BAB II
TINJAUAN PUSTAKA

2.1 Artificial Intelligence

Artificial Intelligence (AI) merupakan representasi pengetahuan yang berhubungan erat dengan teknologi berbasis komputerisasi dan menjadikan teknologi komputer perangkat teknologi cerdas yang memiliki kemampuan menyelesaikan aktivitas atau kinerja persis dengan kemampuan manusia (Jamaludin, 2020). Di negara Indonesia pengimplementasian AI paling banyak digunakan yaitu di bidang vision dan natural language processing seperti halnya Chatbot. Chatbot adalah salah satu teknologi kecerdasan buatan atau artificial intelligence berbasis percakapan singkat yang dapat membalas atau merespons percakapan manusia secara otomatis dan responsif (Jud, 2018).

Tren artificial intelligence pada masa kini diantara-nya:

1 Computer Vision

Yaitu bidang yang meliputi metode untuk melakukan pengolahan, melakukan analisis, dan memahami data visual

2 Natural Language Processing

Yaitu bidang yang lebih berfokus dalam melakukan pengolahan bahasa natural atau bahasa umum yang digunakan manusia, seperti chatbot

3 Virtual Reality

Yaitu bidang yang menangani cara teknologi dalam komunikasi manusia dengan simulasi komputer berdasarkan obyek nyata maupun animasi.

2.2 Chatbot

Chatbot merupakan suatu program dalam kecerdasan buatan yang dirancang untuk bisa berkomunikasi langsung dengan manusia atau user.Perbedaan chat bot dengan sistem pemrosesan bahasa alami(*Natural language processing System*) adalah penggunaan algoritma yang sederhana.Walaupun banyak bots yang bisa menginterpretasikan serta menanggapi input manusia, sebenarnya bots tersebut hanya mengartikan kata kunci dalam input dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari data yang telah ada dalam database yang telah dibuat sebelumnya.(Richard S, 2010).Chat dapat diartikan sebagai pembicaraan.Bot merupakan sebuah program yang mengandung sejumlah data serta jika diberikan masukan maka akan memberikan suatu jawaban. Chatbot dapat menjawab pertanyaan dengan membaca tulisan yang diketikkan oleh pengguna melalui keyboard. (Andriyani, 2004).

Pada awalnya, bots ini diuji melalui turing test, yaitu dengan merahasiakan identitasnya sebagai mesin sehingga dapat membohongi orang yang berbicara dengannya. Jika pengguna tidak dapat mengidentifikasi bots sebagai suatu program komputer, maka chatbot tersebut dikategorikan sebagai kecerdasan buatan (*artificial intelligence*).(Richard S, 2010). Salah satu chatbot yang terkenal adalah Eliza yang dikembangkan oleh Joseph Weizenbaum di MIT (Massachusetts Institute of Technology). Eliza mensimulasikan suatu percakapan antara seorang psikiater dengan pasiennya dalam bahasa Inggris yang alami. Eliza adalah chatbot pertama kali yang dibuat pada tahun 1964 sampai 1966 oleh Professor Joseph Weizenbaum di MIT (Massachusetts Institute of Technology), dengan tujuan untuk mempelajari komunikasi natural language antara manusia dengan mesin. Eliza adalah pelopor dari chatbot, Eliza dikenal sebagai suatu program chat yang berprofesi sebagai seorang psikiater. Eliza mensimulasikan suatu percakapan antara seorang psikiater dengan pasiennya menggunakan metode biasa , yang bisa mencerminkan perasaan pasien dengan mengajukan pertanyaan-pertanyaan seperti: How do you, Why do you feel like, What do you think about.

