maka target atau sasaran tersebut harus diketahui terlebih dahulu ciri-ciri atau kriterianya agar sistem dapat dibangun dan menuntun dengan jelas dan tegas setiap aktivitas menuju tujuan yang telah ditetapkan [3].

Sistem terdiri dari 3 fungsi, yang menunjukkan bahwa sistem sebagai proses tidak bisa berdiri sendiri, yaitu [3]:

1. *Input*

   Input adalah segala sesuatu yang masuk kedalam suatu sistem. *Input* ini bervariasi bisa berupa energi, manusia, data, modal, bahan baku, layanan atau lainnya. *Input* merupakan pemicu bagi sistem untuk melakukan proses yang diperlukan.

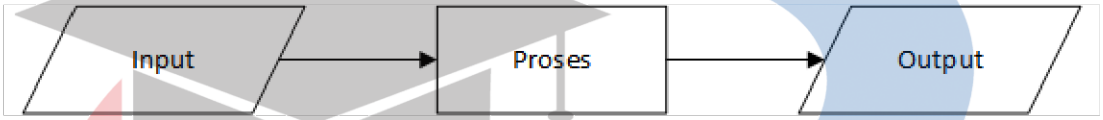
2. Proses

   Proses merupakan perubahan dari *input* menjadi *output*. Proses mungkin dilakukan oleh mesin, orang, atau komputer. Umumnya kita mengetahui bagaimana *input* dirubah menjadi *output* akan tetapi pada situasi tertentu proses tidak diketahui

secara detail karena perubahan ini terlalu kompleks. Proses mungkin berupa perakitan yang menghasilkan satu macam *output* dari berbagai macam *input* yang disusun berdasarkan aturan tertentu.

3. *Output*

   Output seperti halnya *input* mungkin berbentuk produk, servis, informasi dalam bentuk *print out* komputer atau energi seperti *output* dari dinamo. *Output* adalah hasil dari suatu proses yang merupakan tujuan dari keberadaan sistem. Seperti dijelaskan sebelumnya *output* suatu sistem bisa menjadi *input* untuk sistem yang lain yang setelah diproses menjadi *output* yang lain.



Gambar 1.1 Tiga Fungsi Sistem

Sistem memiliki pendekatan yang ditekankan dalam sebuah prosedur jaringan kerja secara saling hubung, mengelompok serta bekerja bersama untuk mendapatkan pencapaian sasaran yang diinginkan. Dalam prosedur terdapat instruksi dengan tahapan-tahapan yang berurutan dimana apa (*what*) yang dikerjakan, siapa (*who*) yang melakukan pekerjaan, kapan (*when*) pengerjaannya dan bagaimana (*how*) cara kerjanya. Pendekatan lebih menekankan pada bagaimana komponen dengan artian bahwa "sistem" merupakan interaksi dari kumpulan elemen dalam suatu tujuan yang dicapai [4].

1.1.2 Informasi

Informasi merupakan hasil dari pengolahan data, akan tetapi tidak semua hasil dari pengolahan tersebut bisa menjadi informasi, hasil pengolahan data yang tidak memberikan makna atau arti serta tidak bermanfaat bagi seseorang bukanlah merupakan informasi bagi orang tersebut Dari uraian tentang informasi ini, ada tiga hal penting yang harus diperhatikan di sini, yaitu [3]:

1. Informasi merupakan hasil pengolahan data.
2. Memberikan makna atau arti.
3. Berguna atau bermanfaat.

Secara umum, informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan. Informasi merupakan data yang telah diklasifikasikan atau diolah atau diinterpretasi untuk digunakan dalam proses pengambilan keputusan [2].

Fungsi informasi yaitu menambahkan pengetahuan atau mengurangi ketidakpastian pemakai informasi, karena informasi berguna memberikan gambaran tentang suatu permasalahan sehingga pengambil keputusan dapat menentukan keputusan lebih cepat. Informasi juga memberikan standar, aturan, maupun indikator bagi pengambil keputusan [5].

Menurut Raymond McLeod, suatu informasi yang berkualitas harus memiliki ciri-ciri sebagai berikut [2]:

1. Akurat, informasi harus mencerminkan keadaan yang sebenarnya dan informasi tersebut harus bebas dari kesalahan-kesalahan.
2. Tepat waktu, informasi itu harus tersedia atau ada pada saat informasi tersebut diperlukan dan tidak terhambat.
3. Relevan, informasi yang diberikan harus sesuai dengan yang dibutuhkan.
4. Lengkap, informasi harus diberikan secara lengkap karena bila informasi yang dihasilkan sebagian-sebagian akan memengaruhi dalam mengambil keputusan.
5. *Correctness*, berarti informasi yang dihasilkan atau dibutuhkan harus memiliki kebenaran.
6. *Security*, berarti informasi yang dihasilkan mempunyai manfaat yang lebih besar dibandingkan dengan biaya mendapatkannya dan sebagian besar informasi tidak dapat ditaksir keuntungannya dan dengan satuan nilai uang tetapi dapat ditaksir nilai efektivitasnya.

1.1.3 Sistem Informasi

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 [2]. Sistem informasi adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat

manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan informasi yang diperlukan untuk pengambilan keputusan [2]. Sistem informasi berfungsi untuk menghasilkan informasi untuk memenuhi kebutuhan aktivitas organisasi. Informasi yang bernilai tinggi dihasilkan oleh suatu sistem informasi yang efektif dan efisien. Untuk itu, sistem informasi yang efektif dan efisien menghendaki intervensi manajemen secara tepat [6].

Menurut John Burch dan Gary Grudnitski, sistem informasi terdiri dari komponen-komponen sebagai berikut [6]:

1. Blok Masukan (*Input Block*)

   Input mewakili data yang masuk ke dalam sistem informasi. *Input* disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.
2. Blok Model (*Model Block*)

   Blok model ini terdiri dari kombinasi prosedur, logika dan model matematik yang akan memanipulasi data *input* dan data yang tersimpan di basis data dengan cara tertentu untuk menghasilkan *output* yang diinginkan.
3. Blok Keluaran (*Output Block*)

   Produk dari sistem informasi adalah *output* yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.
4. Blok Teknologi (*Technology Block*)

   Teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan *output* dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri 3 bagian utama, yaitu:

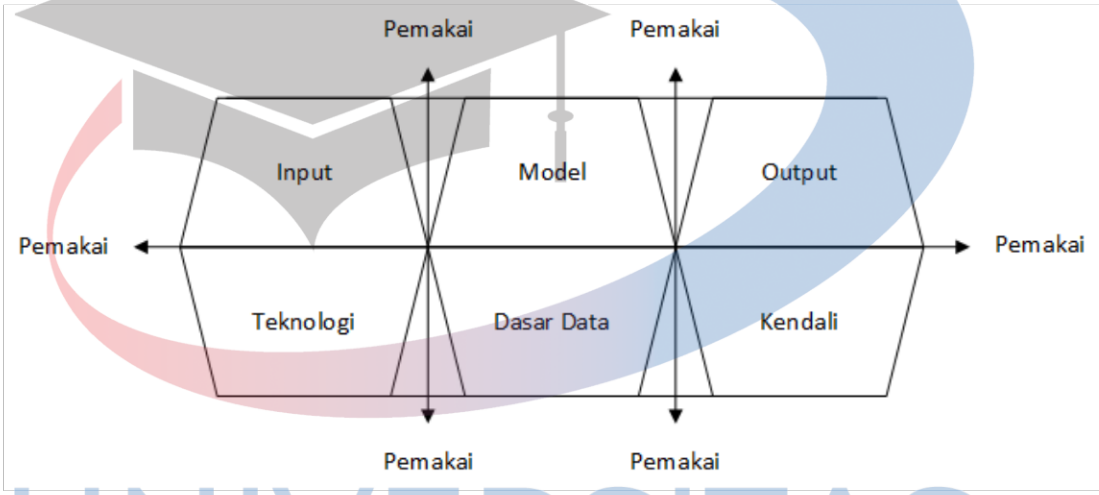
   a. Teknisi (*Humanware* atau *Brainware*)
   b. Perangkat Lunak (*Software*)
   c. Perangkat Keras (*Hardware*)
5. Blok Basis Data (*Database Block*)

   Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Basis data diakses atau dimanipulasi

dengan menggunakan perangkat lunak paket yang disebut dengan DBMS (*Database Management System*).

