BAB II
KAJIAN LITERATUR

2.1 Pengertian Sistem

Dalam konsep dasar sistem, istilah "sistem" merujuk pada sesuatu yang terdiri dari komponen-komponen yang saling berkaitan dan bekerjasama untuk mencapai tujuan tertentu. Entitas yang terbuat dari elemen atau komponen yang saling terhubung. Hal ini bertujuan untuk mencapai tujuan tertentu yang bersifat terikat dan terbatas. Antara setiap elemen atau komponen tidak akan saling bertentangan atau berlawanan, karena semuanya saling bergantung dan saling membutuhkan satu sama lain untuk mencapai tujuan tertentu [1].

2.2 Sistem Informasi

Sistem informasi adalah kombinasi terorganisir dari orang-orang, perangkat keras (*hardware*), perangkat lunak (*software*), jaringan komunikasi, dan sumber daya data yang digunakan untuk menintegrasikan, merubah, dan menyebarkan informasi dalam sebuah perusahaan [2]. Sistem informasi memiliki peranan penting dalam suatu organisasi atau perusahaan, semakin berkembangnya suatu perusahaan semakin penting pula peran yang dimiliki sistem informasi.

Sistem informasi memiliki komponen-komponen diantaranya adalah [3]:

1. *Input*

   Input adalah data yang dimasukkan kedalam sistem informasi untuk diproses. Contohnya, ketika ingin melacak pengiriman barang melalui kurir, *user* harus memasukkan nomor resi terlebih dahulu. Nomor resi itulah yang dsebut sebagai *Input*.

2. Prosedur atau proses

   Input atau data yang dimasukkan kedalam sistem akan diproses dengan prosedur tertentu untuk menghasilkan *output* yang diinginkan. Prosedur dapat berupa algoritma atau langkah-langkah, logika, model matematika yang tentunya sudah diimplementasikan oleh pengembang sistem.

3. *Output*

   Setelah *Input* selesai diproses, sistem akan menghasilkan *output* yang diinginkan oleh pemakai sistem. *Output* tersebut berupa informasi yang bermanfaat yang dapat membantu pemakai sistem dalam mengambil keputusan.

4. *Komponen Kontrol*

   Kontrol pengendalian yang dimaksud adalah pengendalian yang dirancang untuk menanggulangi gangguan terhadap sistem informasi. Gangguan tersebut bisa berupa bencana, *human error*, serangan *hacker*, dan lain-lain. Dengan adanya pemeliharaan, sistem informasi akan berjalan dengan lancar dan dapat memberikan informasi yang sesuai untuk pemakai sistem.

2.2.1 Perkembangan Sistem Informasi

Sistem informasi telah mengalami perkembangan yang pesat sejak pertama kali diperkenalkan pada pertengahan abad ke-20. Berikut adalah beberapa tahap utama dalam evolusi sistem informasi [4]:

1. Era Komputerisasi Awal (1950-an sampai 1960-an): Pada masa ini, sistem informasi pertama kali diimplementasikan untuk tujuan otomasi proses manual, seperti pengolahan data akuntansi dan penggajian. Komputer digunakan untuk menjalankan tugas-tugas rutin secara lebih cepat dan efisien.
2. Era Sistem informasi Manajemen (1970-an sampai 1980-an): Seiring dengan perkembangan teknologi komputer, sistem informasi berkembang menjadi alat untuk mendukung manajemen dalam membuat keputusan strategis. Sistem informasi Manajemen (SIM) mulai diterapkan untuk membantu pengelolaan data operasional dan strategis dalam organisasi.
3. Era Sistem Pendukung Keputusan (1980-an sampai 1990-an): Pada tahap ini, sistem informasi berkembang menjadi alat yang lebih canggih, yang tidak hanya mengelola data, tetapi juga menganalisis informasi secara lebih mendalam. Sistem Pendukung Keputusan *(Decision Support Systems)* mulai diterapkan untuk membantu pengambilan keputusan yang kompleks dengan menggunakan data historis dan simulasi.
4. Era Internet dan *E-business* (1990-an sampai 2000-an): Perkembangan internet membawa perubahan besar dalam sistem informasi. Sistem informasi mulai diintegrasikan dengan jaringan global, memungkinkan *e-business* dan *e-commerce* berkembang pesat. Data *real time* dan analisis berbasis *web* mulai diimplementasikan dalam berbagai sektor.
5. Era *Big Data* dan AI (2000-an – Sekarang): Saat ini, sistem informasi mengandalkan kemampuan analisis data besar (*Big Data*) dan kecerdasan buatan *(Artificial Intelligence)* untuk memberikan wawasan yang lebih mendalam dan prediktif. Sistem

informasi tidak hanya mendukung pengambilan keputusan, tetapi juga mampu belajar dari data untuk meningkatkan akurasi dan relevansi Informasi yang diberikan.

2.3 Produksi

Produksi merupakan kegiatan yang mentrans*form*asikan masukan (*Input*) menjadi keluaran (*output*), tercakup semua aktivitas atau kegiatan yang menghasilkan barang atau jasa, serta kegiatan-kegiatan lain yang mendukung atau menunjang usaha untuk menghasilkan produk tersebut yang berupa barang atau jasa. Didalam produksi terdapat 5 komponen utama yang saling berinteraksi dan terkait disebut sebagai 5M yang terdiri dari *Man, Money, Material, Machine* dan *Methode*.

Manusia (*Man*) yang merujuk pada sumber daya manusia (SDM) yang dimiliki suatu organisasi atau badan usaha, uang (*Money*) yang merujuk pada pendanaan dan perhitungan rasional dalam tiap proses produksi, bahan produksi (*Material*) bahan-bahan yang digunakan untuk mencapai tujuan produksi, mesin (*Machine*) adalah tekonologi, infrastuktur dan berbagai alat yang digunakan untuk mempermudah dan menciptakan efisiensi produksi guna mencapai keuntungan yang lebih tinggi dan metode (*Methode*) adalah cara pelaksanaan suatu tugas dalam produksi dengan pertimbangan terhadap keempat komponen sebelumnya [5].

Dalam produksi juga terdapat beberapa tahapan yang dilakukan secara sistematis untuk mengubah bahan baku menjadi barang jadi yang memiliki nilai guna. Tahapan ini meliputi perencanaan (*Planning)*, penyusunan rute (*Routing*), penjadwalan (*Scheduling*), pengiriman (*Dispatching*) dan peninjauan ulang *(Follow Up*).

Perencanaan (*Planning*) adalah tahapan dimana organisasi menentukan sumber daya yang diperlukan untuk menghasilkan *output* tertentu. Setelah menentukan sumber daya, organisasi mengatur rute (*Routing*) yang menentukan urutan pekerjaan dan siapa yang bertanggung jawab dalam tiap-tiap alur pekerjaan. Penjadwalan (*Scheduling*) dibuat setelah rute untuk menjadwalkan durasi operasi dalam tiap produksi yang dijalankan serta kapan pekerja harus mulai mengerjakan tugasnya. Setelah tahapan diatas selesai dan menghasilkan barang jadi, diakukan pengiriman barang (*Dispatching*) untuk mendistribusikan barang jadi kepada konsumen. Tahapan terakhir yaitu peninjauan ulang (*Follow Up*) yang bertujuan untuk mengevaluasi kembali hasil dari proses produksi yang terjadi dengan perencanaan yang dibuat [5].