2.3 Artificial Intelligence Markup Language (AIML)

Artificial Intelligence Markup Language (AIML) merupakan bahasa scripting interpreter yang merupakan turunan dari *Extensible Markup Language* (XML) dengan fungsi yang lebih spesifik. Salah satu fungsinya yaitu membuat system stimulus-response berbasis pengetahuan. Dokumen AIML terdiri dari objek – objek yang dipisahkan oleh tag – tag tertentu layaknya dokumen XML atau HTML (Kurniawan, 2006). Obyek AIML tersusun atas unit-unit yang disebut topics dari categories, berisi data yang sudah terparsing maupun belum terparsing. Data yang ter-parsing berisi karakter – karakter, beberapa diantaranya berupa data karakter, yang lainnya dapat berupa elemen AIML. Elemen AIML mengkapsulasi pengetahuan dalam bentuk stimulus response di dokumen. AIML berisi kumpulan pola serta respon yang dapat digunakan oleh chatbot untuk penulusuran jawaban setiap kalimat yang diberikan. Interpreter AIML diperlukan untuk menerima input dan melakukan penulusuran sebuah jawaban pada dokumen AIML. Saat ini tersedia banyak interpreter AIML dalam berbagai bahasa pemrograman sehingga proses pembuatan chatbot dapat terfokus pada penyusunan dokumen AIML. AIML memiliki bentuk seperti XML yang memiliki beberapa tag markup.Berikut ini adalah beberapa tag markup yang digunakan :

1. <aiml>

Tag <aiml> berfungsi untuk mendefinisikan awal dan akhir dari file AIML.

2. <category>

Tag <category> berfungsi untuk mendefinisikan ilmu pengetahuan yang ditanamkan ke AI dalam bentuk kategori dan dilanjutkan dengan tag <pattern>.

3. <pattern>

Tag <pattern> berfungsi untuk mendefinisikan suatu karakter, kata, atau kalimat yang di inputkan oleh user kepada AI.

4. <template>

Tag <template> berfungsi untuk memberikan balasan/ respon kepada user terkait dengan pesan yang diterima oleh AI yang didapatkannya dari tag <pattern>.

Selain keempat tag markup tersebut ada beberapa tag yang terdapat pada AIML yaitu :

1. <star>

Tag ini berfungsi untuk mencocokkan semua karakter pada tag <pattern> yang dibuat dalam bentuk karakter *.

2. <srai>

Tag <srai> berfungsi untuk membuat *multiple tag* atau memanggil tag lain.

3. <random>

Tag <random> berfungsi untuk membuat proses pengacakan pesan balasan yang akan didapatkan oleh user.Pada tag ini harus ada statement list / tag <list> sehingga ada list atau daftar yang akan dipilih oleh tag <random>.

4.

Tag <li> berfungsi untuk membuat list dalam file AIML.

5. <set>

Tag <set> berfungsi untuk membuat atau memberi set nilai variable dalam file AIML.

6. <get>

Tag <get> merupakan kebalikan dari tag <set> yaitu untuk mendapatkan nilai variable yang ada dalam file AIML.

7. <that>

Tag ini berfungsi untuk membuat AI dapat membahas suatu topik tertentu yang dibahas oleh user dan AI memberikan suatu pesan balasan yang sesuai dengan jawaban user.

8. <topic>

Tag <topic> berfungsi untuk membuat AI dapat membahas suatu topik tertentu lebih dalam lagi sehingga pembahasannya bisa lebih panjang. Serta biasanya tag ini digunakan sebagai jawaban pengguna yang memiliki jawaban “yes” atau “no”.

9. <think>

Tag <think> berfungsi untuk membuat AI dapat menyimpan variable namun, nilai variable tersebut bisa tidak diberitahukan kepada user.

10. <condition>

Tag ini berfungsi untuk membuat fungsi percabangan dari jawaban yang akan diberikan AI kepada user.

Pembahasan tentang TAG AIML dan Contoh Penggunaan :

Seperti penjelasan sebelumnya bahwa pada AIML ada 4 tag utama dan 10 tag pendukung , totalnya adalah 14 tag dalam AIML. Berikut ini penjelasan fungsi detail dan contoh penggunaannya dalam sebuah percakapan atau komunikasi dengan user.