6. Blok Kendali (*Control Block*)

   Banyak hal yang dapat merusak sistem informasi, seperti misalnya bencana alam, *temperature*, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidakefisienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi.



Gambar 1.2 Komponen Sistem Informasi

1.2 Aplikasi *Mobile*

Aplikasi *mobile* adalah perangkat lunak yang berjalan pada perangkat *mobile* seperti *smartphone* atau *tablet*. Aplikasi *mobile* juga dikenal sebagai aplikasi yang dapat diunduh dan memiliki fungsi tertentu sehingga menambah fungsionalitas dari perangkat mobile itu sendiri [7]. Aplikasi *mobile* dapat diartikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah-pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil [7]. Dengan menggunakan aplikasi *mobile*, pengguna dapat dengan mudah melakukan berbagai macam aktifitas mulai dari hiburan, berjualan, belajar, mengerjakan pekerjaan kantor, *browsing*, dan lain sebagainya [8].

1.3 Pengembangan Web

1.3.1 Aplikasi Berbasis Web

Aplikasi berbasis web (*web-based application, web application*) adalah aplikasi perangkat lunak *client-server* dimana klien (*user interface*) berjalan di *web browser*. Aplikasinya sendiri disimpan di sebuah *web server*, begitu juga data-data disimpan di *database server* [9]. Aplikasi berbasis web memudahkan pengembang karena aplikasi ini dapat berjalan di berbagai *platform* sistem operasi. Tentu saja karena dijalankan melalui *web browser*. Oleh karena itu, aplikasi dapat dijalankan di sistem berbasis *Windows*, *Linux*, atau *macOS* [9]. Pada dasarnya, Aplikasi berbasis web adalah sistem perangkat lunak yang berdasarkan pada teknologi dan standar *World Wide Web Consortium* (W3C) [10].

Interaksi antara pengguna dengan aplikasi berbasis web dibagi ke dalam 3 langkah, yaitu [10]:

1. Permintaan (*Request*)

   Pengguna mengirimkan permintaan ke *web server*, biasanya melalui halaman web yang ditampilkan pada *web browser*.

2. Pemrosesan (*Process*)

   Web server menerima permintaan yang dikirimkan oleh pengguna, kemudian memproses permintaan tersebut.

3. Jawaban (*Response*)

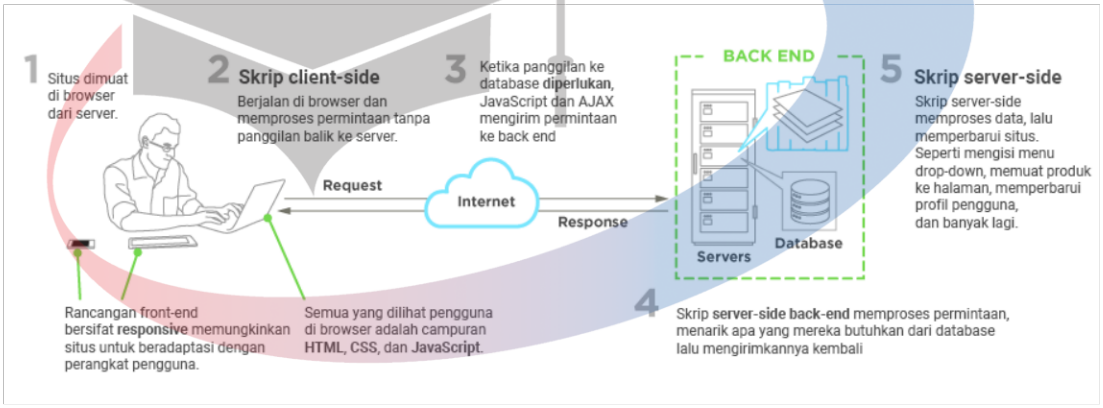
   Browser menampilkan hasil dari permintaan pada *browser window*.

1.3.2 Pengembangan *Front-End* dan *Back-End*

Front-end adalah segala sesuatu yang menghubungkan antara pengguna dengan sistem *back-end*. Biasanya merupakan sebuah *user interface* dimana pengguna akan berinteraksi dengan sistem [11]. Berdasarkan dari definisi tersebut, *front-end* adalah salah satu bagian utama dari aplikasi web yang menghubungkan secara langsung dengan pengguna melalui *user interface*. Pekerjaan yang sering muncul sebagai seorang pengembang *front-end* adalah perancang *user interface* dan *user experience*. Seorang pengembang *front-end* tidak akan membuat program atau aplikasi yang berjalan di *logic* bisnis, tapi fokusnya akan lebih banyak ke rancangan antarmuka

(*user interface*) dan bagaimana membuat rancangan yang nyaman digunakan oleh pengguna (*user experience*) [11].

Back-end atau sering disebut *server-side* pada dasarnya adalah tempat dimana proses suatu aplikasi atau sistem berjalan. Pada sisi *back-end*, data ditambahkan, diubah atau dihapus. *Back-end* mengurusi segala sesuatu yang biasanya tidak dilihat atau berinteraksi langsung kepada pengguna, seperti *database* dan *server* [11]. Orang yang bekerja sebagai pengembang *back-end* adalah *programmer* atau pengembang yang fokus pada keamanan, rancangan sistem, dan manajemen data pada sistem. Pengembang *back-end* dibutuhkan dalam pengembangan sistem atau aplikasi dinamis yang memiliki data yang selalu berubah-ubah [11].



Gambar 1.3 Hubungan Antara *Front-End* dan *Back-End* dalam Web

JavaScript adalah bahasa yang digunakan untuk membuat program yang digunakan agar dokumen *Hypertext Markup Language* (HTML) yang ditampilkan dalam *web browser* menjadi lebih interaktif, tidak sekedar indah saja. *JavaScript* memberikan beberapa fungsionalitas ke dalam suatu halaman web, sehingga dapat menjadi sebuah program yang disajikan dengan menggunakan antarmuka web [12].

JavaScript merupakan bahasa *script*, bahasa yang tidak memerlukan *compiler* untuk menjalankannya, cukup dengan *interpreter*. Tidak perlu ada proses *compilation* terlebih dahulu agar program dapat dijalankan. *Web browser* seperti *Netscape Navigator* dan *Internet Explorer* adalah dua contoh *interpreter*, karena kedua browser ini telah dilengkapi dengan *interpreter JavaScript*. Tetapi tidak semua *web browser* dapat menjadi *interpreter JavaScript* karena belum tentu *web browser* tersebut dilengkapi dengan interpreter *JavaScript* [12]. *JavaScript* adalah bahasa *script* yang

ringan dan mudah digunakan. Dengan adanya *JavaScript* ini, maka kini halaman web tidak sekedar menjadi halaman data dan informasi saja, tetapi juga dapat menjadi suatu program aplikasi dengan antarmuka web [12].

Umumnya, program *JavaScript* adalah program yang ditanamkan (disisipkan) ke dalam halaman web, sehingga halaman (dokumen) web menjadi sebuah aplikasi yang berjalan di dalam *web browser*. Beberapa sistem operasi menggunakan *JavaScript* untuk membuat aplikasi *non-web*, seperti sistem operasi *Microsoft Windows*, yang menggunakan istilah *Windows Scripting Host* (WSH) sebagai *interpreter JavaScript* dan *VBScript*, sehingga program yang dibuat dengan *JavaScript* dan *VBScript* dapat langsung dijalankan di atas sistem operasi, tanpa harus menggunakan *web browser* terlebih dahulu [12].

