BAB II

TINJAUAN PUSTAKA

II.1 Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Muhammad, dkk (2018) mengenai Implementasi Metode Smarter Untuk Rekomendasi Pemilihan Lokasi Pembangunan Perumahan Di Pekanbaru, Muhammad, dkk menyimpulkan bahwa Sistem pendukung keputusan yang dibangun dapat merangkingkan dengan menghasilkan urutan nilai terbesar hingga terkecil alternatif lokasi pembangunan perumahan yang diperoleh dari perhitungan dengan metode SMARTER, sehingga dapat membantu pihak manajer dan developer dalam mengambil keputusan secara terkomputerisasi dengan tepat.

Berdasarkan penelitian yang dilakukan oleh Simarmata, dkk (2019) mengenai Penerapan Metode Smarter Dalam Sistem Pendukung Keputusan Menentukan Kualitas Getah Karet (Studi Kasus : PTPN III Medan), Simarmata, dkk menyimpulkan bahwa dari 5 data getah karet yang diuji, diperoleh hasil pengujian dengan rekomendasi perangkingan dari data getah karet tersebut. Dengan mempertimbang setiap nilai kriteria yang menjadi faktor penentu kualitas getah karet. Maka diperolehlah hasil kualitas getah karet terbaik dengan persentase kualitas sebesar 86%.

Berdasarkan penelitian yang dilakukan oleh Sianturi (2020) mengenai Analisis Metode *Simple Multi Attribute Rating Technique Exploiting Rank* (SMARTER) Pada Sistem Pendukung Keputusan Pemilihan Smartphone Android

Di Toko Shine Cellular Mega Bekasi Hypermall, Sianturi menyimpulkan bahwa Metode *Simple Multi Attribute Rating Technique Exploiting Rank* (SMARTER) dapat diterapkan sebagai metode dalam sistem pendukung keputusan, karena nilai akhir yang sama atau yang mendekati pada perhitungan dapat menjadikan pilihan alternatif terbaik.

Berdasarkan penelitian yang dilakukan oleh Saleh (2017) mengenai Penerapan Metode *Simple Multi Attribute Rating Technique Exploiting Rank* Dalam Sistem Pendukung Keputusan Rekrutmen Asisten Laboratorium Komputer, Saleh menyimpulkan bahwa dari kesepuluh data pelamar yang dijadikan alternative dalam pengujian metode SMARTER, diperoleh hasil akurasi sebesar 80%.

Berdasarkan penelitian yang dilakukan oleh Haryani dkk, (2016) mengenai Sistem Pendukung Keputusan Seleksi Penerimaan Mahasiswa Pengganti Beasiswa Penuh Bidikmisi Universitas Tanjungpura Dengan Menerapkan Metode SMARTER, Haryani dkk, menyimpulkan bahwa metode SMARTER dapat melakukan seleksi penerimaan mahasiswa pengganti beasiswa penuh Bidikmisi Universitas Tanjungpura dengan tingkat validitas sebesar 71,43 %.

Berdasarkan lima penelitian terdahulu, peneliti menyimpulkan bahwa metode SMARTER dapat digunakan dengan berbagai jenis masalah untuk diselesaikan, oleh karena itu penelitian ini menggunakan metode SMARTER untuk perankingan kelompok nelayan penerima bantuan sarana perikanan tangkap dan lima penelitian terdahulu dapat dijadikan referensi untuk penelitian ini.

II.2 Landasan Teori

Berikut ini adalah landasan teori yang berkaitan dengan penelitian yang di ambil dari beberapa jurnal dan buku :

II.2.1 Sistem Pendukung Keputusan

Menurut Alter dalam Buku Konsep dan Aplikasi Sistem Pendukung Keputusan (Kusrini, 2007), Sistem Pendukung Keputusan (SPK) merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan dan pemanipulasian data untuk membantu pengambilan keputusan. SPK tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan metode-metode yang tersedia.

Sedangkan menurut (Sihite, 2020) Sistem Pendukung Keputusan (*Decision Support System*) adalah suatu sistem yang memiliki kemampuan dalam pemecahan masalah/komunikasi untuk kondisi masalah yang terstruktur maupun tidak terstruktur yang mempunyai peran dalam membantu pemecahan masalah dan tidak satupun yang mengetahui bagaimana keputusan yang seharusnya dibuat.