1. <aiml>

Tag <aiml> digunakan untuk penanda bahwa file tersebut adalah file AIML dan tag ini diletakkan di awal dan di akhir dari statement dalam file AIML. Contoh penggunaannya :

```
<?xml version="1.0" encoding="utf-8"?>

<aiml>

        <!-- letakkan tag atau statement di sini -->

</aiml>
```

2. <category>

Tag <category> digunakan untuk mencocokkan kata/topik tertentu yang diberikan kepada AI dalam bentuk kategori dan dilanjutkan dengan tag <pattern>. Contoh penggunaannya.

```
<?xml version="1.0" encoding="utf-8"?>

<aiml>
```

```
        <category>

            <!-- letakkan tag <pattern> di sini -->

        </category>

    </aiml>
```

3. <pattern>

Tag <pattern> digunakan untuk membuat pencocokan kata atau kalimat yang lebih spesifik lagi terhadap inputan si pengguna kepada AI. Contoh penggunaannya

```
<?xml version="1.0" encoding="utf-8"?>
<aiml>
        <category>
            <pattern>
                Nama Kamu Siapa
            </pattern>
        </category>
</aiml>
```

4. <template>

Tag <template> digunakan untuk membuat statement /pesan balasan yang akan diberikan kepada user terkait dengan inputan yang diberikannya kepada AI. Contoh penggunaannya

```
<?xml version="1.0" encoding="utf-8"?>
<aiml>
        <category>
            <pattern>Siapa Nama Kamu </pattern>
            <template>
                Nama saya adalah Chatbot.
```

```
            </template>
        </category>
    </aiml>
```

5. <star>

Tag <star> digunakan untuk membuat pencocokan karakter, kata ataupun kalimat yang kemungkinannya itu bisa lebih dari satu statement pada tag <pattern>. Dalam tag <pattern>, tag <star> ini dibuat dalam bentuk *. Atau dengan kata lain tag <star> seperti fungsi *regular expression* (regex) dalam bahasa pemrograman dan karakter * dalam statement tag <pattern> bisa digunakan untuk pencocokan atau *match* untuk seluruh karakter atau kalau dalam regex di Java ini seperti karakter . yang fungsinya untuk pencocokan seluruh karakter.

2.4 Pencocokan Pola (Pattern Matching)

Pencocokan pola / *pattern matching* merupakan metode yang digunakan untuk mencocokkan pola tertentu dengan sekumpulan kata/string (Budiasa,2009). Pada ilmu computer *pattern matching* telah banyak digunakan, diantaranya adalah mesin pencarian website, editor teks, analisis gambar dan sebagainya. String dapat diartikan sebagai kumpulan dari karakter yang membentuk satu kesatuan.

2.4.1 Graphmaster Pattern Matching

Graphmaster adalah metode yang digunakan untuk menyimpan kategori stimulus-respon dari *AIML*. Metode graphmaster digunakan untuk pencocokan pola dan penggunaan memori agar mencapai hasil yang efisien, dasar kerja dari metode ini adalah dengan menyimpan semua tag (<*category*>) didalam *AIML* dalam bentuk pohon yang bermula dari root sampai dengan patch tertentu dari suatu pattern. Graphmaster berbentuk pohon, ketika klien dari *bot* memasukkan teks sebagai stimulus atau suatu inputan maka *graphmaster* akan mencari kategori untuk mencocokkannya kedalam fungsi (<*pattern*>) yang sesuai dengan kalimat atau teks masukan lalu kemudian akan menghasilkan (<*template*>) sebagai responnya (Mahdiyah dan Andriani, 2013). Graphmaster matching (pencocokan graphmaster)

adalah pencocokanyang bersifat backtrack yaitu menggunakan strategi pencarian mendalam depth first search. Depth first search adalah salah satu pencarian buta (blind II-9 search). Pencarian ini dilakukan dari node awal secara mendalam hingga yang paling akhir atau sampai ditemukan. Dengan kata lain, simpul cabang atau anak yang terlebih dahulu dikunjungi.

