BAB II
KAJIAN PUSTAKA

2.1 Teori Dasar

2.1.1 Kecerdasan Buatan

Menurut Nugraha, Sugianto, & Prasetyo (2016) Kecerdasan buatan berasal dari kata *Artificial Intelligence* yang mengandung arti tiruan atau kecerdasan. Secara harfiah *Artificial Intelligence* adalah kecerdasan buatan. Kecerdasan buatan adalah salah satu bidang dalam ilmu komputer yang membuat komputer agar dapat bertindak dan sebaik seperti manusia (menirukan kerja otak manusia).

Menurut (Kornienko, Kornienko, Fofanov, & Chubik, 2015) *International Joint Conference on Computer Science* yang di adakan di Washington pada tahun 1969, yang mana konsep "*Artificial Intelligence*" (AI) di perkenalkan dan mempunyai hak untuk hidup. Pembelajaran tradisi intelektual AI dimulai oleh Aristotle dalam "*Physics*" yang melihat perbedaan diantara materi dan bentuknya. perbedaan ini adalah basis filosofis dari ide kalkulus simbolis atau abstraksi data. "*Logic*" oleh Aristotle mendekati ide dari kecerdasan buatan, karena beliau mengartikan pernyataan bahwa pembelajaran mengenai pemikiran adalah dasar dari pengetahuan. Aristotle adalah orang pertama yang mengalih ke hukum pemikiran "benar". Sampai pada hari ini, dunia kecerdasan buatan menjadi peran penting di beberapa lingkungan kehidupan manusia dan fokus pada penciptaan komputer dengan pengetahuan seperti manusia. Peneliti percaya bahwa universal

AI akan tiba di tempat yang memiliki pengkhususan kecerdasan buatan. Tetapi untuk menciptakan kecerdasan buatan tingkat lanjut, kita perlu menjawab beberapa pertanyaan terlebih dahulu, sebagai contohnya adalah pertanyaan mengenai peran pengetahuan dalam sistem AI.

2.1.2 Jaringan Saraf Tiruan

Menurut (Ekawati, 2015) Jaringan syaraf tiruan adalah sistem pemroses informasi yang memiliki karakteristik mirip dengan jaringan syaraf biologi. JST dibentuk sebagai generalisasi model matematika dari jaringan syaraf tiruan biologi, dengan asumsi bahwa :

1) Pemrosesan informasi terjadi pada banyak elemen sederhana (*neuron*).
2) Sinyal dikirimkan diantara *neuron-neuron* melalui penghubung-penghubung.
3) Penghubung antar *neuron* memiliki bobot yang akan memperkuat atau memperlemah sinyal.
4) Untuk menentukan output, setiap *neuron* menggunakan fungsi aktivasi (biasanya bukan fungsi linier) yang dikenakan pada jumlahan input yang diterima. Besarnya output ini selanjutnya dibandingkan dengan suatu batas ambang.

JST ditentukan oleh 3 hal:

1) Pola hubungan antar neuron (disebut arsitektur jaringan).
2) Metode untuk menentukan bobot penghubung (disebut metode *training* / *learning* / algoritma).
3) Fungsi aktivasi.

Jaringan Saraf Tiruan (JST) adalah representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia. Istilah buatan digunakan karena jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. JST merupakan suatu model komputasi yang menirukan cara kerja sisitem otak manusia. Seperti halnya jaringan saraf biologis, JST juga memiliki kemampuan untuk belajar dan beradaptasi terhadap masukan-masukan. JST menyerupai otak manusia dalam dua hal, yaitu: (1) Pengetahuan diperoleh jaringan melalui proses belajar; (2) Kekuatan hubungan antar sel syaraf (neuron) yang dikenal sebagai bobot-bobot sinaptik digunakan untuk menyimpan pengetahuan (Kusmaryanto, 2014).

2.1.3 Fuzzy Logic

Menurut (Nur Hasanah & Indriani Widiastuti, 2014) *Fuzzy* secara bahasa diartikan sebagai kabur atau samar-samar. suatu nilai dapat bernilai besar atau salah secara bersamaan. Dalam *fuzzy* dikenal derajat keanggotaan yang memiliki rentang nilai 0(nol) hingga 1(satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

Logika *Fuzzy* merupakan suatu logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Dalam teori logika *fuzzy* suatu nilai bisa bernilai benar atau salah secara bersama. Namun beberapa besar keberadaan dan kesalahan suatu tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* memiliki derajat keanggotaan dalam rentang 0 hingga 1. Berbeda dengan

logika *digital* yang hanya memiliki dua nilai 1 atau 0. Logika *fuzzy* digunakan untuk menterjemahkan suatu bahasa (*linguistic*), misalkan besaran kecepatan laju kendaraan yang diekspresikan dengan pelan, agak cepat, cepat dan sangat cepat. Logika *fuzzy* menunjukkan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti logika klasik (*crisp*)/ tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu anggota himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.

Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang input kedalam suatu ruang output, mempunyai nilai kontinyu. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama. Logika *Fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk linguistik, konsep tidak pasti seperti "sedikit", "lumayan" dan "sangat". Kelebihan dari teori logika fuzzy adalah kemampuan dalam proses penalaran secara bahasa (*linguistic reasoning*). Sehingga dalam perancangannya tidak memerlukan persamaan matematik dari objek yang akan dikendalikan. (Nur Hasanah & Indriani Widiastuti, 2014)

2.1.4 Sistem Pakar

Sistem pakar adalah sistem yang berusaha mengadopsi pengetahuan manusia (Pakar) ke komputer, sehingga komputer dapat menyelesaikan permasalahan tersebut layaknya seorang pakar. (Perwira, 2014)

Sistem pakar adalah program "*artificial intelligence*" (kecerdasan buatan) yang menggabungkan basis pengetahuan dengan mesin inferensi. (Nugraha et al., 2016)

Secara umum, sistem pakar ialah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer dan dirancang untuk memodelkan kemampuan menyelesaikan masalah layaknya seorang pakar pada bidang tertentu. Dengan menggunakan sistem pakar, orang awan pun dapat menyelesaikan atau mendapatkan informasi berkualitas yang sebenarnya hanya dapat diperoleh dengan bantuan para pakar dibidangnya.

Pengertian sistem pakar adalah program komputer yang menyimulasi penilaian dan perilaku manusia atau organisasi yang memiliki pengetahuan dan pengalaman ahli dalam bidang tertentu. Biasanya sistem seperti ini berisi basis pengetahuan yang berisi akumulasi pengalaman dan satu set aturan untuk menerapkan pengetahuan dasar untuk setiap situasi tertentu. Sistem pakar yang canggih dapat ditingkatkan dengan cara menambahkan basis pengetahuan atau set aturan. Diantara banyak pakar yang ada, yang terkenal adalah aplikasi bermain catur dan sistem diagnosis medis (Budihartono & Suhartono, 2014:132).

Menurut penelitian (Nurhayati, 2013:110) sistem pakar adalah sistem yang mampu menirukan penalaran seorang pakar agar komputer dapat menyelesaikan

masalah seperti yang biasa dilakukan oleh para ahli. Pengetahuan yang disimpan didalam sistem pakar umumnya diambil dari seorang manusia yang pakar dalam masalah tersebut. Peran penting seorang pakar dapat digantikan oleh program komputer yang pada prinsip kerjanya untuk memberikan solusi yang pasti seperti yang biasa dilakukan oleh pakar. Sistem pakar biasanya digunakan untuk konsultasi, analisis, diagnosis dan membantu mengambil keputusan.

Menurut (Budihartono & Suhartono, 2014:134) sistem pakar banyak digunakan pada aplikasi terkini dan kompleks karena:

1) Sistem pakar dapat bertindak sebagai konsultan, instruktur, atau pasangan/rekan.
2) Meningkatkan *availability* atau kepakaran tersedia pada semua perangkat komputer.
3) Mengurangi bahaya
4) Permanen
5) Pengetahuan dapat tidak lengkap, namun keahlian dapat diperluas sesuai kebutuhan. Program konvesional harus “lengkap” sebelum mereka dapat digunakan.
6) *Database* yang cerdas, sistem pakar dapat digunakan untuk mengakses *database* secara cerdas, misalnya data mining.

Secara umum ada dua bagian penting atau utama yang dibutuhkan dalam membuat aplikasi kecerdasan buatan yaitu basis pengetahuan *(knowlodge base)* dan mesin inferensi *(Inference Engine)*. Istilah sistem pakar berasal dari istilah *knowledge-based expert system*. Istilah ini muncul karena untuk memecahkan

masalah, sistem pakar menggunakan pengetahuan seorang pakar yang dimasukkan ke dalam komputer. Seseorang yang bukan pakar menggunakan sistem pakar untuk meningkatkan kemampuan pemecahan masalah, sedangkan seorang pakar menggunakan sistem pakar untuk *knowledge assistant.* Tujuan utama sistem pakar bukan untuk menggantikan kedudukan sorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya. Sistem pakar disusun oleh dua bagian utama, yaitu lingkungan pengembangan *(development environment)* dan lingkungan konsultasi *(consultation environment)* (Mukhtar & Samsudin, 2014:22).

2.2 Variabel

Variabel penelitian merupakan segala sesuatu yang berbentuk apa saja yang dimiliki orang, obyek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya (Sugiyono, 2014:38).

2.2.1 Berat Badan

Berat badan merupakan aspek biologis dari manusia yang merupakan bagian dari struktur tubuh dan postur tubuh. Seseorang yang berbadan besar dan bertubuh tinggi dapat dipastikan mempunyai berat badan yang besar. (Agus Mardika, 2014)

Kelebihan berat badan alias gemuk merupakan akibat dari masukan kalori yang berlebihan dibandingkan dengan kalori yang dibutuhkan tubuh. (Fitriah & K D, 2016)

2.2.2 Tinggi Badan

Tinggi badan merupakan parameter yang penting bagi segenap jasad manusia yang terdiri dari badam, anggota badan yang diukur dari telapak kaki sampai kepala bagian atas. (Agus Mardika, 2014)

2.2.3 Indeks Massa Tubuh

Indeks Massa Tubuh merupakan rumus matematis yang berkaitan dengan lemak tubuh orang dewasa, yang dinyatakan sebagai berat badan (dalam kilogram) dibagi dengan kuadat tinggi badan (dalam meter). Rumus ini hanya cocok diterapkan pada usia 19-70 tahun, berstruktur tulang belakang normal, bukan atlet atau binaragawan, dan bukan juga wanita menyusui.