II.2.2 Metode Simple Multi Attribute Rating Technique Exploiting Ranks

Metode SMARTER (*Simple Multi Attribute Rating Technique Exploiting Rank*) merupakan pengembangan dari metode SMART (*Simple Multi-Attribute Rating Technique*). Metode SMART pertama kali diperkenalkan oleh Edward pada tahun 1971 dan baru dinamai sebagai metode SMART pada tahun 1977. Semenjak awal kemunculannya, metode SMART telah dikembangkan menjadi metode SMARTS (*Simple Multi-Attribute Rating Technique Swing*) lalu setelah dimodifikasi dan diperbaiki oleh Edward dan Baron pada tahun 1994 menjadi metode SMARTER (*Simple MultiAttribute Rating Technique Exploiting Rank*. Perbedaan antara metode SMARTER dengan metode SMART dan SMARTS terletak pada

cara pembobotannya. Pembobotan kriteria pada ketiga metode tersebut tergantung pada urutan prioritas atribut dimana pada urutan pertama ditempati oleh atribut yang dianggap paling penting. Pada metode SMART dan SMARTER pembobotan diberikan langsung oleh pengambil keputusan. Tetapi prosedur pembobotan tersebut dianggap tidak proporsional dimana setiap bobot yang diberikan harus mencerminkan jarak dan prioritas setiap kriteria dengan tepat. Untuk mengatasi hal tersebut, pada metode SMARTER digunakan rumus pembobotan Rank Order Centroid (ROC). Rumus metode SMARTER secara umum dapat dilihat pada persamaan II.1 berikut : (Alfa Saleh : 2017)

Un = $\sum_{k=1}^{K} WkUn(Xnk)$

Keterangan:

Un = Nilai akhir

Wk = Bobot dari kriteria ke k

Un (Xnk) = Nilai utility kriteria ke k untuk alternatif ke-h

Nilai Utility juga diperlukan sebelum menghitung nilai akhir, persamaan yang digunakan untuk menghitung nilai Utility dapat dilihat pada persamaan II.2 berikut :

U_i (a_i) = 100% x $\frac{(Ci-Cmin)}{(Cmaks-Cmin)}$

Keterangan :

U_i (a_i) = nilai utility kriteria ke-i untuk kriteria ke-i

Ci = nilai kriteria ke-i

Cmin = Nilai kriteria minimal

Cmax= Nilai kriteria maksimal (Alfa Saleh : 2017)

II.2.3 Aplikasi

Aplikasi dalam Kamus Besar Bahasa Indonesia (KBBI) adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari *user* (pengguna).

Menurut (Riswaya, 2014) aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan.

II.2.4 Database

Menurut (Dicoding, 2020) *Database* atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi. Adapun pengertian lain dari *database* adalah sistem yang berfungsi sebagai mengumpulkan *file*, tabel, atau arsip yang terhubung dan disimpan dalam berbagai media elektronik.

SQL (*Structured Query Language*) adalah bahasa non procedural untuk mengakses data pada database relasional. SQL adalah bahasa database yang dipergunakan dalam menyelesaikan permasalahan dalam database serta mempunyai kelebihan dalam mengolah data. Standar SQL mula-mula didefenisikan oleh ISO *(International Standards Organization)* dan ANSI *(the American National Standards Institute)* yang dikenal dengan sebutan SQL86. Dengan menggunakan SQL, kita dapat melakukan hal-hal berikut:

1. Memodifikasi struktur *database* .
2. Mengubah, mengisi, menghapus isi *database.*
3. Mentransfer data antara database yang berbeda.

SQL ada yang dikembangkan untuk PC dan ada juga yang dikembangkan untuk dapat mengakomodasi database yang sangat besar. Beberapa contohnya antara lain:

1. *Microsoft Accsess*

Digunakan untuk PC, sangat mudah dipakai dimana perintah SQL dapat langsung dimasukkan atau melalui fasilitas yang telah digunakan.

2. *Microsoft Query*

SQL yang dipaket dengan produk lain dari Microsoft Windows, yaitu Microsoft Visual Studio seperti Visual Basic dan Visual C++. Untuk terhubung dengan database lain menggunakan ODBC.

3. *Oracle*

Digunakan untuk perusahaan yang mengggunakan database besar (Eka Iswandy; 2017 : 73).

II.2.5 UML

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi diagram (Alfina & Harahap, 2019).

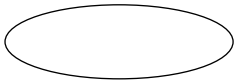
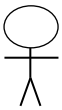
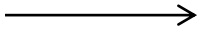
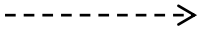
. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML yaitu *Use Case Diagram, Activity Diagram, Class Diagram* dan *Sequence Diagram*.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada Tabel II.1:

Tabel II.1. Simbol *Use Case*

Gambar	**Keterangan**
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.