Proses produksi memiliki jenis yang beragam namun dapat didefnisikan kedalam 2 jenis berbeda yaitu Proses Produksi Terus Menerus (*Continuous Process*) dan Proses

Produksi Putus-Putus (*Intermitten Process*). *Continous Process* adalah jenis proses produksi yang memiliki pola dan urutan yang pasti dan tidak berubah-ubah dalam pelaksanaan produksinya. Sedangkan *Intermitten Process* adalah jenis proses produksi yang memiliki beberapa pola ataupun urutan yang berbeda dalam prosesnya sehingga menghasilkan variasi produksi yang banyak [6].

2.3.1 Sistem Produksi

Setelah mengetahui pengertian sistem dan produksi, maka penting untuk mengetahui pengertian sistem produksi. Sistem produksi adalah suatu gabungan dari beberapa unit atau elemen yang saling berhubungan dan saing menunjang untuk melaksanakan proses produksi dalam suatu perusahaan tertentu [7]. Sistem produksi memiliki berbagai komponen didalamnya seperti komponen struktural yang terdiri dari bahan, peralatan, tenaga kerja dan sebagainya dan komponen fungsional yang terdiri dari perencanaan, pengendalian dan pengawasan manajemen dan operasional.

Pada umumnya terdapat tiga kategori sistem produksi yang berkembang pada saat ini, yaitu [8]:

1. Sistem Produksi *Push*

   Sistem produksi *push* adalah sistem produksi yang menerapkan pendekatan perencanaan yang dikendalikan secara terpusat. Perencanaan tersebut memiliki dampak mulai dari tahapan awal hingga tahapan akhir dalam suatu proses produksi. Produksi dalam sistem ini didorong oleh keyakinan bahwa permintaan di masa depan dapat diprediksi berdasarkan data pemesanan masa lalu atau melalui metode peramalan permintaan. Sistem produksi *push* dapat dijelaskan sebagai sistem perencanaan yang berorientasi dari atas ke bawah, karena semua keputusan mengenai jumlah produksi berasal dari perkiraan permintaan yang telah ditentukan dalam *Master Production Schedule (MPS),* dan kemudian dijalankan melalui *Material Resource Planning (MRP).* Karakteristik sistem ini dapat mempercepat waktu pengiriman karena sudah tersedianya komponen *Work In Proccess (WIP)*. Ketersediaan komponen *work in process* (*WIP*) serta produk jadi terjadi karena produksi berjalan sesuai dengan jadwal dan tidak terganggu, Namun, hal tersebut menyebabkan peningkatan volume produksi, baik dalam bentuk produk yang belum selesai diproses maupun produk yang sudah siap digunakan. Akibatnya, sistem mengalami persediaan yang berlebihan, sehingga mengakibatkan peningkatan biaya penyimpanan.

2. Sistem Produksi *Pull*

Sistem produksi *pull* memulai proses produksinya berdasarkan sinyal selesai dari tahap berikutnya dalam suatu sistem. Produksi dalam sistem ini akan mengirimkan sinyal ke proses produksi sebelumnya agar mengambil produk tersebut yang diperlukan dari hasil proses produksi sebelumnya hanya dalam jumlah dan waktu yang dibutuhkan, serta sinyal ini juga memicu terjadinya produksi guna mengisi kembali stok barang yang telah digunakan atau diambil untuk kebutuhan proses berikutnya. Sistem produksi *pull* menghasilkan produk berdasarkan permintaan nyata dari pelanggan. Saat permintaan pelanggan muncul, maka akan segera dipenuhi dengan persediaan produk jadi. Persediaan produk jadi disediakan terlebih dahulu, dan ketika persediaan produk jadi tersebut semakin habis, maka sinyal kanban akan dikeluarkan untuk memicu proses produksi di unit produksi sebelumnya, sehingga persediaan produk jadi kembali tersedia. Proses tersebut akan terus berlanjut hingga ke proses produksi awal. Oleh karena itu, sistem produksi *pull* memiliki tingkat inventori barang jadi maupun barang setengah jadi yang rendah, karena produksi dilakukan secara terbatas berdasarkan kebutuhan aktual yang terjadi. Namun, sistem produksi *pull* sering menghadapi tantangan karena berpotensi menyebabkan kelangkaan bahan baku setengah jadi apabila diterapkan dalam kondisi permintaan yang bervariasi sedang hingga tinggi, sehingga dibutuhkan mekanisme backordering yang cukup signifikan.

3. Sistem Produksi *Hybrid*

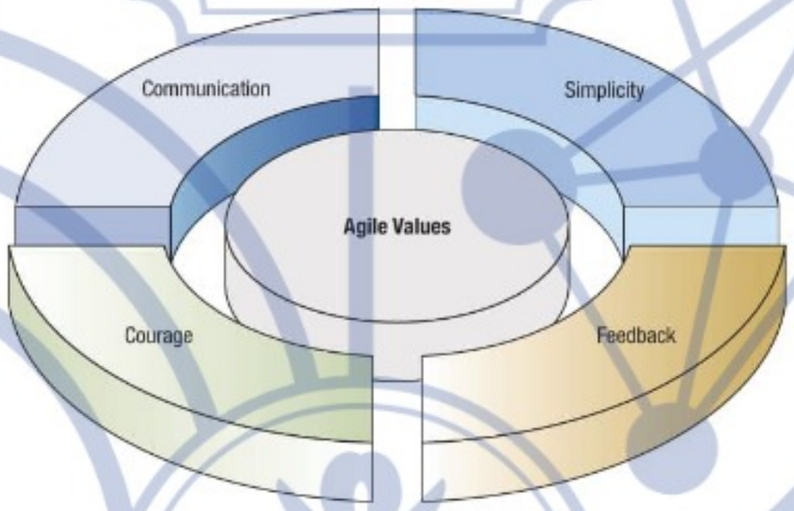
   Sistem ini menggabungkan kedua metode produksi *pull* dan *push* agar dapat memanfaatkan kelebihan masing-masing sistem tersebut dan mengurangi kelemahan yang ada, sehingga dapat mencapai kinerja produksi yang seimbang dan maksimal. Sistem *hybrid* mampu menggabungkan kedua cara kerja sistem *push* dan *pull*, sehingga dapat memanfaatkan kelebihan masing-masing sistem dan mengurangi kelemahan yang ada. Sistem *hybrid* muncul karena kedua sistem sebelumnya yaitu *push* dan *pull* masing-masing memiliki kelebihan dan kekurangan, sehingga sistem ini bertujuan untuk menghilangkan kelemahan tersebut dan memperkuat keunggulan dengan menggabungkan kedua sistem tersebut. Dalam sistem produksi *hybrid*, proses produksi dimulai dengan model produksi *push*, kemudian terdapat titik peralihan di mana sistem produksi berpindah ke model sistem *pull*. Seperti yang telah dijelaskan sebelumnya, titik tersebut merupakan titik persimpangan yang disebut sebagai *junction point* (*JP*). Perintah produksi awal dikeluarkan berdasarkan perkiraan permintaan bulanan hingga barang tersebut memasuki persediaan aman dalam bentuk barang setengah jadi atau *semi-finished products* pada *JP*. Setelah titik produksi *JP* tercapai, produksi akan

dilanjutkan lagi ketika tingkat stok aman barang jadi (*finished products*) menurun karena permintaan dari pelanggan. Hal tersebut akan memengaruhi proses produksi sebelumnya dengan menggunakan instruksi kanban untuk mengubah tingkat stok aman seperti model sistem produksi *pull*

2.4 Metode *Agile*

Metode *Agile* adalah pendekatan dalam pengembangan perangkat lunak yang menekankan pada nilai-nilai, prinsip, serta praktik utama untuk menghadapi perubahan kebutuhan informasi manusia yang terjadi dengan cepat dan dinamis. Berbeda dengan pendekatan tradisional yang tidak fleksibel, pendekatan *agile* bersifat interaktif dan bertahap, di mana sistem terus berkembang melalui serangkaian rilis yang singkat dan berkelanjutan. Karakteristik utamanya mencakup kemampuan untuk bersabar menghadapi perubahan kondisi bisnis dan lingkungan, serta keterlibatan aktif pelanggan atau pengguna sepanjang proses pengembangan.