Framework adalah suatu struktur konseptual dasar yang digunakan untuk memecahkan atau menangani suatu masalah yang kompleks. Singkatnya, *framework* adalah wadah atau kerangka kerja dari sebuah aplikasi web yang akan dibangun. Dengan menggunakan kerangka tersebut, waktu yang digunakan dalam membuat aplikasi web lebih singkat dan memudahkan dalam melakukan perbaikan [13].

Framework juga dapat diartikan sebagai kumpulan *script* (terutama *class* dan *function*) yang dapat membantu *programmer* atau pengembang dalam menangani berbagai masalah-masalah dalam pemrograman, seperti koneksi ke *database*, pemanggilan *variable*, *file*, dan lain-lain sehingga pekerjaan pengembang lebih fokus dan lebih cepat dalam membangun aplikasi [14]. *Framework* adalah komponen pemrograman yang siap digunakan ulang kapan saja sehingga *programmer* tidak harus membuat *script* yang sama untuk tugas yang sama [14]. Secara sederhana, bisa dijelaskan bahwa *framework* adalah kumpulan fungsi (*libraries*) sehingga seorang *programmer* tidak perlu lagi membuat fungsi-fungsi dari awal dan biasanya disebut kumpulan *library*. *Programmer* cukup memanggil kumpulan *library* atau fungsi yang sudah ada di dalam *framework* yang sudah pasti cara menggunakan fungsi-fungsi itu sudah ditentukan sesuai aturan masing-masing [14]. Berikut adalah beberapa *framework* yang dibuat dengan bahasa pemrograman *JavaScript*, yaitu:

1. *React Native*

   React Native adalah sebuah *framework JavaScript* yang dikembangkan oleh *Facebook*. *React Native* memungkinkan pihak pengembang untuk membuat

aplikasi *mobile* atau seluler pada *platform Android* atau *iOS* menggunakan teknologi web [15]. Saat pertama kali diluncurkan, *React Native* hanya dapat digunakan untuk mengembangkan aplikasi berbasis *iOS*. Namun, sekarang *React Native* sudah dapat digunakan untuk mengembangkan aplikasi berbasis *Android* dan menjadikan *framework React Native* sebagai *framework* yang bersifat *hybrid* karena hanya dengan menulis satu kali kode program yang dapat langsung diimplementasikan pada *iOS* dan *Android* [16]. *React Native* akan mengkompilasi aplikasi ke dalam kode *native*, dimana untuk *Android*, akan dikompilasi dengan bahasa pemrograman *Java*, dan untuk *iOS*, akan dikompilasi dengan bahasa pemrograman *Objective-C* [15]. *React Native* dibuat berdasarkan *ReactJS*. Sama seperti *ReactJS*, *React Native* ditulis dari perpaduan *JavaScript* dan *syntax* tambahan *JavaScript* yang disebut JSX [17]. Berikut merupakan beberapa alasan penting mengapa *framework* ini sangat direkomendasikan untuk pengembangan produk aplikasi yang lebih efektif dan efisien, yaitu [18]:

a. Mempunyai kredibilitas yang tinggi

   Framework ini telah banyak digunakan oleh perusahaan besar seperti *Tesla*, *Instagram*, hingga *Walmart*. Selain itu, *React Native* juga cocok untuk digunakan oleh *startup*. Dimana, proses pengembangan menggunakan bahasa pemrograman *JavaScript* yang cocok untuk pengembangan *cross-platform*.

b. Kemudahan dari sisi teknis

   Proses pengembangan program juga mudah untuk dipelajari dan diimplementasikan.

c. Tidak memerlukan *resource* yang besar

   Proses pengembangan yang cukup efektif dan tidak menggunakan kode program yang terlalu banyak.

2. *Express.js*

   Express.js adalah sebuah *framework* dari *JavaScript* yang digunakan untuk membangun sebuah *server* pada *website*. *Express.js* menyediakan banyak fitur untuk membangun *web application* maupun *RESTful API*. Salah satu fitur yang dimiliki oleh *Express.js* adalah mengembangkan *RESTful API* dengan cepat dan mudah yang dapat menunjang dalam pengembangan sistem. Selain dapat membuat *RESTful API* dengan mudah, *Express.js* juga dapat membuat *routing* dengan cukup

mudah. *Routing* adalah menentukan bagaimana suatu aplikasi merespon *request client* ke *endpoint* tertentu, yang merupakan *path* atau *request HTTP Method* tertentu, seperti *GET*, *POST*, *PUT*, dan sebagainya [19]. Terdapat kelebihan yang dimiliki oleh *Express.js* dari segi pengembangan *back-end*, yaitu [20]:

a. Menggunakan *JavaScript*

   Express.js menggunakan bahasa pemrograman *JavaScript*, dengan pengembangan *front-end* dan *back-end* yang dikerjakan dengan bahasa pemrograman yang sama, hal ini tentunya akan membuat waktu proses pengembangan menjadi lebih cepat.

b. Didukung oleh *Google V8 Engine*

   Express.js didukung oleh mesin *Google V8*, sehingga membuat kinerja dari aplikasi lebih baik.

c. Dukungan komunitas yang cukup besar

   Bantuan dari komunitas lebih mudah didapatkan jika pengembang mengalami masalah saat bekerja dengan *Express.js*.

d. Mendukung *caching*

   Express.js mendukung fitur *caching*, keuntungan dari fitur ini adalah pengeksekusian kode tidak perlu lagi dilakukan secara berulang. Sehingga hal ini membuat halaman web memuat lebih cepat dari sebelumnya. Alhasil performa dari aplikasi yang dikembangkan akan semakin tinggi.

1.4 Basis Data

Basis data (*database*) merupakan gabungan *file* data yang dibentuk dengan hubungan atau relasi yang logis dan dapat diungkapkan dengan catatan serta bersifat independen [21]. Basis data didefinisikan sebagai tempat berkumpulnya data yang saling berhubungan dalam suatu wadah (organisasi atau perusahaan) bertujuan agar dapat mempermudah dan mempercepat untuk pemanggilan atau pemanfaatan kembali data tersebut [21]. Penggunaan basis data pada komputer memang tidak terlepas dari hubungan atau relasi antar data dalam bentuk *file* [21]. Menurut Noor, *file* adalah suatu pengumpulan yang terorganisir dari catatan yang saling berhubungan. Misalnya satu garis (*line*) dari faktur dapat berbentuk item, suatu faktur dapat membentuk catatan, suatu serangkaian catatan yang sedemikian ini dapat membentuk suatu *file*,

pengumpulan dari *file* kontrol keuangan dapat berbentuk perpustakaan (*library*), dan keseluruhan perpustakaan yang digunakan oleh suatu organisasi dalam membentuk bank data [21].

Basis data akan menggunakan media penyimpanan (*storage*), yaitu berkaitan dengan setiap alat yang dapat menerima data yang dapat disimpan, dan dapat dipanggil kembali data itu pada waktu berikutnya atau setiap alat yang dapat digunakan untuk menyimpan data [21].

Dengan bantuan basis data ini, diharapkan bahwa sistem informasi yang dibuat dapat terintegrasi antara bagian atau departemen yang satu dengan yang lainnya, sehingga pada akhirnya tidak ada pembatas area dalam perusahaan. Walaupun dalam pelaksanaannya tiap data akan dibatasi oleh penggunanya, namun semua hanya ditujukan untuk membatasi pengaksesan data saja agar tidak terjadi pembuatan manipulasi data oleh orang yang tidak berkepentingan terhadap data tersebut [21].