IMT yang normal antara 18-25. Seseorang dikatakan kurus apabila IMT <18 dan dikatakan gemuk bila IMT > 25. Bila IMT > 30 berarti orang tersebut menderita obesitas dan perlu diwaspadai karena biasanya pada orang obesitas akan dijumpai beberapa penyakit degeneratif seperti Diabetes Melitus, hipertensi, hiperkolesterol dan kelainan metabolik lainnya.

2.2.4 Usia

Menurut (Perwira, 2014) Ada beberapa cara untuk menentukan jumlah kalori yang dibutuhkan diabetisi. Diantaranya dengan memperhitungkan berdasarkan kebutuhan kalori basal yang besarnya 25-30 kalori/kg BB ideal, ditambah dan

dikurangi bergantung pada faktor usia, jenis kelamin, Tinggi Badan (TB), Berat Badan (BB) dan aktifitas fisik.

2.2.5 Kalori

Kalori merupakan satuan pengukuran yang dipakai untuk menyatakan nilai energi dari makanan. (Fitriah & K D, 2016)

Kalori *in* adalah jumlah kalori dari makanan yang masuk kedalam tubuh kita. Sementara, kalori *out* adalah jumlah kalori atau energi yang keluar dari tubuh kita. (Hartanto Alvin, 2018)

2.3 *Software* Pendukung

2.3.1 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. (Murtiwiyati & Lauren, 2013)

Android 6: Marshmallow kini sudah tersedia untuk perangkat-perangkat Nexus dan diprediksi akan tersedia untuk perangkat-perangkat utama sebelum penghujung musim serta untuk perangkat-perangkat lainnya pada pertengahan 2016. Marshmallow menyingkap beberapa modifikasi yang memiliki pengaruh besar. Model otorisasi App kini berupa *opt-in* (memberikan otorisasi yang spesifik sesuai permintaan), berbalik dengan *opt-out* (semuanya diberi otorisasi lalu menggunakan App Ops untuk memberikan izin bagi pihak tertentu). Mode *doze* memungkinkan perangkat untuk melakukan hibernasi saat sedang tidak aktif, memutus daya listrik secara bertahap hingga tidak ada sama sekali. Unit pendukung sensor sidik jari kini telah tersemat didalam sistem operasi tanpa campur tangan dari *vendor* dan USB C kini telah didukung sepenuhnya. Selain itu, Marshmallow memungkinkan pengguna untuk melakukan format micro SD card selayaknya *internal storage* dan berbagi tingkat keamanan internal yang sama persis. (Khanna & Singh, 2016)

2.3.2 Java

Java adalah bahasa berorientasi objek yang dapat digunakan untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, serta aplikasi untuk perangkat-perangkat cerdas yang dapat berkomunikasi lewat internet atau jaringan komunikasi. Dalam Java ada 2 (dua) jenis program berbeda, yaitu aplikasi dan *applet.* Aplikasi adalah program yang biasanya disimpan dan dieksekusi dari komputer lokal sedangkan *applet* adalah program yang biasanya disimpan pada komputer yang jauh, yang dikoneksikan pemakai lewat *web browser*.

Java bukan turunan langsung dari bahasa manapun. OOP *(object oriented programming)* adalah cara yang ampuh dalam pengorganisasian dan pengembangan perangkat lunak. (Lengkong, Sinsuw, & Lumenta, 2015)

2.3.3 Android Studio

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) khusus untuk membangun aplikasi yang berjalan pada platform android. Android Studio ini berbasis pada IntelliJ IDEA, sebuah IDE untuk bahasa pemrograman Java. Bahasa pemrograman utama yang digunakan adalah Java, sedangkan untuk membuat tampilan atau layout, digunakan bahasa XML. Android Studio juga terintegrasi dengan *Android Software Development Kit* (SDK) untuk *deploy* ke perangkat android. (Fikri, Herumurti, & Rahman, 2016)

Android Studio 3.1: pokok IDE Android Studio telah diperbaharui dengan perkembangan dari IntelliJ IDEA melalui pembaharuan 2017.3.3. Perkembangan termasuk penanganan aliran analisis untuk koleksi-koleksi dan kumpulan *String*, meningkatkan inferensi *nullability*, perbaikan cepat yang baru, dan banyak lagi. Android Studio 3.1 mengandung Kotlin versi 1.2.30. Kode Kotlin kini dianalisis dengan pengecekan barisan perintah. (“Android Studio Release Notes | Android Developers,” 2018)

2.3.4 Android SDK (*Software Development Kit*)

Android SDK mencakup perangkat *tools* pengembangan yang komprehensif. Android SDK terdiri dari debugger, libraries, handset emulator, dokumentasi, contoh kode program dan *tutorial*. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk desktop modern), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista. Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau lebih. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan *plugin Android Development Tools* (ADT), dengan ini pengembang dapat menggunakan IDE untuk mengedit dokumen Java dan XML serta menggunakan peralatan command line untuk menciptakan, membangun, melakukan debug aplikasi Android dan pengendalian perangkat Android (misalnya *reboot*, menginstal paket perangkat lunak). (Sulihati & Andriyani, 2016)