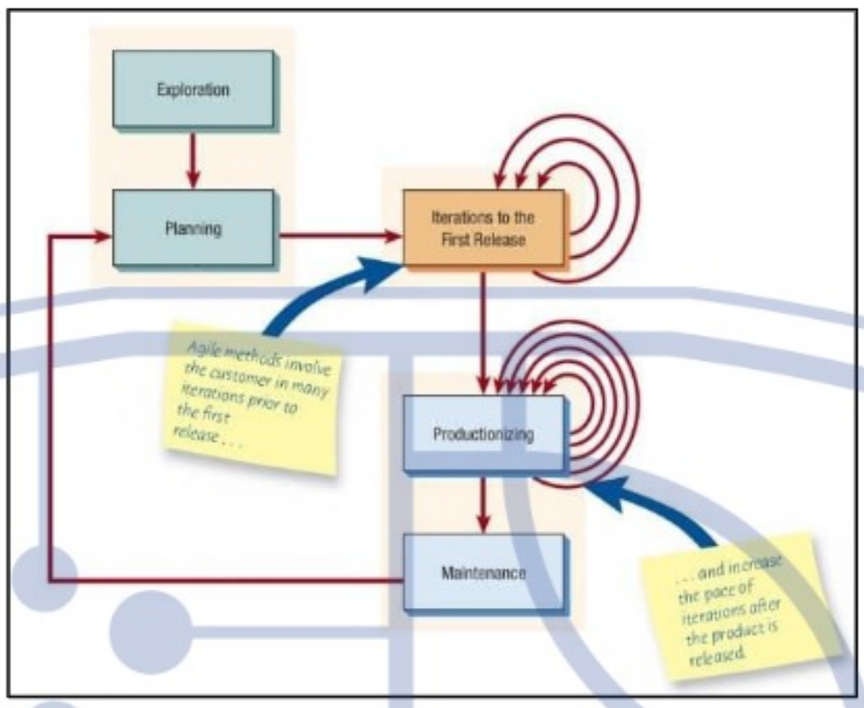
Agile didasarkan pada empat nilai utama yang membentuk lingkungan kerja kolaboratif yang efektif antara pengembang dan pelanggan, yaitu *communication, simplicity, feedback*, dan *courage* seperti yang tertera pada gambar dibawah ini [9].



Gambar 2.1 Empat Nilai Utama Metode Agile

1. *Communication:* Menekankan interaksi langsung agar dapat mencegah terjadinya kesalahpahaman dalam proses desain teknis dan pembaruan proyek.
2. *Simplicity:* Berfokus pada melakukan hal-hal yang paling mudah dan bisa dilakukan hari ini, dengan menyadari bahwa perubahan mungkin diperlukan di masa depan.
3. *Feedback:* Melibatkan respons cepat terhadap kode, jadwal, atau fungsi sistem dari anggota tim serta pelanggan.

4. *Courage:* Kemampuan tim untuk melakukan eksperimen, menghilangkan kode yang tidak efektif, serta bertindak berdasarkan insting dan hasil pengujian untuk mendorong perbaikan secara berkelanjutan.



Gambar 2.2 Tahapan Metode *Agile*

Metode *agile* memiliki beberapa tahapan, yaitu [9]:

1. *Exploration* (Eksplorasi)

   Pada tahap ini, pengembang mengenali masalah yang muncul dan memutuskan apakah masalah tersebut dapat diatasi dengan pendekatan *agile*. Selain itu, dilakukan pemaparan mengenai teknologi dan anggota tim untuk memahami potensi teknologi serta belajar dalam menentukan estimasi waktu penyelesaian setiap tahapan berdasarkan sumber daya yang ada di lapangan.

2. *Planning* (Perencanaan)

   Perencanaan merupakan tahapan di mana terjadi kesepakatan antara pengembang dan pengguna akhir untuk menentukan jangka waktu pelaksanaan proyek. Biasanya, durasi waktu yang dibutuhkan berkisar antara 2 hingga 6 minggu, dengan fokus utama diberikan pada masalah terbesar yang dialami oleh pengguna. Kegiatan yang dilakukan mencakup wawancara, pengamatan, serta peninjauan dokumen-dokumen yang tersimpan. Kegiatan ini membutuhkan kerja sama yang kuat antara kedua pihak agar pembangunan sistem bisa selesai secara tepat waktu dan sesuai harapan.

3. *Iteration to the First Release* (Iterasi menuju Rilisan Pertama)

   Pada tahap ini dalam siklus pengujian, umpan balik dan perubahan atau disebut juga tahap iterasi berlangsung. Pengembang membuat gambar kasar yang mewakili seluruh desain arsitektur sistem tersebut. Pada tahap ini, juga dilakukan pengecekan terhadap fungsi yang telah ditulis oleh pelanggan, kemudian dilakukan evaluasi untuk

menentukan penyesuaian terhadap jadwal dan tugas yang dikerjakan agar tidak memberi beban yang berlebihan kepada tim.

4. *Productizing* (Produksi)

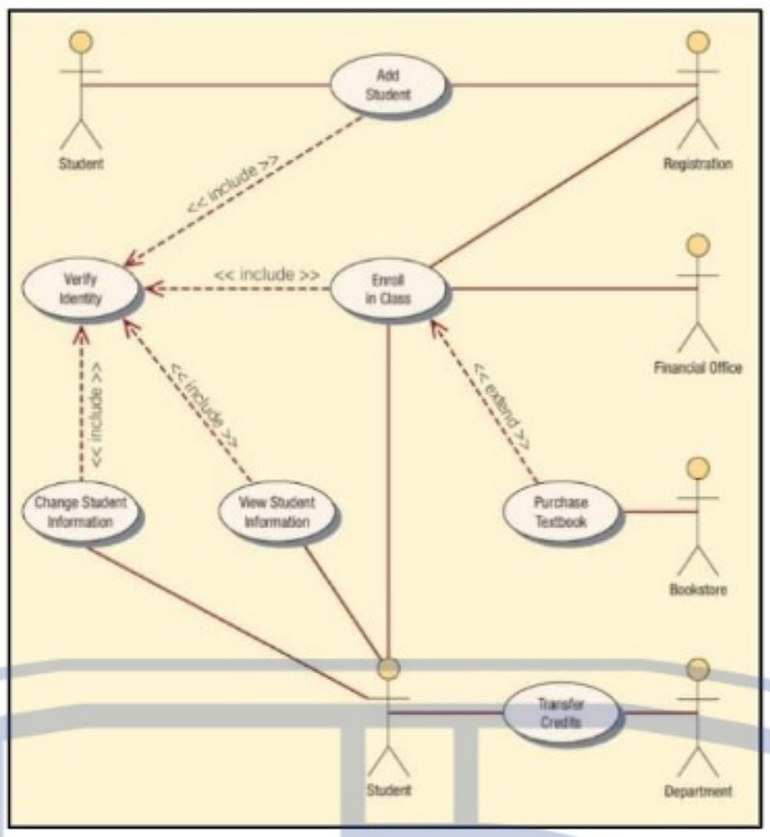
   Tahap ini merupakan tahap peluncuran produk, namun ada beberapa hal yang harus dilakukan sebelum dan setelah peluncuran berlangsung. Sebelum peluncuran, proses iterasi dipercepat dan tim melakukan *briefing* untuk memahami hambatan yang muncul mendekati waktu peluncuran. Setelah diluncurkan, produk terus dikembangkan dengan menambahkan fitur baru sesuai dengan kebutuhan yang muncul seiring berjalannya produk tersebut.