Adapun pengertian sistem basis data adalah suatu sistem penyusunan dan pengelolaan *record* dengan menggunakan komputer, dengan tujuan untuk menyimpan atau merekam serta memelihara data secara lengkap pada sebuah organisasi atau perusahaan, sehingga mampu menyediakan informasi yang optimal yang diperlukan pemakai untuk kepentingan proses pengambilan keputusan [21]. Komponen dasar sistem basis data digunakan untuk membantu kelancaran dari pembuatan dan manajemen basis data, terdapat 4 komponen dasar sistem basis data, yaitu [21]:

1. Data

   Data yang digunakan dalam sebuah basis data, haruslah mempunyai ciri-ciri sebagai berikut:

   a. Data disimpan secara terintegrasi (*integrated*), yaitu basis data merupakan kumpulan dari berbagai macam *file* dari aplikasi-aplikasi yang berbeda yang disusun dengan cara menghilangkan bagian-bagian yang rangkap (*redundant*).
   b. Data dapat dipakai secara bersama-sama (*shared*), yaitu masing-masing bagian dari basis data dapat diakses oleh pemakai dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

2. Perangkat Keras (*Hardware*)

   Terdiri dari semua peralatan perangkat keras komputer yang digunakan untuk pengelolaan sistem basis data, seperti:

a. Peralatan untuk penyimpanan.
b. Peralatan *input* dan *output*.
c. Peralatan komunikasi data.

3. Perangkat Lunak (*Software*)

   Berfungsi sebagai perantara (*interface*) antara pemakai dengan data fisik pada basis data, dapat berupa:

   a. *Database Management System* (DBMS).
   b. Program-program aplikasi dan prosedur-prosedur yang lain, seperti *Oracle*, *SQL Server*, *MySQL*, dll.

4. Pengguna (*User*)

   Terbagi menjadi 2 bagian, yaitu:

   a. *Programmer*, orang atau tim membuat program aplikasi yang mengakses basis data dengan menggunakan bahasa pemrograman.
   b. *End user*, orang yang mengakses basis data melalui *terminal* dengan menggunakan *query language* atau program aplikasi yang dibuat oleh *programmer*.

MySQL merupakan perangkat lunak yang tergolong sebagai *database management system* (DBMS) yang bersifat *open source*. *Open source* menyatakan bahwa perangkat lunak ini dilengkapi dengan kode sumber (kode yang dipakai untuk membuat *MySQL*), selain tentu saja bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi, dan bisa diperoleh dengan cara mengunduh di internet secara gratis [22]. *MySQL* awalnya dibuat oleh perusahaan konsultan bernama *TcX* yang berlokasi di Swedia. Kemudian, perangkat lunak ini di bawah naungan perusahaan *MySQL AB*. Namun, saat ini *MySQL* dimiliki oleh *Oracle Corporation* dan tersedia pula versi yang komersial [22]. *MySQL* adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan *database* sebagai sumber dan pengelolaan datanya. Kepopuleran *MySQL* antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan [23].

Sebagai DBMS, *MySQL* memiliki sejumlah fitur seperti yang dijelaskan sebagai berikut [22]:

1. *Multiplatform*

MySQL tersedia pada beberapa *platform* (*Windows*, *Linux*, *Unix*, dan lain-lain).

2. Andal, cepat, dan mudah digunakan

   MySQL tergolong sebagai *database server* (*server* yang melayani permintaan terhadap *database*) yang andal, dapat menangani *database* yang besar dengan kecepatan tinggi, mendukung banyak sekali fungsi untuk mengakses *database*, dan sekaligus mudah untuk digunakan.

3. Jaminan keamanan akses

   MySQL mendukung pengamanan *database* dengan berbagai kriteria pengaksesan. Sebagai gambaran, dimungkinkan untuk mengatur pengguna tertentu agar bisa mengakses data yang bersifat rahasia (misalnya gaji pegawai), sedangkan pengguna lain tidak boleh.

4. Dukungan *SQL*

   Seperti tersirat dalam namanya, *MySQL* mendukung perintah *Structured Query Language* (SQL). Sebagaimana diketahui, SQL merupakan standar dalam pengaksesan *database* relasional. Pengetahuan akan SQL akan memudahkan siapa pun dalam menggunakan *MySQL*.

1.5 *Rapid Application Development* (RAD)

Rapid Application Development adalah salah satu model proses pengembangan perangkat lunak *sequential linear* yang menekankan pada siklus pengembangan yang sangat pendek [13]. RAD menggunakan metode *iterative* (berulang) dalam mengembangkan sistem dimana *working model* (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan untuk menetapkan kebutuhan (*requirement*) pengguna dan selanjutnya disingkirkan [24]. Jika kebutuhan dapat dipahami dengan baik, tidak menutup kemungkinan tim pengembangan akan menciptakan sistem yang sempurna secara fungsional dalam waktu kira-kira 60 sampai 90 hari [13]. Pada saat RAD diimplementasikan, maka pengguna bisa menjadi bagian dari keseluruhan proses pengembangan sistem dengan bertindak sebagai pengambil keputusan pada setiap tahapan pengembangan. RAD juga menghasilkan suatu sistem dengan cepat karena sistem yang dikembangkan dapat memenuhi keinginan dari para pemakai sehingga dapat mengurangi waktu untuk pengembangan

ulang setelah tahap implementasi [25]. Model RAD memiliki 3 tahapan sebagai berikut [24]:

1. Tahap Perencanaan Persyaratan (*Requirements Planning*)

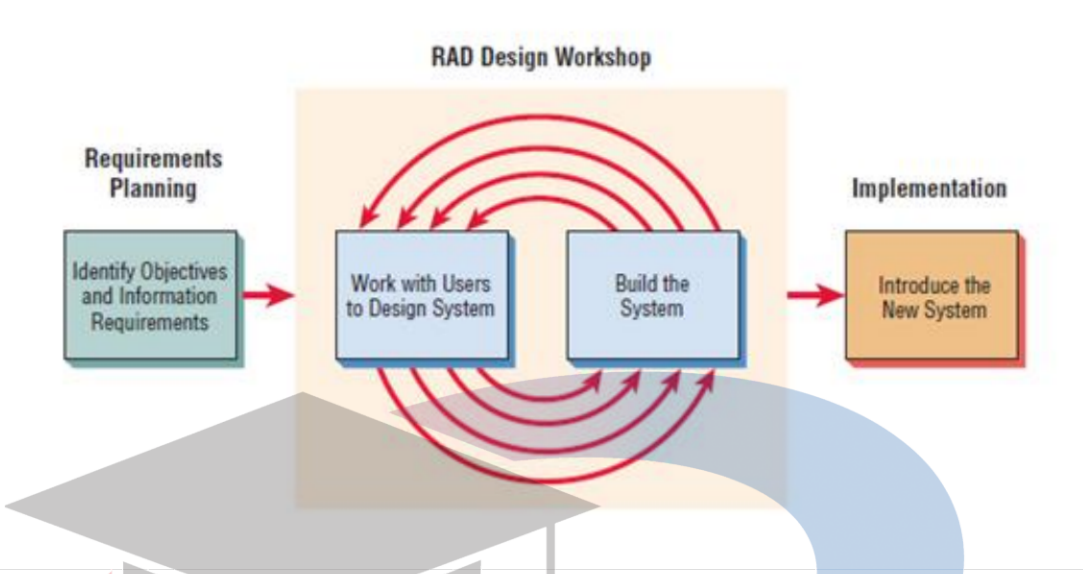
   Pada tahap pertama, pengguna dan analis melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan informasi untuk mencapai tujuan. Orientasi dalam fase ini adalah menyelesaikan masalah-masalah perusahaan. Meskipun teknologi informasi dan sistem bisa mengarahkan sebagian dari sistem yang diajukan, fokusnya akan selalu tetap pada upaya pencapaian tujuan-tujuan perusahaan [25]. Fase ini merupakan hal terpenting karena adanya keterlibatan dari kedua belah pihak.