2.3.5 Android JDK (*Java Development Kit*)

Java adalah sebuah teknologi yang diperkenalkan oleh Sun Microsystems pada pertengahan tahun 1990. Menurut definisi Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada *computer standalone* ataupun pada lingkungan jaringan. Untuk membuat program Java dibutuhkan *compiler* dan *interpreter* untuk program Java berbentuk *Java Development Kit* (JDK) yang diproduksi oleh Sun Microsystems. Sebelum memulai instalasi Android SDK, terlebih dahulu kita harus melakukan instalasi JDK dikomputer. (Harni Kusniyanti, 2016)

2.3.6 iOS

iOS diturunkan dari OS X, yang memiliki fondasi Darwin dan karena itu iOS merupakan sistem operasi Unix. iOS adalah versi bergerak dari sistem operasi OS X yang dipakai di komputer-komputer Apple. Versi iOS dulunya berawal dari versi iPhone OS 1.0 (*initial release*) Juni 2007 untuk iPhone dan iPod Touch hingga iPhone 3.1 - 3.2 versi terakhir dari nama iPhone OS pada September 2009. Pada Juni 2010 iPhone OS berganti nama menjadi iOS yang diterapkan pada iPhone, iPod Touch, iPad yang telah menambahkan beberapa fitur yang menarik dan lebih interaktif sehingga menarik minat pengguna untuk menggunakannya. (Apridianto, Satoto, & Windasari, 2016)

iOS 11.3: iOS 11.3 memperkenalkan fitur baru, meliputi ARKit 1.5 dengan dukungan untuk pengalaman realitas tertambah yang lebih terasa nyata, Kesehatan Baterai iPhone (*Beta*), Animoji baru untuk pengguna iPhone X, dan lainnya. Pembaruan ini juga disertai dengan peningkatan stabilitas dan perbaikan *bug*. (“Mengenai pembaruan iOS 11 - Apple Support,” 2018)

2.3.7 Objective-C

Objective-C adalah sebuah bahasa program berbasis *open source* dan *runtime* terasosiasi dibawah naungan Apple untuk para pengembang program pada *platform* OS X dan iOS. Objective-C mengabstraksi banyak aspek yang menyulitkan perangkat lunak sistem pemrograman C++ namun tetap mempertahankan beberapa semantik atau sintaks yang mudah dipahami. *Runtime* menyediakan dukungan yang

sangat fleksibel untuk *function calls*, *class instantiation*, penggunaan variabel dan anggota *class*. Sebagai contoh, akses semua *class* dan anggota *class* dapat dilakukan berdasarkan sebuah *string name* pada *runtime*. Pada kasus yang sama, *class* apa saja dapat menunjuk lokasi *class* dan *instance* yang lainnya pada saat *runtime* berdasarkan *string description* (Case & Richard, 2016).

2.3.8 Xcode

Seperti Android Studio, Xcode merupakan sebuah IDE atau dikenal *suite of tools* yang diciptakan oleh perusahaan Apple yang dapat digunakan untuk mengembangkan aplikasi di platformnya, yakni Mac OS X, iOS dan Mac TV OS. Tidak seperti fitur dari apple yang justru eksklusif, Apple menyediakan Xcode dengan gratis untuk digunakan oleh siapa saja, dengan syarat hanya dapat beroperasi pada platform yang dimiliki Apple sendiri yaitu Mac OS X, tidak seperti Android studio, Xcode tidak menyediakan Gradle untuk berintegrasi dengan perangkat genggam untuk menjalankan aplikasinya, karena Xcode sendiri hanya dapat membuat aplikasi pada platform Apple, untuk itu semua perangkat lunak yang di perlukan sudah tersedia di compiler Xcode sendiri. Xcode juga memiliki simulator yang digunakan untuk men-*DEBUG* aplikasi yang kita buat, untuk melihat hasil yang dijalankan sesuai dengan tampilan perangkat yang ada.

Gradle pada Android Studio digunakan juga untuk menangani library yang sudah pernah di bahas di subbab sebelumnya, sedangkan Xcode memerlukan aplikasi perangkat lunak pihak ketiga untuk mengimplementasi library pada program aplikasi yang sedang dibuat, seperti Cocoapods dan Carthage. Bahasa

pemrograman yang di gunakan di Xcode juga dilisensi secara eksklusif dari Apple sendiri yaitu Objective-C dan Swift, bahasa yang lebih banyak digunakan adalah Swift untuk zaman sekarang. (Case & Richard, 2016)

XCode 9.3: Mode Testing 64-Bit baru, pengembang kini bisa menggunakan mode testing 64-bit yang baru pada macOS 10.13.4 untuk percobaan perangkat lunak untuk kesesuaian 64-bit. Fitur baru *Energy organizer* akan menampilkan catatan yang dihasilkan ketika aplikasi atau ekstensi aplikasi melewati batas wajar kekuatan CPU di latar depan maupun di latar belakang yang menguras baterai pada perangkat pengguna. (“Xcode Release Notes,” 2018)

2.3.9 Web

Secara sederhana *world wide web* adalah jaringan komputer yang menyediakan berbagai layanan informasi (disebut *server*) dan didalamnya terdapat sekumpulan komputer yang saling terintegrasi dengan jaringan telekomunikasi yang cepat. Dalam *world wide web* dikenal istilah *client server,* merupakan hubungan komunikasi yang dibangun antara *website* sebagai sumber informasi dan *client* sebagai pengguna komputer. (Yatini, 2014)

2.3.10 HTML

HTML atau (*Hyper Text Markup Language*) adalah bahasa standar yang digunakan untuk menampilkan halaman *web*. Yang bisa dilakukan dengan HTML yaitu: mengatur tampilan dari halaman *web* dan isinya, membuat *table* dalam

halaman *web*, mempublikasikan halaman *web* secara *online*, membuat *form* yang bisa digunakan untuk menangani registrasi dan transaksi via *web*, menambahkan objek-objek seperti citra, audio, video, animasi, java aplet dalam halaman *web*, serta menampilkan area gambar (*canvas*) di browser. (Fridayanthie & Mahdiati, 2016)

HTML 5 merupakan teknologi inti dari Internet adalah bahasa *markup* untuk penataan dan penyajian konten *world wide web*. Tujuan utama HTML 5 adalah meningkatkan bahasa dengan dukungan multimedia yang tetap mudah dibaca dan dimengerti.