5. *Maintenance* (Pengelolaan)

   Pengelolaan merupakan proses menjaga sistem agar tetap berada dalam kondisi yang optimal. Pada tahap ini, sistem yang telah diresmikan masih menjalani proses pemeliharaan secara berkala agar *software* tetap beroperasi dan berfungsi dengan baik sesuai harapan. Penambahan atau perubahan fitur juga dilakukan pada tahapan ini untuk menyesuaikan dengan kebutuhan yang terus berubah, serta saran dari pengguna juga dipertimbangkan. Risiko yang mungkin tertunda juga ditangani pada tahapan ini, karena dilakukan secara berkelanjutan, ada kemungkinan terjadi perubahan dan pergantian anggota tim.

2.5 *Use Case* Diagram

Use case diagram awalnya diperkenalkan sebagai diagram untuk digunakan dalam Bahasa pemodelan terpadu berorientasi objek (*UML*), *use case* diagram kini diterapkan tanpa memandang pendekatan pengembangan sistem. *Use case* diagram dapat digunakan sebagai bagian dari pendekatan siklus hidup pengembangan sistem (*SDLC*) dalam metode *agile*. Sebuah *use case* menjelaskan apakah suatu sistem tidak memerlukan deskripsi, dan bagaimana sistem tersebut melakukannya. Artinya, ini adalah model logis dari suatu sistem. Dari persepktif aktor (pengguna), sebuah *use case* harus menghasilkan sesuatu yang bernilai. Oleh karena itu, analis harus menentukan apa yang penting bagi pengguna dan mengingat untuk memasukkannya kedalam *use case* diagram [9].



Gambar 2.3 Contoh *Use case* Diagram

2.5.1 Simbol *Use case*

Pada *use case* diagram terdapat simbol-simbol yang terdiri dari aktor, *use case*, dan *connecting line* yang akan dijelaskan dalam tabel dibawah ini [9]:

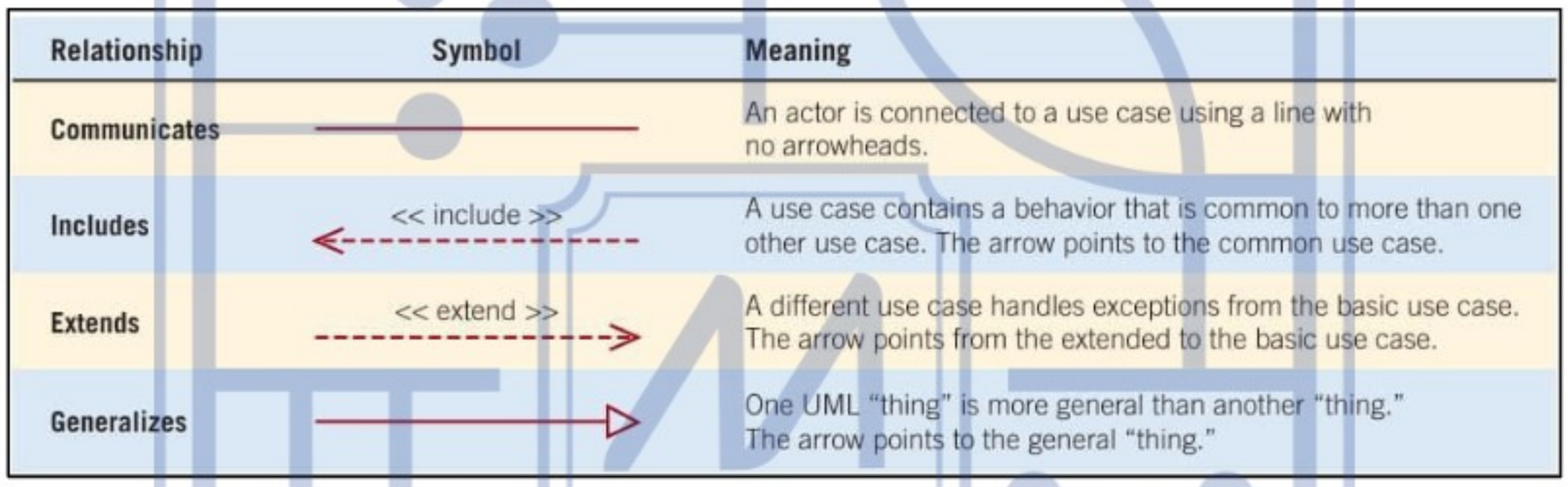
Tabel 2.1 Simbol *Use case*

Nama	Simbol	Keterangan
Aktor		Istilah "aktor" merujuk pada peran khusus pengguna sistem. Aktor berada di luar sistem dan berinteraksi dengan sistem dengan cara tertentu. Aktor dapat berupa manusia, sistem lain, atau perangkat seperti keyboard atau koneksi *web*. Seorang aktor dapat berinteraksi dengan satu atau lebih *use case*, dan sebuah *use case* dapat melibatkan satu atau lebih aktor.
Use case		*Use case* memberikan gambaran bagi pengembang tentang apa yang dibutuhkan pengguna. *Use case* merepresentasikan interaksi dan aktivitas antar aktor didalam sistem yang memicu serangkaian aksi untuk mencapai tujuan pengguna.

Connecting Line		*Connecting Line* menggambarkan jalur komunikasi antara aktor dan *use case*. Garis ini mendefinisikan keterlibatan pengguna didalam sistem untuk memicu sebuah peristiwa untuk mencapai tujuan.

2.5.2 Hubungan *Use case*

Hubungan aktif disebut sebagai hubungan perilaku dan terutama digunakan dalam *use case* diagram. Terdapat empat dasar hubungan perilaku yaitu: *Communication, Includes, Extends,* dan *Generalizes* [9].



Gambar 2.4 Hubungan *Use case*

1. *Communicates*

   Hubungan perilaku *communication* digunakan untuk menghubungkan seorang aktor dengan sebuah *use case*. Ingat bahwa tujuan dari sebuah *use case* adalah memberikan hasil yang berguna bagi aktor yang terlibat dalam sistem tersebut. Oleh karena itu, sangat penting untuk mencatat hubungan antara aktor dan *use case* tersebut.

2. *Includes*

   Hubungan *includes* menggambarkan situasi di mana sebuah *use case* memiliki perilaku yang sering digunakan oleh lebih dari satu *use case*. Dengan kata lain, *use case* yang umum tersebut terdapat dalam *use case* lainnya. Panah yang terputus yang menunjuk ke *use case* umum menunjukkan hubungan *includes.*

3. *Extends*

   Hubungan ini menunjukkan situasi di mana sebuah *use case* memiliki perilaku yang memungkinkan *use case* lain untuk menangani variasi atau pengecualian dari *use case* dasar.