2. Tahap Desain Sistem (*Design System*)

   Pada tahap kedua, peran pengguna sangat diperlukan sebagai penentu apakah tujuan sudah tercapai karena proses ini mencakup proses desain dalam bentuk *prototype* dan perbaikan-perbaikan yang akan dilakukan apabila masih terdapat ketidaksesuaian desain antara pengguna dan analis. Pengguna dapat langsung memberikan komentar apabila terdapat ketidaksesuaian pada desain.

3. Tahap Implementasi (*Implementation*)

   Pada tahap terakhir, *programmer* atau pengembang mulai membangun sistem berdasarkan desain yang telah disetujui oleh pengguna dan analis. Sebelum diimplementasikan pada suatu organisasi, proses pengujian akan dilakukan terlebih dahulu terhadap sistem tersebut untuk mengetahui apakah ada kesalahan atau tidak. Pada tahap ini, pengguna biasa memberikan tanggapan akan sistem yang sudah dibuat sebelum menyetujui sistem tersebut.



Gambar 1.4 Tahapan Model RAD

1.6 Teknik Pengembangan Sistem Informasi

1.6.1 Model *Use Case*

Use case diagram merupakan diagram yang menjelaskan secara visual konteks dari interaksi antara aktor dengan sistem. Setiap *use case* menyatakan spesifikasi perilaku (fungsionalitas) dari sistem yang sedang dijelaskan yang memang dibutuhkan oleh aktor untuk memenuhi tujuannya. Namun demikian, penjelasan detail dari interaksi yang terjadi antara aktor dan sistem, berkaitan dengan sebuah *use case* tertentu, harus dijelaskan secara deskriptif dalam sebuah *use case scenario*. Oleh karena itu, *use case scenario* dan *use case diagram*, yang dibutuhkan dalam pemodelan *use case* dari sebuah sistem, harus mampu menjelaskan fungsionalitas sistem secara lengkap dan valid [26].

Setiap *use case* menyatakan perilaku yang harus dijalankan oleh sistem dalam kaitannya dengan satu atau lebih aktor. Oleh karena itu, *use case* merupakan abstraksi dari interaksi yang terjadi antara aktor dengan sistem sehingga tujuan dari aktor bisa tercapai. Berdasarkan perspektif ini maka sebuah *use case* semestinya dipandang dari sisi aktor dan bukan dari sisi sistem, sehingga penamaan *use* case juga didasarkan atas tujuan yang ingin dicapai oleh aktor [26].

1.6.1.1 *Use Case Scenario*

Use case scenario merupakan penjelasan secara tekstual dari sekumpulan skenario interaksi. Setiap skenario mendeskripsikan urutan aksi atau langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik yang berhasil maupun gagal [26].

Use case scenario dijelaskan secara tekstual dalam beberapa format tergantung kebutuhannya, yaitu singkat (*brief*), informal (*casual*), atau lengkap (*fully dressed*), yang bisa dijelaskan dalam bentuk tabel dengan 1 kolom atau 2 kolom. Pada format singkat, penjelasan diberikan cukup 1 paragraf yang mengacu hanya pada skenario yang berhasil. Pada format informal, penjelasan diberikan dalam beberapa paragraf yang mencakup semua skenario, baik yang berhasil maupun gagal. Sedangkan, pada format lengkap, penjelasan dibuat secara detail disertai dengan bagian-bagian pendukung yang penting. Format terakhir ini yang banyak digunakan di dalam praktik. Bagian-bagian penting tersebut adalah [26]:

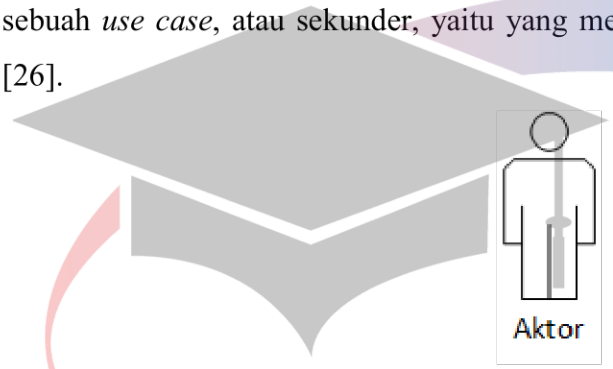
1. *Primary Actor*, yaitu aktor yang menginisiasi layanan sistem untuk mencapai tujuan dari aktor tersebut. Jumlah aktor primer dimungkinkan lebih dari 1.
2. *Preconditions*, yaitu kondisi spesifik yang harus terpenuhi sebelum sebuah UC bisa diinisiasi atau dieksekusi oleh aktor primer. Jumlah prakondisi bisa lebih dari 1 keadaan.
3. *Main or Basic Flow*, yaitu jalur interaksi yang mengarahkan pada skenario yang berhasil sehingga tujuan aktor bisa terpenuhi. Jalur ini hanya terdiri dari 1 jalur saja.
4. *Alternative Flows*, yaitu jalur alternatif dari interaksi yang terjadi antar aktor dengan sistem yang mencakup pencabangan (pilihan) maupun skenario yang gagal sehingga tujuan aktor tidak terpenuhi. Jalur ini bisa terdiri dari lebih dari 1 jalur kemungkinan.
5. *Postconditions*, yaitu kondisi spesifik yang harus terjadi ketika *use case* berhasil dijalankan atau dieksekusi secara lengkap, sebagai representasi dari tujuan yang ingin dicapai oleh aktor primer. Jumlah kondisi akhir bisa lebih dari 1 keadaan.

1.6.1.2 *Use Case Diagram*

Sebuah *use case diagram* menyatakan visualisasi interaksi yang terjadi antara pengguna (aktor) dengan sistem. Diagram ini bisa menjadi gambaran yang bagus untuk

menjelaskan konteks dari sebuah sistem sehingga terlihat jelas batasan dari sistem. Ada 2 elemen penting yang harus digambarkan, yaitu aktor dan *use case* [26].

Aktor adalah segala sesuatu yang berinteraksi langsung dengan sistem, bisa merupakan orang (yang ditunjukkan dengan perannya dan bukan namanya atau personilnya) atau sistem komputer yang lain. Aktor dinotasikan dengan simbol gambar orang-orangan (*stick-man*) dengan nama kata benda di bagian bawah yang menyatakan peran atau sistem. Aktor bisa bersifat primer, yaitu yang menginisiasi berjalannya sebuah *use case*, atau sekunder, yaitu yang membantu berjalannya sebuah *use case* [26].



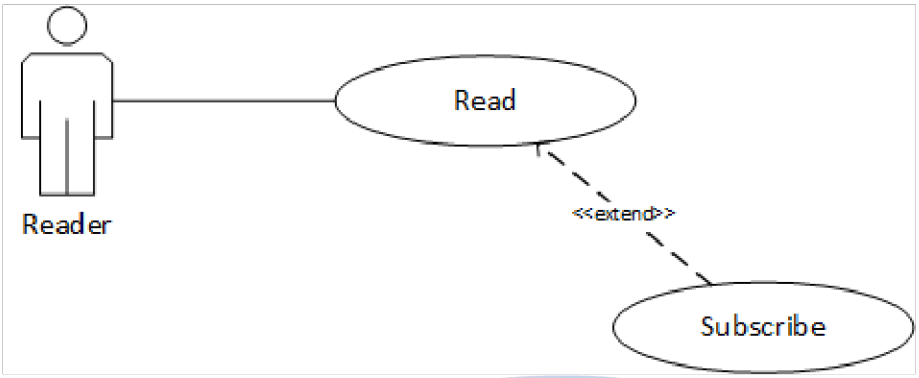
Gambar 1.5 Aktor

Use case dinotasikan dengan simbol elips dengan nama kata kerja aktif di bagian dalam yang menyatakan aktivitas dari perspektif aktor. Setiap aktor dimungkinkan untuk berinteraksi dengan sistem dalam banyak *use case*. Sebaliknya, setiap *use case* bisa dijalankan oleh lebih dari satu aktor [26].