Fitur baru HTML 5 antara lain.

1) Unsur <canvas> untuk menggambar 2D.
2) Unsur <video> dan <audio> untuk media pemutaran.
3) Dukungan untuk penyimpanan lokal.
4) Konten baru dengan elemen spesifik seperti <article>, <footer>, <header>, <nav>, <section>. (Yatini, 2014)

2.3.11 PHP

PHP singkatan dari *Perl Hypertext Preprocessor* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang berintegrasi dengan HTML dan berada pada *server* (*server side* HTML *embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman *web* dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client.* Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru/*up to date*. Semua *script* PHP dieksekusi pada *server* dimana *script* tersebut dijalankan.

PHP dirancang untuk dapat bekerja sama dengan *database server* dan dibuat sedemikian rupa sehingga pembuatan dokumen HTML yang dapat mengakses *database* menjadi begitu mudah. Tujuan dari bahasa *scripting* ini adalah untuk membuat aplikasi dimana aplikasi tersebut dibangun oleh PHP pada umumnya akan memberikan hasil kepada *web browser*, tetapi proses keselurahannya dijalan di *server*. (Fridayanthie & Mahdiati, 2016)

2.3.12 XAMPP

XAMPP adalah sebuah *software* yang berfungsi untuk menjalankan *website* berbasis PHP dan menggunakan pengolah data MYSQL di komputer lokal. XAMPP berperan sebagai *server web* pada komputer lokal. XAMPP juga dapat disebut sebuah *Cpanel server virtual*, yang dapat membantu melakukan *preview* sehingga dapat dimodifikasi *website* tanpa harus *online* atau terakses dengan internet.

Sebagai informasi kata XAMPP merupakan singkatan dari:

X: Berarti program ini dapat dijalankan diberbagai platform, misalnya Windows, Linux, mac OS, dan Solaris.

A: **Apache**, merupakan aplikasi *web server*, dan bertugas untuk menghasilkan halaman *web* yang benar kepada *user* berdasarkan kode PHP yang dituliskan oleh pembuat halaman *web*. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *database* diakses terlebih dahulu (misalnya dalam MySQL) untuk mendukung halaman *web* yang dihasilkan.

M: **MySQL**, merupakan aplikasi *database server*. Pengembangnya disebut *Structured Query Language* (SQL). SQL merupakan bahasa terstruktur yang digunakan untuk mengolah *database* beserta isinya. Pengguna dapat memanfaatkan MySQL untuk menambahkan, mengubah dan menghapus data yang berada dalam *database*.

P: **PHP**, bahasa pemrograman lainnya yang serupa, dan lain sebagainya. (Fridayanthie & Mahdiati, 2016)

2.3.13 MySQL

MySQL (*My Structure Query Language*) adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengelolaan datanya. MySQL bersifat *open source* dan menggunakan SQL (*Structured Query Language*). MySQL biasa dijalankan diberbagai *platform* misalnya *windows*, *Linux*, dan lain sebagainya. (Fridayanthie & Mahdiati, 2016)

2.3.14 Laravel

Laravel adalah sebuah MVC *web development framework* yang didesain untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan dan perbaikan serta meningkatkan produktifitas pekerjaan dengan sintak yang bersih dan fungsional yang dapat mengurangi banyak waktu untuk implementasi. Laravel merupakan *framework* dengan versi PHP yang *up-to-date*,

karena Laravel mensyaratkan PHP versi 5.3 keatas. Laravel merupakan farmework PHP yang menekankan pada kesederhanaan dan fleksibilitas pada desainnya. (Artikel, Bisnisbisnis, & Luthfi, 2017)

2.3.15 React Native

React Native adalah sebuah *framework* Javascript yang digunakan untuk membuat aplikasi berbasis mobile pada iOS dan Android yang nyata dan dapat melakukan *rendering* lokal. Dikembangkan dari React, *library* Javascript pada Facebook yang membangun antar-muka pengguna, namun menargetkan *mobile platform* (bukan browser). Dengan kata lain: pengembang web sekarang dapat membuat aplikasi berbasis mobile yang terlihat dan terasa "alami" (*native*), semuanya berkat sebuah *library* Javascript yang telah kita ketahui dan sukai. Dengan tambahan, karena kebanyakan *code* yang anda buat dapat dibagikan antar platform, React Native memudahkan pengembangan program secara simultan untuk Android maupun iOS.