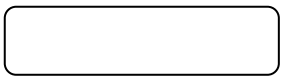
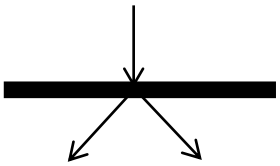
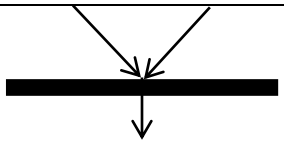
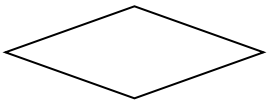
<--------	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Janiver W. Janis; 2020)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada Tabel II.2:

Tabel II.2. Simbol *Activity Diagram*

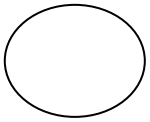
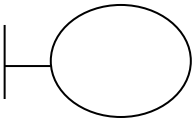
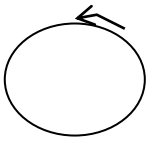
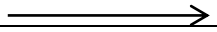
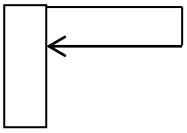
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Janiver W. Janis; 2020)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada Tabel II.3:

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message,* simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

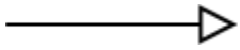
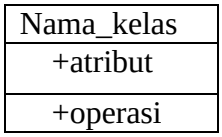
(Sumber : Janiver W. Janis; 2020)

4. Diagram Kelas (*Class Diagram)*

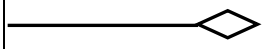
Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations, Generalization* dan *Aggregation,*

Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti (Ade Hendini, 2016 : 111). Simbol *class diagram* dan *multiplicity class diagram* dapat dilihat pada Tabel II.4 dan Tabel II.5.:

Tabel II.4. Simbol *Class Diagram*

Gambar	Keterangan
	*Generalization*, untuk menghubungkan antar kelas dengan arti umum-khusus. Jadi jika ada kelas dengan arti umum-khusus. Jadi jika ada kelas bermakna umum dan kelas bermakna khusus dapat menggunakan simbol ini.
	*Class*, untuk sebuah kelas pada struktur sistem. Penulisan tidak boleh menggunakan spasi. Simbol ini memiliki 3 susunan, yaitu kotak pertama adalah nama kelas, kedua atribut dan ketiga operasi.
	*Interface*, untuk simbol *interface* atau dalam bahasa indonesianya antar muka. Konsep yang digunakan pun sama dengan pemrograman berorientasi object (OOP).
	Association,digunakan untuk menghubungkan atau merelasikan kelas satu dengan kelas yang lainnya dengan makna umum.
	Directed Association, adalah relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.

Tabel II.4. Simbol *Class Diagram (*Lanjutan)

	*Aggregation*, adalah relasi antar kelas dengan makna semua bagian.
	Dependency, adalah relasi antar kelas dengan makna kebergantungan antar kelas.

(Sumber : Janiver W. Janis; 2020)

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1

n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Janiver W. Janis; 2020)

Di Toko Shine Cellular Mega Bekasi Hypermall, Sianturi menyimpulkan bahwa Metode *Simple Multi Attribute Rating Technique Exploiting Rank* (SMARTER) dapat diterapkan sebagai metode dalam sistem pendukung keputusan, karena nilai akhir yang sama atau yang mendekati pada perhitungan dapat menjadikan pilihan alternatif terbaik.

Berdasarkan penelitian yang dilakukan oleh Saleh (2017) mengenai Penerapan Metode *Simple Multi Attribute Rating Technique Exploiting Rank* Dalam Sistem Pendukung Keputusan Rekrutmen Asisten Laboratorium Komputer, Saleh menyimpulkan bahwa dari kesepuluh data pelamar yang dijadikan alternative dalam pengujian metode SMARTER, diperoleh hasil akurasi sebesar 80%.

Berdasarkan penelitian yang dilakukan oleh Haryani dkk, (2016) mengenai Sistem Pendukung Keputusan Seleksi Penerimaan Mahasiswa Pengganti Beasiswa Penuh Bidikmisi Universitas Tanjungpura Dengan Menerapkan Metode SMARTER, Haryani dkk, menyimpulkan bahwa metode SMARTER dapat melakukan seleksi penerimaan mahasiswa pengganti beasiswa penuh Bidikmisi Universitas Tanjungpura dengan tingkat validitas sebesar 71,43 %.

Berdasarkan lima penelitian terdahulu, peneliti menyimpulkan bahwa metode SMARTER dapat digunakan dengan berbagai jenis masalah untuk diselesaikan, oleh karena itu penelitian ini menggunakan metode SMARTER untuk perankingan kelompok nelayan penerima bantuan sarana perikanan tangkap dan lima penelitian terdahulu dapat dijadikan referensi untuk penelitian ini.