Gambar 1.6 *Use Case*

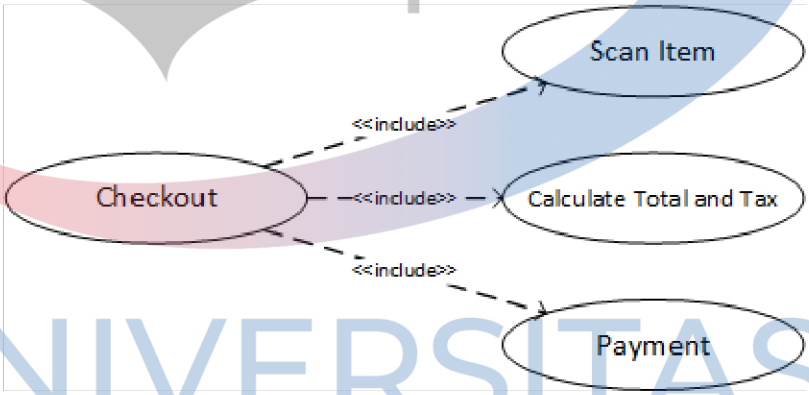
Antar aktor maupun antar *use case* bisa memiliki relasi, masing-masing dengan spesifikasi yang berbeda. Sebuah *use case*, disebut dengan *base use case*, bisa memiliki relasi dengan 1 atau lebih *use case* yang lain, disebut dengan *supplier use case*, dalam bentuk **extend** dan/atau **include** [26].

Relasi **extend** menyatakan bahwa fungsionalitas dari *base use case* bisa diperluas oleh *supplier use case*, jika dibutuhkan, di dalam eksekusi alur alternatif yang ada pada *use case scenario* dari *base use case* [26].



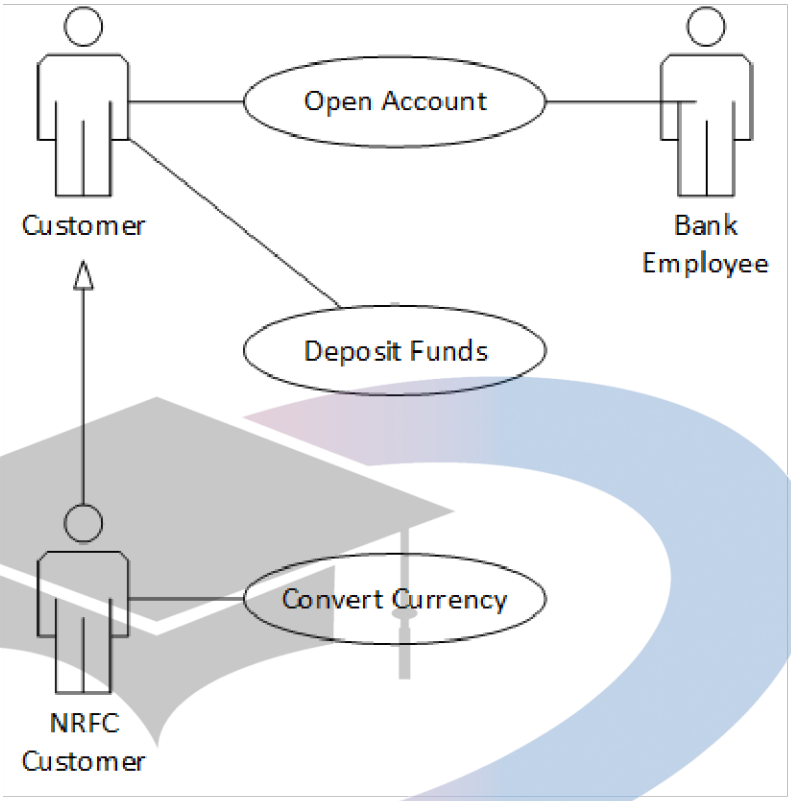
Gambar 1.7 Relasi *Extend*

Relasi **include** menyatakan bahwa fungsionalitas dari *base use case* selalu hanya bisa dipenuhi dengan bantuan dari *supplier use case* di dalam eksekusi alur utama yang ada pada *use case scenario* dari *base use case* [26].



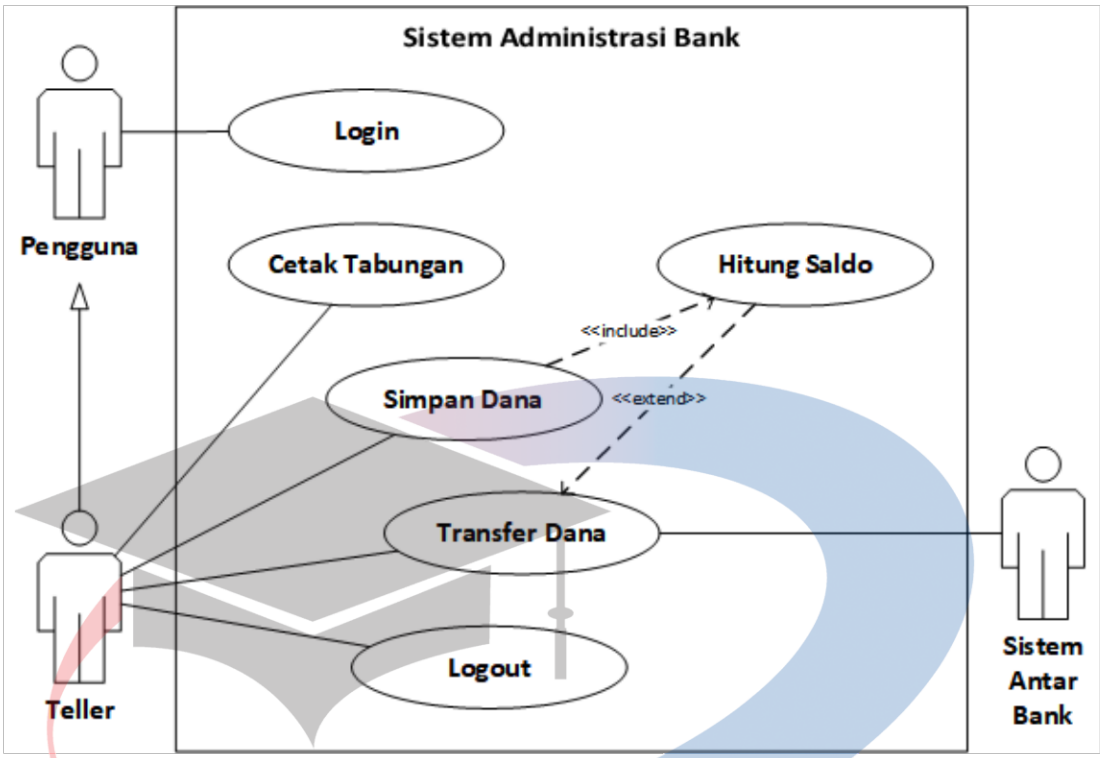
Gambar 1.8 Relasi *Include*

Dalam hal ini, relasi **include** dan **extend** tidak menjelaskan urutan eksekusi apapun antara *base use case* dan *supplier use case*, baik dalam alur utama maupun alternatif yang dijelaskan dalam *use case scenario* dari *base use case*. Selanjutnya, sebuah aktor, disebut aktor induk, bisa memiliki relasi **inheritance** dengan aktor yang lain, disebut aktor turunan, yang menyatakan bahwa sebuah aktor merupakan turunan dari aktor yang lain. Aktor turunan akan memiliki hak akses terhadap fungsionalitas sistem yang lebih luas dibandingkan dengan aktor induk [26].