Kelebihan Depth first search yaitu (Desiani dan Arhami, 2006):

1. Cepat mencapai kedalaman ruang pencarian.
2. Lebih efisien untuk ruang pencarian dengan banyak cabang karena tak perlu mengevaluasi semua simpul pada pada suatu level tertentu pada daftar open.
3. Memerlukan memori yang lebih kecil karena hanya node-node pada lintasan yang aktif saja yang akan disimpan.

Kelemahan Depth first search yaitu(Desiani dan Arhami, 2006):

1. Memungkinkan tidak ditemukannya tujuan yang diharapkan.
2. Hanya akan mendapatkansatu solusi pada setiap pencarian.

2.4.2 Algoritma Graphmaster Pattern Matching

Langkah-langkah algoritma graphmaster pattern matching (Mahdiyah dan Andriyani, 2013):

1. Sebagai contoh kata “APA” artinya inputan yang akan dicocokkan
2. Inisialisasikan istilah pertama artinya simbol atau node “_” kemudian cek apakah masukan terdiri asal node “_” ? Jika ya, cari subgraph dianak node yang berhubungan dengan “_”. Coba seluruh kata yang tersisa dari masukan yang mengikuti kata “APA” buat melihat kecocokan. Bila tidak ada lakukan langkah angka 3.
3. Cek apakah masukan terdiri dari istilah “atomic” ? Bila ya, cari subgraph dianak node yang mempunyai korelasi menggunakan “APA” menggunakan bagian belakang dari masukan. Jika tidak ditemukan kata kunci yang cocok maka lakukan langkah ke 4
4. Cek apakah masukan terdiri dari node “*” ? Jika ya, cari subgraph dianak

node yang memiliki korelasi menggunakan “*”. Coba residu akhiran dari semua masukan yang mengikuti istilah “APA” untuk mencari kecocokan. Jika tidak ditemukan yang cocok maka akan kembali ke induk node serta ambil kata “APA”menjadi masukan awal.

5. Jika masukan ialah kosong serta nodemapper memiliki kata kunci jawaban kemudian data yang cocok ditemukan, maka pencarian berhenti serta kembalikan data yang cocok menjadi hasil.

Jika nodemapper memiliki istilah kunci “*” serta menunjukkan kesebuah node, prosedur pemecahan ditetapkan buat menemukan yang akan terjadi yang cocok.Prosedur pemecahan graphmaster pattern matching mempunyai beberapa ketentuan,diantaranya ialah :

1. Dalam pencarian node, yang menjadi prioritas primer atau yang pertama dicari artinya node “_”, kemudian string atau kata yang cocok memiliki prioritas ke 2 dan node “*” merupakan prioritas terakhir.
2. Sebuah pattern atau masukan tidak diurutkan sesuai alphabet, tujuannya supaya pencarian node berurutan yaitu dimulai dari node “_” kemudian node atau string yang cocok baru lalu menuju ke node “*”.
3. Pencocokan yang dilakukan adalah pencocokan sesuai kata bukan berdasarkan kategori, masukan pengguna akan dipisahkan sesuai spasi lalu dicocokkan
4. Algoritma ini mengkombinasikan aneka macam masukan seperti pattern, (<thatpattern>) dan (<topic>) , thatpattern adalah respon akan diberikan Jika pertanyaan yang diberikan berkaitan dengan pertanyaan sebelumnya sedangkan topic berarti pembicaraan yang focus pada satu pembahasan.
5. prosedur pemecahan ini dilakukan dengan melakukan pencarian secara mendalam (Back tracking) atau yang dikenal menggunakan algoritma Deph First Search(DFS).