II.2 Landasan Teori

Berikut ini adalah landasan teori yang berkaitan dengan penelitian yang di ambil dari beberapa jurnal dan buku :

II.2.1 Sistem Pendukung Keputusan

Menurut Alter dalam Buku Konsep dan Aplikasi Sistem Pendukung Keputusan (Kusrini, 2007), Sistem Pendukung Keputusan (SPK) merupakan sistem informasi interaktif yang menyediakan informasi, pemodelan dan pemanipulasian data untuk membantu pengambilan keputusan. SPK tidak dimaksudkan untuk mengotomatisasikan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan metode-metode yang tersedia.

Sedangkan menurut (Sihite, 2020) Sistem Pendukung Keputusan (*Decision Support System*) adalah suatu sistem yang memiliki kemampuan dalam pemecahan masalah/komunikasi untuk kondisi masalah yang terstruktur maupun tidak terstruktur yang mempunyai peran dalam membantu pemecahan masalah dan tidak satupun yang mengetahui bagaimana keputusan yang seharusnya dibuat.

II.2.2 Metode Simple Multi Attribute Rating Technique Exploiting Ranks

Metode SMARTER (*Simple Multi Attribute Rating Technique Exploiting Rank*) merupakan pengembangan dari metode SMART (*Simple Multi-Attribute Rating Technique*). Metode SMART pertama kali diperkenalkan oleh Edward pada tahun 1971 dan baru dinamai sebagai metode SMART pada tahun 1977. Semenjak awal kemunculannya, metode SMART telah dikembangkan menjadi metode SMARTS (*Simple Multi-Attribute Rating Technique Swing*) lalu setelah dimodifikasi dan diperbaiki oleh Edward dan Baron pada tahun 1994 menjadi metode SMARTER (*Simple MultiAttribute Rating Technique Exploiting Rank*. Perbedaan antara metode SMARTER dengan metode SMART dan SMARTS terletak pada

cara pembobotannya. Pembobotan kriteria pada ketiga metode tersebut tergantung pada urutan prioritas atribut dimana pada urutan pertama ditempati oleh atribut yang dianggap paling penting. Pada metode SMART dan SMARTER pembobotan diberikan langsung oleh pengambil keputusan. Tetapi prosedur pembobotan tersebut dianggap tidak proporsional dimana setiap bobot yang diberikan harus mencerminkan jarak dan prioritas setiap kriteria dengan tepat. Untuk mengatasi hal tersebut, pada metode SMARTER digunakan rumus pembobotan Rank Order Centroid (ROC). Rumus metode SMARTER secara umum dapat dilihat pada persamaan II.1 berikut : (Alfa Saleh : 2017)

Un = $\sum_{k=1}^{K} WkUn(Xnk)$

Keterangan:

Un = Nilai akhir

Wk = Bobot dari kriteria ke k

Un (Xnk) = Nilai utility kriteria ke k untuk alternatif ke-h

Nilai Utility juga diperlukan sebelum menghitung nilai akhir, persamaan yang digunakan untuk menghitung nilai Utility dapat dilihat pada persamaan II.2 berikut :

U_i (a_i) = 100% x $\frac{(Ci-Cmin)}{(Cmaks-Cmin)}$

Keterangan :

U_i (a_i) = nilai utility kriteria ke-i untuk kriteria ke-i

Ci = nilai kriteria ke-i

Cmin = Nilai kriteria minimal

Cmax= Nilai kriteria maksimal (Alfa Saleh : 2017)

II.2.3 Aplikasi

Aplikasi dalam Kamus Besar Bahasa Indonesia (KBBI) adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari *user* (pengguna).

Menurut (Riswaya, 2014) aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan.

II.2.4 Database

Menurut (Dicoding, 2020) *Database* atau basis data adalah kumpulan data yang dikelola sedemikian rupa berdasarkan ketentuan tertentu yang saling berhubungan sehingga mudah dalam pengelolaannya. Melalui pengelolaan tersebut pengguna dapat memperoleh kemudahan dalam mencari informasi, menyimpan informasi dan membuang informasi. Adapun pengertian lain dari *database* adalah sistem yang berfungsi sebagai mengumpulkan *file*, tabel, atau arsip yang terhubung dan disimpan dalam berbagai media elektronik.