Gambar 1.9 Relasi *Inheritance*

Gambar 2.10 mengilustrasikan sebuah *use case diagram* sederhana yang terdiri dari 2 buah aktor primer, yaitu Pengguna dan Teller, serta 1 aktor sekunder, yaitu Sistem Antar Bank. Sistem tersebut memiliki fungsionalitas yang direpresentasikan dalam 5 *use case*, yaitu Login, Cetak Tabungan, Simpan Dana, Transfer Dana, dan Hitung Saldo. Aktor Teller merupakan turunan dari aktor Pengguna, ketika Pengguna dinyatakan valid saat melakukan *login*. Siapapun bisa melakukan *login*, itulah disebut dengan Pengguna. Teller, saat mentransfer dana, **bisa** melibatkan proses perhitungan saldo jika transfer yang dilakukan menggunakan rekening nasabah (ditunjukkan dengan relasi ***extend***). Di sisi lain, perhitungan saldo **harus** selalu dilakukan saat Teller melakukan penyimpanan dana di rekening nasabah (ditunjukkan dengan relasi ***include***). Untuk mengakhiri akses, Teller melakukan *logout* [26].



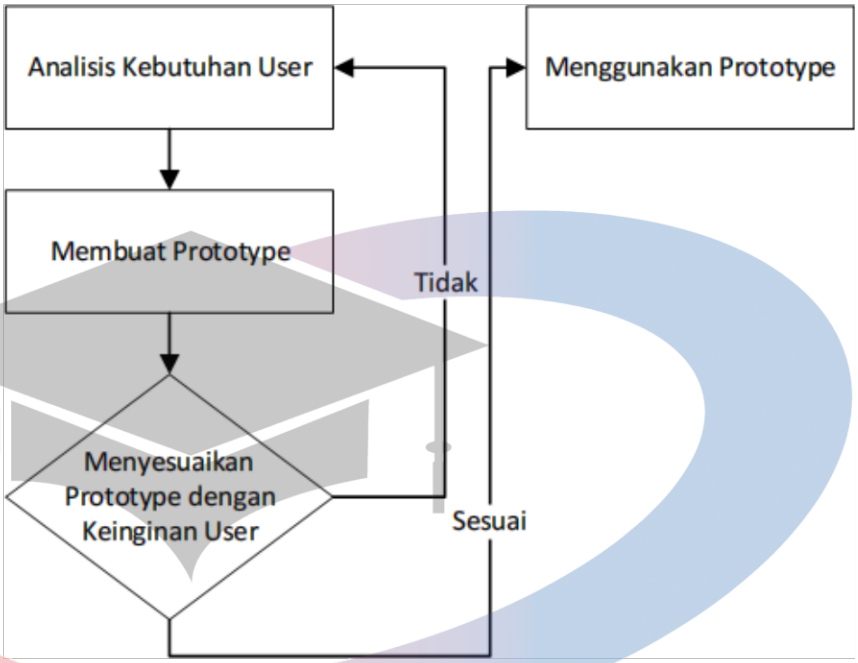
Gambar 1.10 Sebuah *Use Case Diagram* sederhana dari Sistem Administrasi Bank (parsial)

1.6.2 *Prototyping*

Prototyping merupakan teknik pengembangan sistem yang menggunakan *prototype* untuk menggambarkan sistem, sehingga pengguna atau pemilik sistem mempunyai gambaran pengembangan sistem yang akan dilakukannya. Teknik ini sering digunakan apabila pemilik sistem tidak terlalu menguasai sistem yang akan dikembangkannya, sehingga dia memerlukan gambaran dari sistem yang akan dikembangkannya tersebut. Dengan teknik *prototyping*, pengembang bisa membuat *prototype* terlebih dahulu sebelum mengembangkan sistem yang sebenarnya [27]. Dalam pengembangan sistem informasi, *prototype* sering diwujudkan dalam bentuk *user interface* program aplikasi dan contoh-contoh reporting yang akan dihasilkan, sehingga dengan demikian pengguna sistem akan mempunyai gambaran tentang sistem yang akan digunakannya nanti. McLeod dan Schell mendefinisikan 2 tipe dari *prototype* yaitu [27]:

1. *Evolutionary Prototype*

Merupakan *prototype* yang secara terus-menerus dikembangkan hingga *prototype* tersebut memenuhi fungsi dan prosedur yang dibutuhkan oleh sistem. Berikut gambar dari tahapan *evolutionary prototype*:



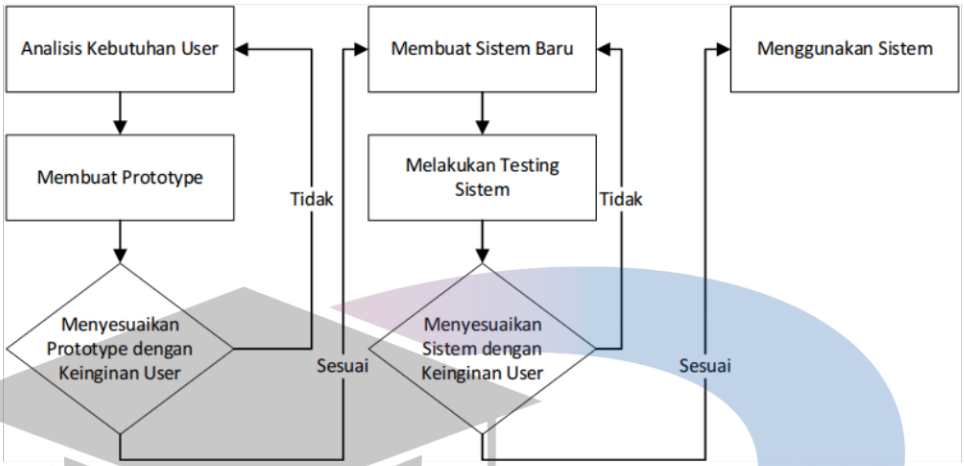
Gambar 1.11 Tahapan Langkah *Evolutionary Prototype*

a. Analisis Kebutuhan *User*, pengembang dan pengguna atau pemilik sistem melakukan diskusi dimana pengguna atau pemilik sistem menjelaskan kepada pengembang tentang kebutuhan sistem yang mereka inginkan.
b. Membuat *Prototype*, pengembang membuat *prototype* dari sistem yang telah dijelaskan oleh pengguna atau pemilik sistem.
c. Menyesuaikan *Prototype* dengan Keinginan *User*, pengembang menanyakan kepada pengguna atau pemilik sistem tentang *prototype* yang sudah dibuat, apakah sesuai atau tidak dengan kebutuhan sistem.
d. Menggunakan *Prototype*, sistem mulai dikembangkan dengan *prototype* yang sudah dibuat.

2. *Requirements Prototype*

Merupakan *prototype* yang dibuat oleh pengembang dengan mendefinisikan fungsi dan prosedur sistem dimana pengguna atau pemilik sistem tidak bisa

mendefinisikan sistem tersebut. Berikut gambar dari tahapan *requirement prototype*:



Gambar 1.12 Tahapan Langkah *Requirements Prototype*

a. Analisis Kebutuhan *User*, pengembang dan pengguna atau pemilik sistem melakukan diskusi dimana pengguna atau pemilik sistem menjelaskan kepada pengembang tentang kebutuhan sistem yang mereka inginkan.
b. Membuat *Prototype*, pengembang membuat *prototype* dari sistem yang telah dijelaskan oleh pengguna atau pemilik sistem.
c. Menyesuaikan *Prototype* dengan Keinginan *User*, pengembang menanyakan kepada pengguna atau pemilik sistem tentang *prototype* yang sudah dibuat, apakah sesuai atau tidak dengan kebutuhan sistem.
d. Membuat Sistem Baru, pengembang menggunakan *prototype* yang sudah dibuat untuk membuat sistem baru.
e. Melakukan *Testing* Sistem, pengguna atau pemilik sistem melakukan uji coba terhadap sistem yang dikembangkan.
f. Menyesuaikan dengan Keinginan *User*, sistem disesuaikan dengan keinginan *user* dan kebutuhan sistem, jika sudah sesuai sistem siap digunakan.
g. Menggunakan Sistem.