2.5 Android

Android merupakan sistem operasi yang dirancang oleh Google dengan berbasis kernel linux untuk mendukung kinerja suatu perangkat elektronik layar sentuh, seperti tablet atau *smartphone* (Yosef Murya, 2014). Jadi, Android digunakan dengan sentuhan, atau gesekan atau ketukan pada layar *gadget*. Sistem Operasi Android bersifat *open source* atau bebas untuk digunakan, dimodifikasi, diperbaiki serta didistribusikan oleh para pengembang perangkat lunak. Dengan sifat *open source* ,suatu perusahaan teknologi bebas mengunakan OS ini diperangkatnya tanpa memerlukan lisensi alias gratis.Para pengembang aplikasi bebas membuat aplikasi dengan kode-kode sumber yang dikeluarkan google. Dengan seperti itu, android memiliki jutaan *support* aplikasi gratis atau berbayar yang dapat diunguh melalui google play store.

Pada awalnya Android Inc adalah sebuah perusahaan software kecil yang didirikan pada bulan Oktober 2003 di Palo Alto, California USA. Didirikan oleh senior di beberapa perusahaan yang berbasis *IT* dan *communication*; Andy Rubin, Rich Miner, Nick Sears dan Chris White. Menurut Rubin, Android Inc didirikan untuk mewujudkan suatu mobile device yang lebih peka terhadap lokasi dan peferensi pemilik. Dengan kata lain, Android Inc berusaha mewujudkan *mobile device* yang lebih mengerti pemiliknya. Sistem operasi ini membuka pintu untuk para pengembang/developer untuk mengembangkan software dengan android SDK (*Soft Development Kit*), yang menyediakan tool dan API yang dibutuhkan untuk memulai mengembangkan aplikasi platform Android menggunakan pemrograman Java.

2.6 Pemrograman Java

Java merupakan bahasa pemrograman yang dapat dijalankan di berbagai perangkat komputer maupun telepon. Java pertama kali dibuat oleh James Gosling yang tergabung dalam perusahaan Sun Microsystem pada tahun 1995 (Rosa dan Shalahuddin, 2014). Definisi Java menurut Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* atau pada lingkungan jaringan. Java berdiri dari sebuah mesin interpreter yang diberi nama Java Virtual Machine (JVM). *Platform* Java terdiri atas

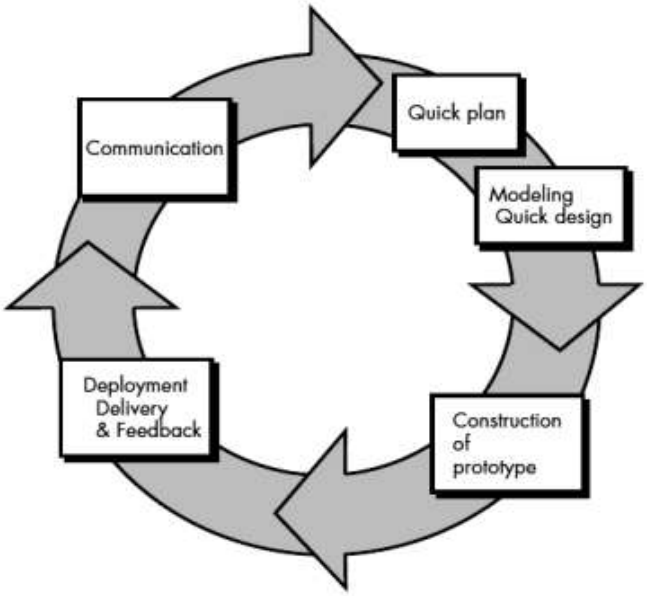
kumpulan *library* JVM, kelas-kelas *loader* dipaket dalam sebuah *compiler*, *debuger,* serta perangkat lain yang dipaket dalam Java Development Kit. Java mengadopsi bahasa program C dan C++ ke arah lebih sederhana serta dukungan paket-paket untuk pengembangan segala aplikasi yang dibuat dengan pemrograman*java*. Aplikasi *java* biasanya pada umumnya dikompilasi ke dalam *p-code* (*bytecode*) dan dapat dijalankan di berbagai mesin virtual *java* (JVM). Java lebih bersifat umum (*general purpose*), dan secara khusus di desain untuk memanfaatkan dependensi implementasi seminimal mungkin. Oleh karena fungsionalitas, aplikasi *java* mampu berjalan di beberapa platform sistem operasi yang berbeda. Pemrograman *java* lebih berorientasi kepada pengembangan aplikasi yang bersifat OOP (*Object Oriented Programming*) yaitu pemrograman yang berorientasi terhadap sebuah objek. Java memiliki dukungan berbagai jenis paket *library-library* yang berguna bagi pengembang aplikasi ke tahap lebih lanjut. Salah satu kelemahan dari pemrograman *java* yaitu pemakaian memori yang memakan terlalu banyak, sehingga dapat memungkinkan mempengaruhi kinerja komputer bekerja lebih lambat. Akan tetapi secara global pemrograman *java* memiliki nilai positif dari pemrograman yang lain karena sifatnya yang *open source* sehingga dapat memudahkan para pengembang untuk mengembangkan aplikasi yang terbuat dari bahasa pemrograman *java.*