Hampir sama dengan React pada web, aplikasi-aplikasi React Native ditulis dengan perpaduan JavaScript dan XML-esque *markup* yang dikenal dengan JSX. Pada implementasinya, "*bridge*" React Native memanggil API *rendering* lokal (*native rendering*) pada Objective-C (untuk iOS) atau Java (untuk Android). Dengan begitu aplikasi akan melakukan *render* menggunakan komponen UI pada mobile yang *real* (nyata), bukan tampilan web (webviews) dan akan terlihat dan terasa seperti aplikasi *mobile* lainnya. React Native juga mengekspos antar-muka JavaScript untuk API pada *platform*, jadi aplikasi React Native dapat mengakses

fitur-fitur pada *platform* seperti kamera ponsel ataupun lokasi pengguna. (Chinnathambi, 2016)

2.3.16 UML

Pada perkembangan Teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan Teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan koneksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu di telaah dimanakah UML digunakan dan hal apa yang ingin divisualkan. Secara analogi jika dengan bahasa yang kita gunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalah hal yang salah. Sistem informasi bukanlah ilmu pasti, maka juka ada banyak

perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar. (Ariani Sukamto & Shalahuddin, 2015)

2.3.16.1 *Use Case Diagram*

Menurut (Ariani Sukamto & Shalahuddin, 2015) *use case* atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

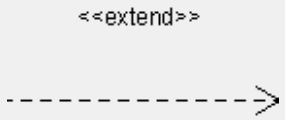
Menurut (Ariani Sukamto & Shalahuddin, 2015) Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case.*

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor

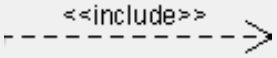
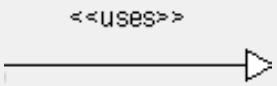
Tabel 2.1 Simbol-simbol diagram use case

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama *use case*
Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor

Tabel 2.1 Lanjutan

Asosiasi / association 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan

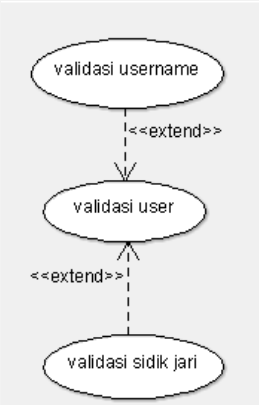
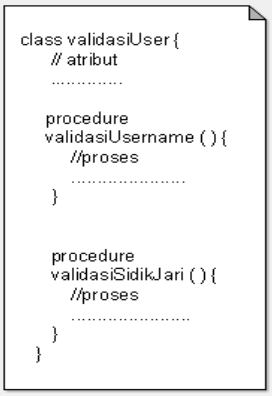
Tabel 2.1 Lanjutan

Generalisasi / generalization 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
Menggunakan / *include* / *uses*  	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai include di *use case*: 1. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan 2. *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan

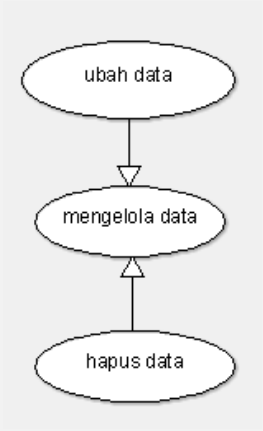
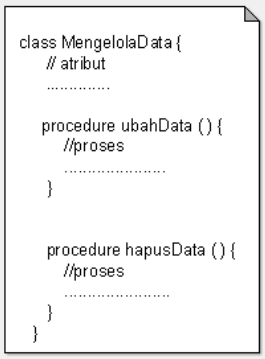
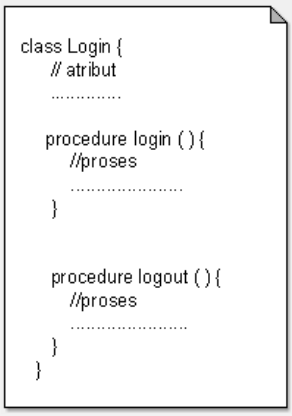
Sumber : (Ariani Sukamto & Shalahuddin, 2015)

Menurut (Ariani Sukamto & Shalahuddin, 2015) *use case* nantinya akan menjadi kelas proses pada diagram kelas segingga perlu dipertimbangkan penamaan yang dilakukan apakah sudah layak menjadi kelas atau belum sesuai dengan aturan pendefinisian kelas yang baik. Berikut adalah aturan perubahan *use case* yang layak menjadi kelas proses sehingga layak dijadikan sebagai *use case :*

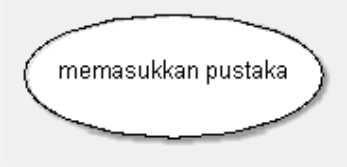
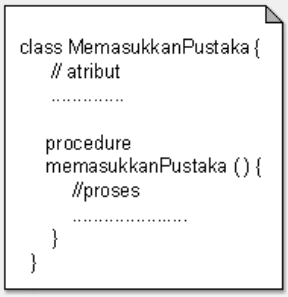
Tabel 2.2 Aturan perubahan *use case*

Hubungan	Keterangan
Ekstensi / *extend* 	Pada hubungan ekstensi maka dapat hanya diambil *use case* induknya yang dijadikan kelas dengan metode berupa *use case* ekstensinya 

Tabel 2.2 Lanjutan

Generalisasi / generalization 	Pada hubungan generalisasi maka dapat hanya diambil *use case* umumnya yang dijadikan kelas dengan metode berupa *use case* khususnya 
Use case yang berdiri sendiri 	Metode yang mungkin bisa ada di dalam kelas proses login adalah sebagai berikut: 