4. *Generalizes*

Hubungan *generalizes* mengandung makna bahwa satu hal memiliki ciri yang lebih umum (mewakili contoh yang tipikal) dibandingkan dengan hal lainnya. Hubungan ini dapat muncul antara dua aktor atau dua *use case*.

2.5.3 *Use Case Scenario*

Setiap *use case* memiliki deskripsi, deskripsi tersebut disebut sebagai sebuah *use case scenario*. *Use case* utama mewakili alur kejadian standar dalam sistem, dan jalur alternatif menggambarkan variasi perilaku. *Use case scenario* merupakan alat yang digunakan untuk memahami serta mendokumentasikan interaksi antara pengguna atau aktor dengan sistem. *Use case scenario* membantu dalam mengidentifikasi kebutuhan fungsional sistem dengan cara yang mudah dipahami oleh pihak non-teknis, seperti pengguna bisnis dan pemangku kepentingan. *Use case scenario* dapat menjelaskan apa yang terjadi jika barang yang dibeli kehabisan stok, atau jika perusahaan kartu kredit menolak permintaan pembelian pelanggan [9].

Use case name:	Register for Conference	**UniqueID:** Conf RG 003
Area:	Conference Planning	
Actor(s):	Participant	
Stakeholder:	Conference Sponsor, Conference Speakers	
Level:	Blue	
Description:	Allow conference participant to register online for the conference using a secure website.	
Triggering Event:	Participant uses Conference Registration website, enters userID and password, and clicks the logon button.	
Trigger type:	External Temporal	

Steps Performed (Main Path)	**Information for Steps**
1. Participant logs in using the secure web server.	userID, Password
2. Participant record is read and password is verified.	Participant Record, userID, Password
3. Participant and session information is displayed on the Registration web page.	Participant Record, Session Record
4. Participant enters information on the Registration Web form and clicks **Submit** button.	Registration Web Form
5. Registration information is validated on the Web server.	Registration Web Form
6. Registration Confirmation page is displayed to confirm registration information.	Confirmation Web Page
7. Credit card is charged for registration fees.	Secure Credit Card Web Page
8. Add Registration Journal record is written.	Confirmation Web Page
9. Registration record is updated on the Registration Master.	Confirmation Web Page, Registration Record
10. Session record is updated for each selected session on the Session Master.	Confirmation Web Page, Session Record
11. Participant record is updated for the participant on the Participant Master.	Confirmation Web Page, Participant Record
12. Successful Registration Confirmation web page is sent to the participant.	Registration Record Confirmation Number

Preconditions:	Participant has already registered and has created a user account.
Postconditions:	Participant has successfully registered for the conference.
Assumptions:	Participant has a browser and a valid userID and password.
Success Guarantee:	Participant has registered for the conference and is enrolled in all selected sessions.
Minimum Guarantee:	Participant was able to logon.
Requirements Met:	Allow conference participants to be able to register for the conference using a secure website.
Outstanding Issues:	How should a rejected credit card be handled?
Priority:	High
Risk:	Medium

Gambar 2.5 Contoh *Use case scenario*

Pada *use case scenario* terdapat tiga area utama yaitu: *Header area* yang berisi identifikasi kasus dan pihak yang menginisiasi, langkah-langkah yang dilakukan, dan area *footer* yang berisi persyaratan, asumsi, pertanyaan, dan informasi lainnya [9].

1. *Case identifiers and initiators*: Area ini membantu pembaca memahami dan mengenali *use case* tersebut. Bagian ini memberikan orientasi awal mengenai *use case*.
2. *Steps performed:* Area ini berisi langkah langkah yang dilakukan serta informasi yang diperlukan untuk merepresentasikan alur peristiwa standar serta langkah langkah yang dilakukan untuk menyelesaikan suatu *use case.*
3. *A footer area that contains preconditions, assumptions, questions, and other information:* Area ini mencakup kondisi awal yang harus dipenuhi sebelum skenario dimulai, serta hasil akhir yang diharapkan setelah skenario selesai dilakukan. Bagian ini juga sering dipakai untuk menjelaskan pengecualian atau jalur lain dari perilaku utama.

2.5.4 Level *Use Case*

Pada *use case scenario* terdapat 5 level yang akan dijelaskan sebagai berikut [9]:

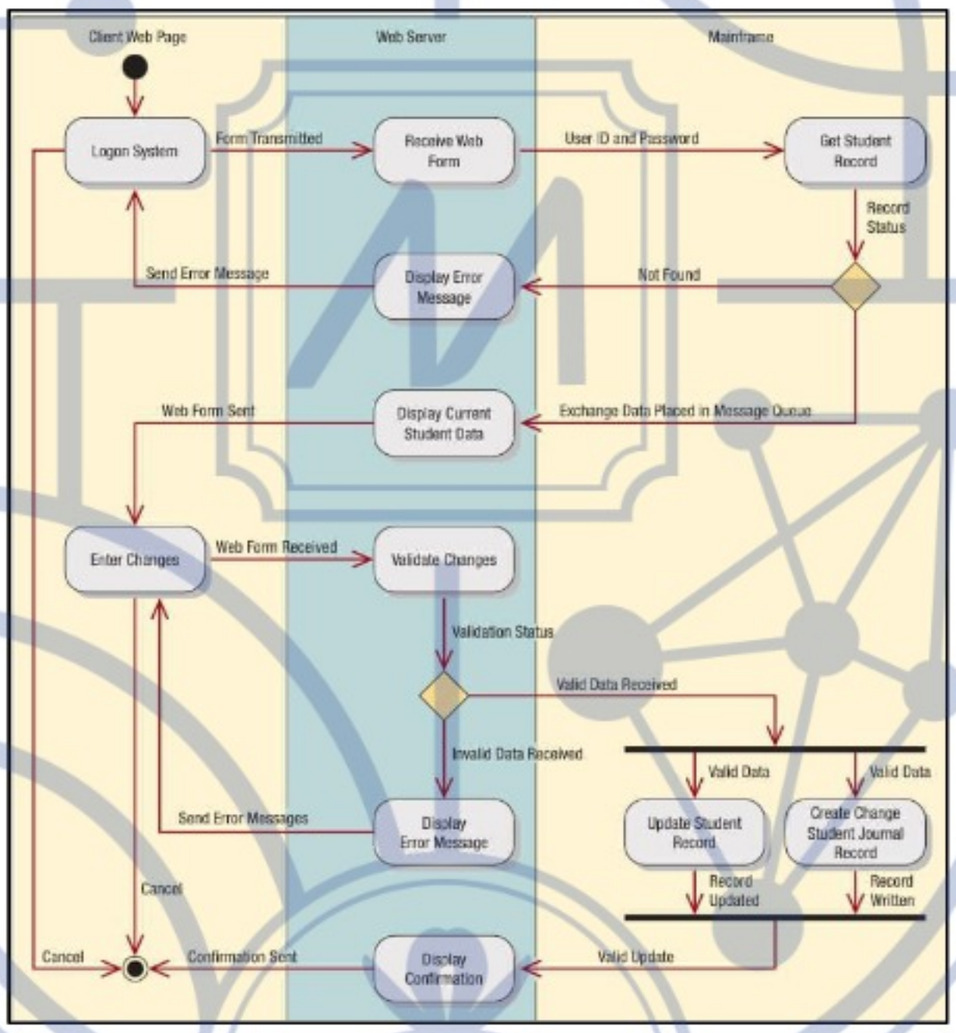
1. *White* merupakan tingkat paling tinggi, ini adalah tingkat perusahaan dan mungkin hanya terdapat empat hingga lima *use case* pada tingkat ini di seluruh organisasi. Contoh-contoh mungkin termasuk mempromosikan produk, menjual barang kepada konsumen, mengatur inventaris, mengelola jaringan pasokan, dan mengoptimalkan proses pengiriman.
2. *Kite* berposisi di tingkat yang lebih rendah dibandingkan *white* tetapi tetap berada di tingkatan tinggi, memberikan perpektif secara keseluruhan. Penerapan *kite* dapat berada di level unit bisnis atau departemen dan merupakan gambaran ringkas dari tujuan. Contohnya temasuk untuk mendaftarkan siswa, melakukan pemesanan tiket pesawat, hotel, mobil, atau kapal.
3. *Blue* terletak di permukaan air dan umumnya digunakan untuk mencerminkan tujuan dari pengguna. Tingkat ini sering dianggap paling menarik bagi pengguna serta paling mudah dimengerti oleh bisnis. Biasanya ditulis untuk kegiatan bisnis, dan setiap individu seharusnya bisa menyelesaikan satu aktivitas pada tingkat *blue* dalam rentang waktu 2 hingga 20 menit. Contohnya termasuk mendaftarkan mahasiswa yang melanjutkan pendidikan, menambah pelanggan baru, menempatkan barang ke dalam keranjang belanja, serta menyelesaikan proses pembayaran untuk pesanan.
4. *Indigo* atau *fish* merupakan contoh penggunaan yang menampilkan banyak rincian, sering kali pada tingkat fungsional atau subfungsional. Beberapa contohnya mencakup