2.7 Metode Prototype

Prototype digunakan untuk menggali kebutuhan secara lebih cepat. Biasanya saat pembuatan *prototype*, keterlibatan user sangat dibutuhkan. Manfaat utama prototype adalah untuk mengurangi resiko tidak diterimanya hasil pengembangan suatu perangkat lunak serta pengulangan kerja di kemudian hari (Munawar, 2018). Manfaat itu diantaranya :

1. Lebih antusiasnya pengguna akhir dan pelanggan dalam proses penggalian kebutuhan dan umpan balik.
2. Dapat Mengurangi resiko proyek, karena penggunaan user interface sudah dieksplorasi dari awal.
3. Memberikan kemudahan dalam penggunaan bisa lebih ditingkatkan.

4. Dapat menjembatani antara produk perangkat lunak dengan kebutuhan pengguna.



Gambar 2.1 Metode Prototype

Langkah-langkah dalam metode *Prototype* dimulai dengan dilakukannya Komunikasi antara Tim Pengembang perangkat lunak dengan para pelanggan. Tim pengembang perangkat lunak akan melakukan pertemuan- pertemuan dengan para Stakeholder untuk mendefinisikan sasaran keseluruhan untuk perangkat lunak yang akan dikembangkan, mengidentifikasi spesifikasi kebutuhan apa pun yang saat ini diketahui, dan menggambarkan area-area dimana definisi lebih jauh pada literasi selanjutnya merupakan keharusan. Literasi pembuatan Prototype direncanakan dengan cepat dan pemodelan (dalam bentuk "Rancangan Cepat") dilakukan. Suatu rancangan cepat berfokuskan pada representasi semua aspek perangkat lunak yang akan terlihat oleh para pengguna akhir

2.8 Unified Modeling Language (UML)

UML (*Unified Modeling Language*) adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requerement*, membuat analisa & desain, serta menggambarkan arsitektur dalam pemrograman berorintasi

objek (Rosa dan Shalahuddin, 2014). Sedangkan Mulyani (2016:48) mengatakan UML (*Unified Modeling Language*) adalah "Sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem".

Dari beberapa penjelasan teori tersebut dapat disimpulkan bahwa UML (*Unified Modeling Language*) adalah bahasa yang sering digunakan untuk membangun sebuah sistem perangkat lunak dengan melakukan penganalisaan desain dan spesifikasi dalam pemrograman berorintasi objek.