Tabel 2.2 Lanjutan

Use case yang kurang tepat sebagai sebuah *use case* yang berdiri sendiri 	Kurang tepat karena kelasnya akan menjadi:  Kelas yang hanya terdiri dari satu metode sebenarnya kurang efisien

Sumber:(Ariani Sukamto & Shalahuddin, 2015)

2.3.16.2 *Activity Diagram*

Menurut (Ariani Sukamto & Shalahuddin, 2015) Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

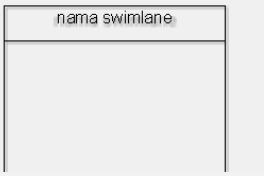
1) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan prises bisnis sistem yang didefinisikan.
2) Urutan atau pengelompokan tampilam dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang oerlu didefinisikan kasus ujinya
4) Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.3 Simbol-simbol *diagram* aktivitas

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

Tabel 2.3 Lanjutan

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
Swimlane  Atau  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber: (Ariani Sukamto & Shalahuddin, 2015)

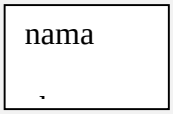
2.3.16.3 *Sequence Diagram*

Menurut (Ariani Sukamto & Shalahuddin, 2015) Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk meggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

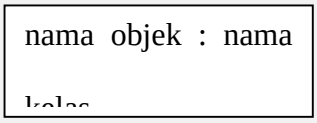
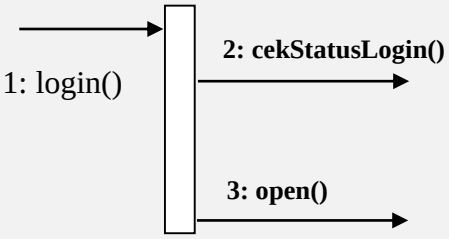
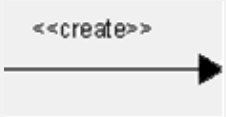
Menurut (Ariani Sukamto & Shalahuddin, 2015) Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Berikut adalah simbol-simbol yang ada pada diagram sekuen:

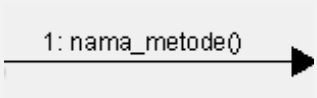
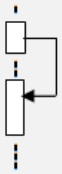
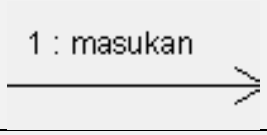
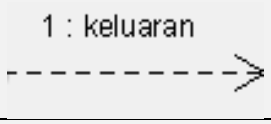
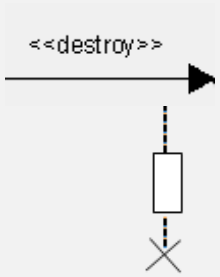
Tabel 2.4 Simbol-simbol *diagram sequence*

Simbol	Deskripsi
Aktor  Atau  Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor

Tabel 2.4 Lanjutan

Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek
Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cekStatusLogin() dan open() dilakukan di dalam metode login() Aktor tidak memiliki waktu aktif
Pesan tipe create 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Tabel 2.4 Lanjutan

Pesan tipe call 	 1: nama_metode() Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe send 	Menyatakan bahwa suatu objek mengirim data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Sumber: (Ariani Sukamto & Shalahuddin, 2015)

Menurut (Ariani Sukamto & Shalahuddin, 2015) penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan,

pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu. Semua metode di dalam kelas harus ada di dalam diagram kolaborasi atau sekuen, jika tidak ada berarti perancangan metode di dalam kelas itu kurang baik. Hal ini dikarenakan ada metode yang tidak dapat dipertanggungjawabkan kegunaannya.

2.3.16.4 *Class Diagram*

Menurut (Ariani Sukamto & Shalahuddin, 2015) Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas dan Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang di rancang dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau *programmer* dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1) Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan

2) Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai

3) Kelas yang diambil dari pendifinisian *use case* (*controller*)

   Kelas yang Menangani fungsi fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4) Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut multivalue pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungawabkan atau tetap ada di dalam perancangan kelas. Misalkan dalam tabel TTelepon TAnggota pada studi kasus maka perancangan kelas dapat mengandung kelas Telepon dan Anggota atau perancangan kelas hanya terdiri dari kelas Anggota di mana di dalamnya ada sebuah atribut berupa larik (*array*) bertipe *string* dengan nama telepon.

5) Ilustrasinya dapat dilihat pada gambar berikut

Gambar 2.1 Perancangan kelas data untuk tabel dari atribut *multivalue*

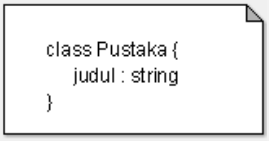
Menurut (Ariani Sukamto & Shalahuddin, 2015) Hal terpenting adalah kolom telepon tetap dapat diakses dari perancangan kelas. Kelas data biasanya adalah kelas yang terkait dengan pengaksesan tabel pada basis data sesuai dengan nama kelasnya, misalnya kelas Anggota maka akan digunakan untuk mengakses tabel yang menyimpan data anggota.

Jenis-jenis kelas di atas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas (*utility class*) seperti Koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

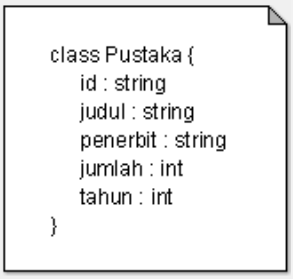
Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa ang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang lemah.

Perhatikan juga jika sebuah kelas hanya terdiri dari atribut saja, atau sebuah atribut saja, atau hanya sebuah metode. Berikut pertimbangan dalam membuat kelas:

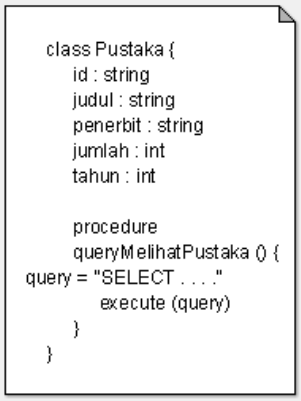
Tabel 2.5 Perancangan kelas data untuk tabel dari atribut *multivalue*

Pertimbangan	Keterangan
Sebuah kelas yang hanya berisi sebuah atribut	Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan satu atribut adalah sebagai berikut:  Sebuah berkas atau *file* hanya berisi seperti di atas tentu saja tidak efisien. Cara memperbaiki rancangan adalah dengan menggabungkan satu atribut tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.

Tabel 2.5 Lanjutan

Sebuah kelas yang hanya berisi atribut	Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan hanya berisi atribut saja adalah sebagai berikut:  Sebuah berkas atau *file* hanya berisi seperti di atas tentu saja tidak efisien. Cara memperbaiki rancangan adalah dengan menggabungkan atribut tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.

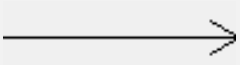
Tabel 2.5 Lanjutan

Sebuah kelas yang hanya berisi sebuah metode	Kelas seperti ini kurang efisien sehingga perlu dipikirkan kembali perancangan yang dilakukan, karena secara fisik kelas dengan hanya berisi sebuah metode saja adalah sebagai berikut:  Sebuah berkas atau *file* hanya berisi seperti di atas tentu saja tidak efisien. Cara memperbaiki rancangan adalah dengan menggabungkan atribut dan metode tersebut ke kelas lain yang memiliki keterkaitan lebih dekat.

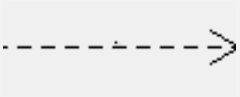
Sumber: (Ariani Sukamto & Shalahuddin, 2015)

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.6 Simbol-simbol diagram kelas

Simbol	Deskripsi
Kelas 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Antarmuka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association* 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / directed association 	Relasi antarakelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus)

Tabel 2.6 Lanjutan

Kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas
Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

Sumber: (Ariani Sukamto & Shalahuddin, 2015)

2.4 Penelitian Terdahulu

1) Penelitian Rifki Indra Perwira (2014) dengan judul "Purwarupa system pakar untuk menentukan jumlah kalori diet bagi penderita diabetes mellitus" menggunakan metode *forward chaining* yang mana sistemnya dirancang dengan beberapa masukan seperti tinggi badan, berat badan, jenis kelamin, usia, aktifitas dan kategori badan dengan aturan broca untuk perhitungan menu diet, dan memberi *output* berupa jumlah kalori.
2) Penelitian Nurul Annisa Shaleha, Dini Destiani (2015) dengan judul "Perancangan system pakar identifikasi jenis makanan diet sehat bagi penderita hiperkolesterol" yang menggunakan metode *forward chaining* dan outputnya merupakan penentuan makanan diet sehat bagi penderita hiperkolesterol.

3) Penelitian Rofiqoh Dewi (2014) dengan judul "Sistem pakar diet sehat bertipe genotype menggunakan metode certainty factor" yang menghasilkan informasi diet berdasarkan *genotype*, memberikan prosedur, olahraga serta asupan dalam menunjang diet berdasarkan *genotype* tersebut secara optimal sehingga dapat diakses dengan mudah kepada masyarakat luas
4) Penelitian Firna Yenila, S.Kom, M.Kom (2012) dengan judul "Sistem pakar pengaturan diet sehat berdasarkan golongan darah" yang menggunakan metode *forward chaining* yang menghasilkan bahwa jenis golongan darah dan berat badan relative sangat mempengaruhi dalam penentuan menu sehat bagi *user* dalam melaksanakan program diet.
5) Penelitian Titik Lusiani, Anita Qoirah (2010) dengan judul "Sistem pakar untuk menentukan menu makanan sehat pada penderita diabetes mellitus" yang melibatkan jenis aktifitas dan status gizi pengguna untuk menentukan menu makanan sehat untuk penderita diabetes mellitus.

2.5 Kerangka Pemikiran

Kerangka pemikiran peneliti dalam penelitian ini dimulai dengan penginputan biodata pribadi dari pengguna aplikasi, yang berupa nama, nama pengguna, kata sandi, tinggi badan, berat badan, usia, dan jenis kelamin. Kemudian akan di validasi terlebih dahulu untuk mengetahui apakah pengguna tersebut pantas untuk menjalankan program diet atau tidak. Jika hasil validasi berupa pengguna tersebut pantas untuk menjalankan program diet, maka akan dikalkukasikan nilai

maksimal kalori per hari untuk pengguna tersebut. Maka aplikasi akan merekomendasikan jenis-jenis makanan yang sesuai berdasarkan nilai kalorinya, dari rekomendasi dan nilai kalorinya, pengguna dapat menyesuaikan pola makan untuk mencapai berat badan ideal.

Gambar 2.2 Kerangka Pemikiran