pemilihan kelas, pembayaran biaya pendidikan, pencarian kode bandara untuk kota tertentu, serta pembuatan daftar pelanggan setelah mengetikkan nama.

5. Hitam atau krem, mirip dengan dasar lautan, merupakan ilustrasi paling rinci untuk penggunaan pada subfungsi. Salah satu contohnya adalah untuk memverifikasi *login* yang aman, menambahkan kolom baru dengan *HTML* yang dinamis.

2.6 *Activity* Diagram

Activity diagram adalah urutan aktivitas dalam suatu proses, termasuk aktivitas sekuensial dan parallel serta keputusan yang dibuat. *Activity* diagram dibuat dengan menanyakan apa yang terjadi pertama, apa yang terjadi kedua, dan seterusnya. *Activity* diagram digunakan untuk menyusun rencana pengujian. Setiap kejadian harus diuji untuk melihat apakah *activity* diagram mengarah pada keadaan selanjutnya [9].

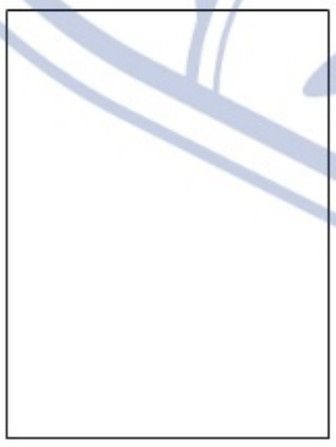


Gambar 2.6 Contoh *Activity* diagram

Activity diagram memiliki simbol simbol yang memiliki artinya masing masing, yaitu [9]:

Tabel 2.2 Simbol *Activity* diagram

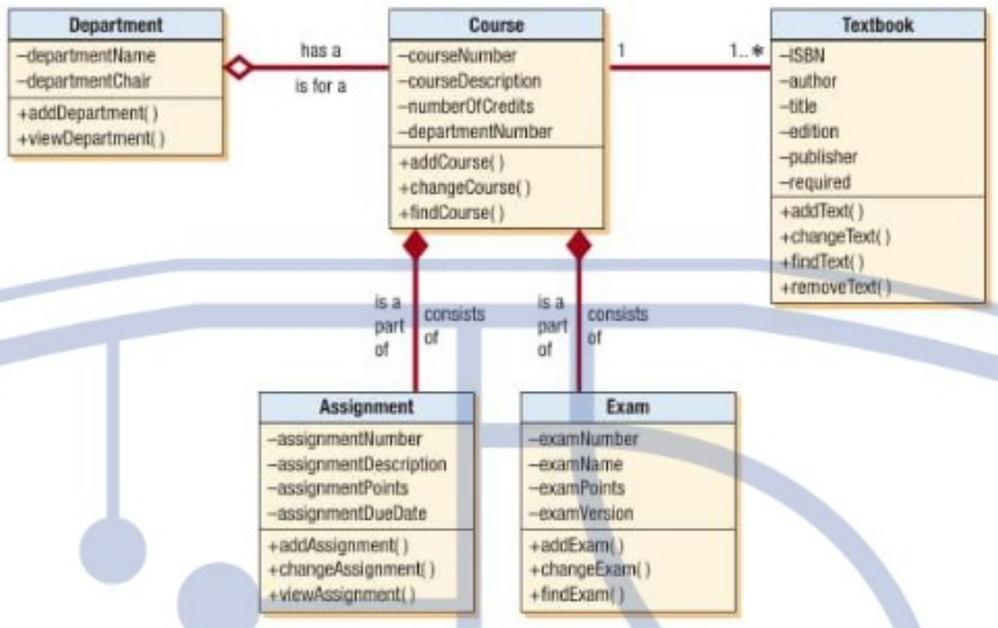
Nama	Simbol	Keterangan
Initial State		Dilambangkan dengan simbol lingkaran yang terisi penuh menunjukkan awal dari diagram.

Final State		Dilambangkan dengan lingkaran hitam yang dikelilingi lingkaran putih menunjukkan keadaan akhir dari diagram.
Activity		Dinyatakan dalam bentuk persegi panjang dengan sudut yang melengkung. Mewakili suatu aktivitas, baik aktivitas manual seperti menandatangin dokumen, maupun aktivitas otomatis seperti suatu metode.
Event		Dinyatakan dengan garis panah yang menunjukkan aliran dari satu aktivitas ke aktivitas lainnya. Melambangkan suatu peristiwa. Peristiwa mewakili hal-hal yang terjadi pada waktu dan tempat.
Decision		Dinyatakan dengan bentuk bentuk belah ketupat yang menunjukkan keputusan atau penggabungan aliran aktivitas. Keputusan memiliki satu panah yang masuk ke dalam belah ketupat dan beberapa panah yang keluar. Penggabungan menunjukkan beberapa peristiwa yang bergabung membentuk satu peristiwa.
Swimlane		*Swimlane* berbentuk seperti persegi panjang yang mengelilingi simbol lain, menunjukkan pembagian dan digunakan untuk menunjukkan aktivitas mana yang dilakukan pada plat*form* mana, atau menunjukkan aktivitas yang dilakukan oleh kelompok pengguna yang berbeda.

2.7 *Class* Diagram

Class Diagram adalah salah satu jenis diagram struktural dalam *Unified Modeling Language* (*UML*) yang digunakan untuk menggambarkan struktur statis dari sistem berbasis

objek. Diagram ini menunjukkan berbagai kelas dalam sistem, sifat-sifatnya, cara kerjanya, serta bagaimana kelas-kelas tersebut saling berkaitan. *Class* diagram memiliki peran penting dalam tahap analisis dan desain sistem karena memberikan gambaran dasar atau kerangka kerja untuk pengembangan kode serta basis data [9].