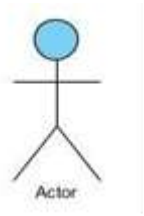
UML (*Unified Modeling Language*) memiliki diagram-diagram yang digunakan dalam pembuatan aplikasi berorintasi objek, diantaranya (Rosa dan Shalahuddin,2014:155)

1. *Use Case Diagram*

Use Case Diargam merupakan pemodelan untuk melakukan (*behavior*) sistem informai yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.Berikut adalah simbol-simbol yang ada pada diagram *use case* (Rosa dan Shalahuddin, 2014:156):

Tabel 2.1 *Simbol-simbol Use Case Diagram*

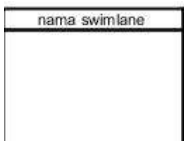
Simbol	Deskrpsi
Use case **Name Use**	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling tertukar pesan antar unit atau actor,biasanya dinyatakan dengan menggunakan kata kerja diawal *frase* nama *Use Case*

Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informaasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor merupakan gambar orang, tapi aktor belum tentu merupakan orang: biasanya dinyatakan menggunakan kata benda diawal *frase* nama actor
Asosiasi / association 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* <<extend>> 	Relasi *use case* tambahan kesebuah *use case* dinamakan *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek,biasanya *use case* tambahan mempunyai nama depan yang sama dengan *use case* yang ditambahkan.
Generalisasi / *generalization* 	Hubungan generalisasi & spesialisasi (umum-khusus) antara 2 buah *use case* dimana fungsi yang satu merupakan fungsi yang lebih umum dari lainnya.
Menggunakan / *include/uses* <<include>> 	Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini

2. *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Perlu diperhatikan bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.Berikut adalah simbol-simbol yang ada pada diagram aktivitas (Rosa dan Shalahuddin, 2014:162):

Tabel 2.2 *Simbol-simbol Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimelane  Atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.9 Penelitian Terdahulu

Banyak penelitian yang sudah dilakukan tentang bagaimana membangun sebuah aplikasi *chatbot* yang maksimal, maka diperlukan studi literatur sebagai tahap metode penelitian yang akan dilakukan. Berikut ini beberapa penelitian yang dilakukan sebelumnya dan memenuhi topik yang sesuai dengan pembahasan dalam penelitian ini, antara lain:

1. Penelitian dengan judul "Perancangan Aplikasi Teknologi *Chatbot* Untuk Industri Komersial 4.0" pada tahun 2019 yang dilakukan oleh Rani Natadian menerangkan bahwa aplikasi yang dibangun mampu berperan sebagai *customer*

service dalam menangani pertanyaan-pertanyaan dengan pola yang berbeda-beda dengan memberikan jawaban sesuai masing-masing kategori.

2. Penelitian dengan judul "Membangun Aplikasi *Chatbot* Berbasis Web Pada CV Unomax Indonesia" pada tahun 2019 yang dilakukan oleh Teddy Wijaya menerangkan bahwa aplikasi yang dibangun mampu menjawab semua pertanyaan dengan cepat sesuai yang diharapkan sehingga pelanggan lebih dimudahkan tanpa ada keterbatasan waktu.
3. Penelitian dengan judul "Rancang Bangun Aplikasi *Chatbot* Sebagai Bentuk Pelayanan Prima Untuk Penerimaan Mahasiswa Baru" pada tahun 2019 yang dilakukan oleh Murhadi menerangkan bahwa aplikasi yang dibangun dapat memenuhi faktor sikap (*attitude*), perhatian (*attention*), tindakan (*action*), kemampuan (*ability*), penampilan (*appearance*), dan tanggung jawab (*accountability*).
4. Penelitian dengan judul "Rancang Bangun Aplikasi Chatbot Berbasis AIML Dengan Metode Pattern Matching (Studi Kasus : Akademik Uin Suska Riau)" pada tahun 2017 yang dilakukan oleh Agus Triono dibuat menggunakan basis pengetahuan AIML dan algoritma Pattern Matching dalam pencocokan pola kata kunci masukan yang digunakan untuk mencari jawaban yang sesuai dari masukan. Pada penelitian ini *Chatbot* dibangun dengan menggunakan bahasa pemrograman PHP.