SQL (*Structured Query Language*) adalah bahasa non procedural untuk mengakses data pada database relasional. SQL adalah bahasa database yang dipergunakan dalam menyelesaikan permasalahan dalam database serta mempunyai kelebihan dalam mengolah data. Standar SQL mula-mula didefenisikan oleh ISO *(International Standards Organization)* dan ANSI *(the American National Standards Institute)* yang dikenal dengan sebutan SQL86. Dengan menggunakan SQL, kita dapat melakukan hal-hal berikut:

1. Memodifikasi struktur *database* .
2. Mengubah, mengisi, menghapus isi *database.*
3. Mentransfer data antara database yang berbeda.

SQL ada yang dikembangkan untuk PC dan ada juga yang dikembangkan untuk dapat mengakomodasi database yang sangat besar. Beberapa contohnya antara lain:

1. *Microsoft Accsess*

Digunakan untuk PC, sangat mudah dipakai dimana perintah SQL dapat langsung dimasukkan atau melalui fasilitas yang telah digunakan.

2. *Microsoft Query*

SQL yang dipaket dengan produk lain dari Microsoft Windows, yaitu Microsoft Visual Studio seperti Visual Basic dan Visual C++. Untuk terhubung dengan database lain menggunakan ODBC.

3. *Oracle*

Digunakan untuk perusahaan yang mengggunakan database besar (Eka Iswandy; 2017 : 73).

II.2.5 UML

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah elemen grafis yang bisa dikombinasikan menjadi diagram (Alfina & Harahap, 2019).

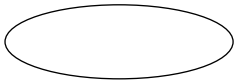
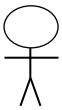
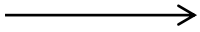
. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML yaitu *Use Case Diagram, Activity Diagram, Class Diagram* dan *Sequence Diagram*.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

5. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada Tabel II.1:

Tabel II.1. Simbol *Use Case*

Gambar	**Keterangan**
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.

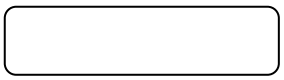
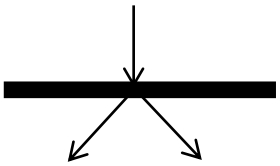
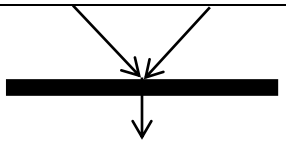
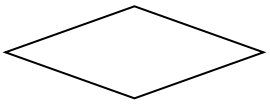
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Janiver W. Janis; 2020)

6. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada Tabel II.2:

Tabel II.2. Simbol *Activity Diagram*

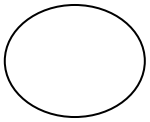
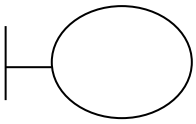
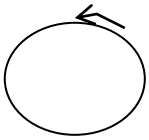
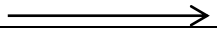
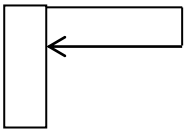
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Janiver W. Janis; 2020)

7. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada Tabel II.3:

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message,* simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

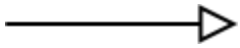
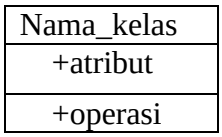
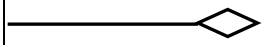
(Sumber : Janiver W. Janis; 2020)

8. Diagram Kelas (*Class Diagram)*

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility,* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut

dengan *multiplicity* atau kardinaliti (Ade Hendini, 2016 : 111). Simbol *class diagram* dan *multiplicity class diagram* dapat dilihat pada Tabel II.4 dan Tabel II.5.:

Tabel II.4. Simbol *Class Diagram*

Gambar	Keterangan
	*Generalization,* untuk menghubungkan antar kelas dengan arti umum-khusus. Jadi jika ada kelas dengan arti umum-khusus. Jadi jika ada kelas bermakna umum dan kelas bermakna khusus dapat menggunakan simbol ini.
	*Class,* untuk sebuah kelas pada struktur sistem. Penulisan tidak boleh menggunakan spasi. Simbol ini memiliki 3 susunan, yaitu kotak pertama adalah nama kelas, kedua atribut dan ketiga operasi.
	*Interface,* untuk simbol *interface* atau dalam bahasa indonesianya antar muka. Konsep yang digunakan pun sama dengan pemrograman berorientasi object (OOP).
	*Association,*digunakan untuk menghubungkan atau merelasikan kelas satu dengan kelas yang lainnya dengan makna umum.
	Directed Association, adalah relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
Tabel II.4. Simbol *Class Diagram (*Lanjutan)	
	*Aggregation,* adalah relasi antar kelas dengan makna semua bagian.
	Dependency, adalah relasi antar kelas dengan makna kebergantungan antar kelas.

(Sumber : Janiver W. Janis; 2020)

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1

n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Janiver W. Janis; 2020)