Gambar 2.7 Contoh *Class* Diagram

Dalam *UML class* diagram digambarkan sebagai sebuah segi empat dengan tiga bagian utama yaitu [9]:

1. *Class Name*: Nama yang membedakan kelas tersebut dari kelas lainnya, biasanya menggunakan kata benda dan dimulai dengan huruf kapital.
2. Atribut: Hal-hal yang diketahui oleh kelas mengenai ciri-ciri atau sifat-sifat objek (benda atau keadaan).
3. *Methods*: Apa yang diketahui oleh kelas tentang cara melakukan sesuatu (tindakan atau proses).

Dalam *class* diagram kelas dikelompokkan menjadi empat kategori utama guna mendukung proses analisis serta desain sistem berorientasi objek secara lebih terstruktur. Berikut penjelasan mengenai *entity, interface, abstract,* dan *control* [9].

1. *Entity (Entitas):* Kelas entitas merepresentasikan objek-objek dari dunia nyata seperti orang, tempat, atau benda yang memiliki keberadaan yang berlangsung dalam jangka waktu yang panjang dalam sistem. Analis perlu menentukan atribut mana yang penting bagi organisasi. Sebagai contoh, kelas Mahasiswa akan menyimpan atribut identitas seperti nama, alamat, dan Indeks Prestasi Akademik. Fungsinya adalah menyimpan informasi yang diperlukan oleh sistem selama jangka waktu yang lama.
2. *Interface* (Antarmuka): *Interface* memberikan cara bagi pengguna untuk berkomunikasi dan berinteraksi dengan sistem. *Interface* berperan sebagai perantara yang menghubungkan pengguna atau aktor dengan sistem internal. Fungsinya adalah

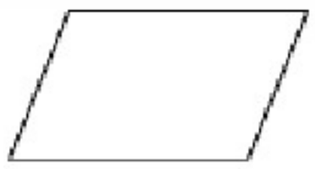
menerima masukan dari pengguna dan menampilkan hasil atau keluaran dari pemrosesan sistem kepada pengguna.

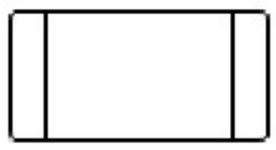
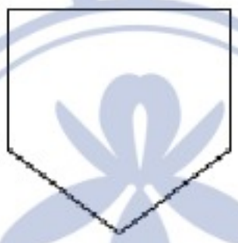
3. *Abstract* (Abstrak): Kelas abstrak adalah kelas yang tidak dapat dibuat menjadi objek secara langsung. Kelas abstrak bertindak sebagai kerangka dasar yang digunakan oleh kelas-kelas lain melalui mekanisme pewarisan. Nama kelas abstrak biasanya ditandai dengan huruf miring.
4. *Control* (Kontrol): Kelas kontrol, juga dikenal sebagai kelas aktif, digunakan untuk mengatur alur aktivitas, dan kelas ini berfungsi sebagai koordinator dalam proses pengimplementasian kelas. Untuk membuat kelas yang dapat digunakan kembali, diagram kelas dapat mencakup beberapa kelas kontrol yang kecil. Kelas kontrol sering kali disederhanakan selama proses perancangan sistem.

2.8 *Flowchart* Diagram

Flowchart program adalah cara untuk merepresentasikan algoritma yang digunakan untuk menghasilkan solusi dari suatu masalah. *Flowchart* terdiri dari simbol yang dihubungkan antara satu sama lain dengan panah yang harus diikuti Ketika bekerja melalui *flowchart*. *Flowchart* digunakan untuk menyampaikan kepada orang lain bagaimana suatu langkah dilaksanakan. *Flowchart* biasa dipakai ketika sebuah proyek baru mendalami perencanaan proyek tersebut. *Flowchart* berfungsi untuk menjelaskan bagaimana segala sesuatunya saat ini beroperasi dan bagaimana hal itu bisa diperbaiki. *Flowchart* juga mendukung dalam mengidentifikasi komponen-komponen penting dari suatu proses sekaligus menandai dengan jelas batas dimana suatu proses selesai dan proses berikutnya dimulai. *Flowchart* biasanya memiliki simbol simbol khusus. Berikut beberapa simbol standar yang digunakan dalam *flowchart* diagram [10]:

Tabel 2.3 Simbol *Flowchart* Diagram

Nama	**Simbol**	**Keterangan**
Terminal		Digunakan untuk menunjukkan awal dan akhir dari serangkaian proses
Input/Output		Digunakan untuk menunjukkan operasi *Input/output*
Proses		Digunakan untuk menunjukkan pemrosesan apa pun yang dilakukan, biasanya yang dilakukan oleh sistem komputer

Proses yang telah ditentukan		Digunakan untuk menunjukkan proses yang tidak didefinisikan secara khusus dalam *flowchart*
Komentar		Digunakan untuk menulis menyatakan penjelasan yang diperlukan untuk memperjelas sesuatu.
Garis Alir		Digunakan untuk menghubungkan simbol-simbol
Input/Output Dokumen		Digunakan ketika *Input* berasal dari dokumen dan *output* menuju dokumen
Keputusan/*Decision*		Digunakan untuk menunjukkan titik mana pun dalam proses di mana keputusan harus dibuat untuk menentukan tindakan lebih lanjut
Konektor Dalam Halaman		Digunakan untuk menghubungkan bagian-bagian dari *flowchart* yang dilanjutkan pada halaman yang sama
Konektor Luar Halaman		Digunakan untuk menghubungkan bagian-bagian dari diagram alur yang dilanjutkan ke halaman terpisah
Delay		Digunakan untuk menunjukkan penundaan atau menunggu dalam proses untuk mendapatkan *Input* dari beberapa proses lain.

2.9 *Database*

Database adalah sekumpulan informasi yang disimpan didalam perangkat komputer sehingga dapat dilihat dan diperiksa menggunakan program komputer yang digunakan untuk memperoleh suatu informasi atau mencapai suatu tujuan. *Database* berguna untuk

mengelola data dalam jumlah yang besar secara lebih efisien dan menjaga keamanan dalam jangka panjang [11]. Tujuan utama *database* adalah menyediakan wadah atau sumber daya untuk menampung dan menyimpan data serta berfungsi agar pengambilan data dapat dilakukan dengan lebih mudah, *database* dirancang untuk mengelola data dalam jumlah yang besar [12].

Database memberikan beberapa manfaat bagi perusahaan, diantaranya adalah [12]:

1. Memberikan kemudahan dan kecepatan dalam pengambilan data, *database* membantu perusahaan memudahkan pengkategorian data secara tepat. Hal ini dapat memberikan hasil yang optimal dalam pencarian dan penginputan data.
2. Sistem *database* dapat melindungi data-data yang ada didalamnya, sistem *database* akan mengamankan setiap isi dari data dengan menggunakan sandi dimana hanya yang berwenang yang dapat mengakses sandi tersebut.
3. *Database* dapat menghemat biaya, perusahaan yang menggunakan *database* dapat membantu penghematan biaya dalam membeli peralatan komputer, seperti *hard disk*. Karena menggunakan *Database* juga dapat memungkinkan perusahaan cabang untuk mengakses atau membuka data dari pusat.
4. Pengendalian data dilakukan secara singkat, *server* akan menyimpan data untuk kemudian dapat diakses oleh perusahaan cabang. Data juga dapat dikumpulkan disatu *server* agar mudah dilihat oleh pengguna lain.