Kelebihan dari teknik pengembangan *prototyping* adalah proses pengembangan yang menghemat waktu dan biaya. Pengguna atau pemilik sistem ikut terlibat dalam pengembangan, sehingga kemungkinan-kemungkinan terjadinya kesalahpahaman dalam sistem bisa diminimalisir. Implementasi akan menjadi mudah,

karena pengguna atau pemilik sistem sudah mempunyai gambaran tentang sistem. Kualitas sistem yang dihasilkan baik, dan memungkinkan tim pengembang sistem memprediksi dan memperkirakan pengembang-pengembang sistem selanjutnya [27].

Sedangkan kelemahannya, pengguna atau pemilik sistem bisa terus-menerus menambah kompleksitas sistem hingga sistem menjadi sangat kompleks, hal ini bisa menyebabkan pengembang meninggalkan pekerjaannya sehingga sistem yang dikerjakan tidak akan pernah terselesaikan [27].

1.7 *Point of Sale* (POS)

Point of Sale secara umum dapat diartikan sebagai sebuah sistem yang memungkinkan diadakannya proses transaksi. POS dapat digunakan di semua transaksi penjualan seperti restoran, supermarket, hotel, dan toko-toko retail. Karena itu, POS juga dapat diartikan sebagai proses pelayanan transaksi dalam sebuah toko retail. Dari semua pengertian yang dijelaskan tersebut, maka dapat diambil kesimpulan bahwa *point of sale* dapat diartikan sebagai sebuah sistem yang memungkinkan diadakannya transaksi yang di dalamnya termasuk juga penggunaan mesin kasir [28].

Aplikasi *point of sale* adalah perangkat lunak yang banyak digunakan pada usaha retail seperti swalayan, minimarket, apotek, *cafe*, dan lain-lain [28]. Ada beberapa keuntungan jika memakai aplikasi *point of sale*, antara lain peningkatan kualitas layanan, dengan adanya POS, maka perusahaan akan dengan mudah dalam menjalankan proses transaksi yang tepat, cepat dan sistematis. Keuntungan lainnya yaitu memudahkan proses mengontrol dan mengambil keputusan. Biasanya, proses *controlling* dapat dengan mudah dilakukan sebab semua laporan dapat tersedia dengan cepat, sehingga memudahkan proses pengambilan keputusan baik secara kolektif maupun personal [29].

Bisa disimpulkan bahwa *point of sale* pada warung kopi yaitu sebuah sistem proses transaksi yang mencakup pemesanan, pembayaran, dan pembukuan dalam aplikasi untuk memudahkan proses transaksi tersebut, dengan tujuan untuk pembuatan laporan yang cepat dan selalu tersedia, sehingga proses pengambilan keputusan dapat lebih mudah untuk dilakukan.

1.7.1 Pemesanan

Pemesanan adalah proses pengolahan pesanan atau order melibatkan penyiapan pesanan untuk pengiriman dan penerimaan pesanan ketika pengiriman-pengiriman tiba. Proses pemesanan meliputi sejumlah kegiatan, seperti memeriksa kredit pelanggan, pencatatan penjualan, membuat catatan akuntansi yang sesuai, mengatur barang yang akan dikirim, penyesuaian catatan persediaan, dan tagihan pelanggan. Pemesanan merupakan suatu aktivitas yang dilakukan oleh konsumen sebelum membeli. Dari penjelasan tentang pemesanan yang ada, maka didapatkan kesimpulan bahwa pemesanan merupakan proses atau cara memesan baik berupa barang maupun jasa yang dilakukan antara dua pihak atau lebih [30]. Bisa disimpulkan bahwa pemesanan pada warung kopi merupakan sebuah aktivitas konsumen melakukan pemesanan menu pesanan yang telah disediakan oleh warung kopi.

Proses pemesanan yang sudah sangat umum dilakukan adalah sistem pemesanan secara manual. Pada sistem ini, pelayan harus menghampiri meja pelanggan untuk menyerahkan menu pesanan, kemudian pelayan tersebut mencatat menu makanan dan minuman yang dipesan oleh pelanggan, dan menyerahkan catatan tersebut ke dapur. Proses pemesanan juga dapat dilakukan secara langsung oleh pelayan dengan memberikan menu pesanan dan membiarkan pelanggan untuk menulis pesanannya di kertas [31].

1.7.2 Pembayaran

Pembayaran adalah proses memindahkan dana dari satu pihak ke pihak lainnya. Pembayaran berhubungan dengan proses pembayaran akan sesuatu seperti jasa, barang, tagihan, dan lainnya [32]. Bisa disimpulkan bahwa proses pembayaran pada warung kopi merupakan aktivitas konsumen melakukan pembayaran menu pesanan yang telah dipesan pada warung kopi secara tunai.

Pembayaran tunai merupakan pembayaran atas harga barang atau jasa secara tunai, dimana pihak pembeli menyerahkan uang sebagai bukti pembayaran sebesar harga barang atau jasa yang dibeli bersamaan dengan surat pesanan. Pembayaran tunai ini biasanya dilakukan dengan menggunakan uang tunai. Alat pembayaran tunai adalah uang kartal yang terdiri dari uang kertas dan uang logam [33].

1.7.3 Pembukuan

Pembukuan adalah pencatatan transaksi keuangan. Transaksi meliputi penjualan, pembelian, pendapatan, dan pengeluaran oleh perseorangan ataupun organisasi dan entitas bisnis (perusahaan) [34]. Pembukuan juga tidak hanya dilakukan oleh perusahaan yang berskala besar, namun juga perlu diterapkan pada perusahaan yang masih berskala kecil sehingga dapat memanajemen keuangan dengan baik [34]. Pembukuan pada dasarnya adalah perekaman atau pencatatan semua informasi mengenai transaksi dan kegiatan keuangan dari pebisnis tentang proses akuntansi mereka. Hasil dari proses akuntansi berupa pelaporan keuangan atau pelaporan akuntansi sebagai bentuk informasi keuangan kepada pihak-pihak yang membutuhkan [35].

Dengan pencatatan keuangan atau penyelenggaraan pembukuan yang sesuai dengan aturan, maka pelaku usaha dapat mengetahui perkembangan usahanya, dapat memberikan informasi yang berguna bagi pengambilan keputusan usahanya. Adapun beberapa manfaat dari pencatatan keuangan atau penyelenggaraan pembukuan, yaitu [36]:

1. Mengontrol pemanfaatan uang yang telah dibelanjakan sehingga manajemen dapat mengendalikan biaya dan pengeluaran kas sesuai dengan tujuan usaha.
2. Menganalisis sumber penghasilan.
3. Merencanakan aliran kas.
4. Melindungi uang usaha yang ada, karena terdapat catatan terperinci mengenai saldo uang usaha dan seluruh transaksi usahanya.
5. Mengetahui perubahan aset usaha, penambahan atau pengurangan utang usaha serta penambahan ataupun pengurangan modal usaha.
6. Mengetahui jenis aset usaha, jenis utang usaha dan jenis modal usaha yang dimiliki.
7. Mengetahui perkembangan usaha.
8. Dapat membantu dalam pengambilan keputusan yang lebih baik.
9. Meningkatkan kepercayaan investor karena dengan data yang akurat, investor ataupun pihak bank akan lebih percaya dengan usaha yang dikelola.