2.10 Aplikasi

Aplikasi adalah suatu perangkat lunak yang dibuat khusus untuk memenuhi kebutuhan berbagai aktivitas dan pekerjaan misalnya: pelayanan masyarakat, aktivitas niaga, periklanan, *game*, dan berbagai aktivitas lainnya [13]. Tidak dapat dipungkiri bahwa aplikasi sangat membantu manusia dalam melakukan komunikasi, aplikasi terbagi menjadi beberapa jenis, yaitu [14]:

1. Aplikasi *Desktop*

   Aplikasi *Desktop* merupakan suatu aplikasi *software* yang ada pada *desktop* (PC dan laptop). Umumnya jenis aplikasi ini beroperasi tanpa terhubung dengan koneksi internet atau bisa dijalankan secara *offline*. Untuk menggunakannya pengguna harus menginstal terlebih dahulu di sistem operasi pada laptop maupun komputer. Aplikasi *desktop* dapat digunakan tanpa harus terhubung ke internet dan kemampuan control penuh atas data yang disimpan. Aplikasi *desktop* bisa dibuat dengan berbagai macam Bahasa

pemrograman seperti C#, Java, dan Delphi. Contoh aplikasi *desktop* adalah Microsoft Word, Microsoft Excel, Adobe Photoshop, Notepad, Corel Draw, dan lain-lain.

2. Aplikasi *Web*

   Jenis aplikasi ini memanfaatkan *web server* dan *browser* untuk menjalankannya, seperti Chrome, Firefox, dan Opera. Aplikasi *web* dapat berjalan dengan adanya jaringan internet maupun intranet (jaringan LAN). Kemudahan dalam mengakses adalah ciri utama yang membuat aplikasi *web* lebih banyak diminati dan lebih mudah diimplemantasikan pada berbagai kehidupan. Aplikasi berbasi *web* umumnya dikembangkan menggunakan Bahasa HTML, CSS, dan JavaScript. Contoh aplikasi *web* antara lain : Gmail, Trello, Google Docs, Google Spreadsheet, Youtube, dan lain-lain.

3. Aplikasi *Mobile*

   Aplikasi *mobile* merupakan perangkat lunak berupa aplikasi yang dikembangkan menggunakan program komputerisasi untuk disematkan pada perangkat *mobile*, seperti ponsel, tablet, dan jam tangan digital. Perkembangan mengenai bahasa pemrograman yang khusus diperuntukkan untuk aplikasi mobile semakin pesat. Beberapa Bahasa pemrograman yang biasa digunakan untuk membangun aplikasi *mobile* adalah Bahasa Kotin, Java, Objective C, Swift, Dart, JavaScript, dan React Native. Contoh aplikasi *mobile* antara lain: WhatsApp, Grab, Gojek, Halodoc, Instagram, dan lain-lain.

2.11 Bahasa Pemrograman C#

C# dikatakan sebagai C Sharp, merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft pada awal 2000-an, dipimpin oleh Anders Hejlsberg. Salah satu elemen dari kerangka .Net dan dirancang sebagai bahasa pemrograman umum yang mudah dipahami, yang dapat dimanfaatkan untuk menciptakan berbagai jenis aplikasi, termasuk aplikasi berbasis konsol, windows, *web*, dan *mobile*[15].

Ada beberapa tipe data yang digunakan dalam C#, yaitu [15]:

Tabel 2.4 Tipe Data C#

Type Data	Keterangan
Int	*Int* adalah singkatan dari *Integer* yaitu angka tanpa bagian decimal atau pecahan, dan memegang angka dari -2.147.483.648 hingga 2.147.483.647. Contohnya termasuk 15, 407, -908, 6150 dan lain-lain.

Byte	*Byte* juga mengacu pada bilangan integral, tetapi memiliki rentang yang lebih sempit dari 0 hingga 255. Sebagian besar waktu, kita menggunakan int alih-alih *byte* untuk bilangan integral. Namun, jika memprogram untuk mesin yang memiliki ruang memori terbatas, *user* harus menggunakan *byte* jika *user* yakin nilai variable tidak akan melebihi rentang 0 hingga 255. Misalnya, jika *user* perlu menyimpan usia pengguna, *user* dapat menggunakan tipe data *byte* karena kecil kemungkinan usia pengguna akan melebihi 255 tahun.
Float	*Float* mengacu pada angka *floating point*, yang merupakan angka dengan tempat desimal seperti 12,43, 5,2 dan -9,12. float dapat menyimpan angka dari -3,4 x 1038 hingga +3,4 x 1038. Menggunakan penyimpanan 8 *byte* dan memiliki presisi sekitar 7 digit. Artinya, jika *user* menggunakan *float* untuk menyimpan angka seperti 1.23456789 (10 digit), angka tersebut akan dibulatkan menjadi 1.234568 (7 digit).
Double	*Double* juga merupakan angka *floating point*, tetapi dapat menyimpan rentang angka yang jauh lebih luas. Ini dapat menyimpan angka dari (+/-)5,0 x 10-324 hingga (+/-)1,7 x 10308 dan memiliki presisi sekitar 15 hingga 16 digit. double adalah tipe data *floating point default* di C#. Dengan kata lain, jika Anda menulis angka seperti 2.34, C# memperlakukannya sebagai *double* secara default.
Decimal	*Decimal* menyimpan angka desimal tetapi memiliki rentang yang lebih kecil daripada *float* dan *double*. Namun, ia memiliki presisi yang jauh lebih besar sekitar 28-29 digit. Jika program Anda memerlukan tingkat presisi yang tinggi saat menyimpan bilangan non integral, Anda harus menggunakan tipe data decimal. Contohnya adalah ketika Anda menulis aplikasi keuangan di mana presisi sangat penting.
Bool	*Bool* adalah singkatan dari boolean dan hanya dapat menampung dua nilai: *true* dan *false*. Ini biasanya digunakan dalam pernyataan aliran kontrol.

Char	*Char* adalah singkatan dari *character* dan digunakan untuk menyimpan satu karakter *Unicode* seperti 'A', '%', '@' , 'p' dll.

2.12 SQL Server

SQL Server adalah salah satu dari *relational database management system* (*RDBMS*) yang menggunakan bahasa SQL untuk mengelola dan memanipulasi data. *Tools* yang dikembangkan oleh Microsoft ini mendapatkan perhatian dikalangan bisnis karena fitur kemanan, skalabilitas, integrasi dan konektivitas mudah dengan sistem eksternal. SQL Server pertama kali diluncurkan pada tahun 1998 dengan versi pertamanya yaitu SQL Server 7.0 yang memperkenalkan fitur-fitur inovatif seperti penyetelan otomatis dan penguncian baris yang dinamis. Versi selanjutnya yaitu SQL Server 2000 dan SQL Server 2005 terus menetapkan fondasi dengan menyediakan dukungan untuk aplikasi berbasis *web XML*, analitik yang canggih dan kemampuan penyimpanan pada gudang data [16]:

SQL Server 2012 menghadirkan peningkatan keamanan data seperti solusi pemulihan bencana. Selain itu, per*forma* kueri yang meningkat dan penggabungan data yang berbeda-beda *forma*t dengan *PolyBase* dan *JSON* serta integrasi *cloud* yang dinamis menambah daftar keunggualn dari perkembangan SQL Server. Pada SQL Server 2019 berfokus pada pengolahan data besar, penguatan fitur *Machine Learning* sambil menyediakan opsi untuk terhubung pada *database* lain seperti *Oracle*. Versi modern seperti SQL Server 2022 menghadirkan *server* yang mendukung kemajuan teknologi terbaru seperti analitik secara *real time* dan integrasi AI [16]: