BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian sebelumnya yang pernah dilakukan oleh Wahyu Joni Kurniawan yang berjudul "Sistem Pendukung Keputusan Seleksi Atlet *Poomsae* Taekwondo Dengan Metode *Analytic Hierarchy Process*". Kriteria yang menjadi acuan dalam proses seleksi atlet poomsae, seperti kedisiplinan, sikap, kerajinan, teknik, daya tahan, keindahan, sabuk, tim, dan beberapa sub kriteria dari sabuk, yakni hijau, biru, merah, dan hitam. Kriteria kuantitatif dan kualitatif yang digunakan cukup banyak sehingga menyulitkan pelatih dalam memilih atlet terbaik. Sistem yang dibuat berfungsi untuk membantu pelatih dalam memilih anggota yang lebih tepat untuk mengikuti pertandingan poomsae, dengan pemanfaatan metode ini bisa memberikan motivasi bagi para anggota yang ingin menjadi calon atlet agar bisa memaksimalkan diri agar bisa sesuai dengan kriteria yang sudah ditentukan[3].

Penelitian yang lain dilakukan oleh Ai Musrifah yang berjudul "Penerapan Aplikasi Sistem Pendukung Keputusan Terhadap Perkembangan Balita Berbasis Android" di lakukan di posyandu sekar arum yang berada di Kecamatan Tanggeung. Posyandu saat ini masih terdapat beberapa kendala seperti perkembangan balita dikarenakan kurangnya penyampaian informasi mengenai perkembangan balita, hal ini membuat orangtua sulit mengatasi gejala mengenai balita sendiri, sulit mencari informasi mengenai balita seperti info perawatan, penyakit dan imunisasinya, selain itu orangtua masih sering lalai memperhatikan perkembangan balitanya. Sistem ini digunakan untuk membantu orangtua dan petugas kesehatan posyandu dalam mengambil keputusan terhadap informasi mengenai perkembangan balita. Sistem ini bertujuan untuk memberikan kemudahan dalam menunjang penyajian informasi mengenai data perkembangan balita, konsultasi, *reminder*, dan informasi seputar balita. Kriteria yang dipakai dalam sistem ini seperti umur, jenis kelamin, tinggi badan, dan berat badan. Sistem ini dibangun menggunakan *platform* android studio dengan bahasa pemrograman java[4].

Penelitian selanjutnya dengan judul "Sistem Pendukung Keputusan Seleksi Atlet Taekwondo Untuk Kenaikan Sabuk Hitam Dengan Metode *Simple Additive Weighting*" yang disusun oleh Muthia Gidriani Maelan, dan A.S. Purnomo. Sistem ini digunakan untuk

membantu seleksi atlet taekwondo dalam mengikuti kenaikan sabuk hitam, dan mengukur tingkat akurasi sistem seleksi atlet taekwondo untuk kenaikan sabuk hitam menggunakan sistem pendukung keputusan dengan sistem pemilihan manual. Pengambilan keputusan dilakukan selama ini yaitu dengan pendekatan sistematis terhadap permasalahan melalui proses pengumpulan data menjadi informasi serta ditambah dengan faktor-faktor yang perlu dipertimbangkan dalam pengambilan keputusan. Kriteria yang digunakan dalam sistem ini antara lain fisik, *poomsae*, gerakan, tandangan, *kyorugi*, dan *kyupa*[5].

Penelitian kali ini yang membedakan dengan penelitian yang sebelumnya adalah kriteria yang digunakan dalam sistem ini yaitu MFT (*Multistage Fitness Test*), lari 300 meter, *push up, sit up, shuttle run*, tendangan sabit 5 detik, tendangan sabit 10 detik, dan tendangan 1 menit. Sistem ini dapat memilih atlet terbaik, menampilkan perangkingan, dan nilai perangkingan dengan menggunakan metode *simple additive weighting* dengan basis android hal ini akan mempermudah dalam pengolahan dan penyampaian informasi baik bagi admin, pelatih, maupun pemilik.

2.2 Landasan Teori

2.2.1 Sistem

Sekumpulan elemen yang saling berkaitan dan terpadu guna mewujudkan suatu tujuan disebut dengan sistem. Suatu elemen dikatakan menjadi bagian sistem jika elemen tersebut memberikan manfaat dalam mencapai tujuan yang sama[6].

2.2.2 Metode Pengembangan Sistem Metode *Prototyping*

Metode *prototyping* adalah sebuah metode pendekatan dalam pengembangan sistem perangkat lunak dengan membuat sebuah program secara cepat dan bertahap sehingga dapat langsung dievaluasi oleh pengguna atau *client*. Metode ini adalah model atau contoh awal dari sistem yang berisi fitur dan fungsionalitas yang akan dibangun. Tujuan utama dari metode *prototyping* adalah untuk memberikan gambaran aplikasi visual dan nyata yang akurat kepada pengguna atau *client* tentang bagaimana sistem akan beroperasi dan memenuhi kebutuhan mereka. Metode *prototyping* memberikan sebuah pendekatan antara *developer* dengan *client* untuk terus berkomunikasi selama pembuatan aplikasi berlangsung sehingga *developer* akan mendapatkan *feedback* dari *client* yang akan digunakan untuk memperbaiki aplikasi yang dibuat[7]. Roger

S. Pressman dalam bukunya *Software Engineering: A Practitioner's Approach* menyatakan terdapat lima tahapan yang harus dilakukan pada metode *prototyping*. Penjelasan dari lima tahapan dalam metode *prototyping* sebagai berikut[8]:

1. *Communication*

Tahap pertama adalah komunikasi antara pengembang perangkat lunak dan *client* secara aktif. Tujuannya untuk mengidentifikasikan, mengumpulkan dan memahami berbagai kebutuhan sistem yang akan di rancang secara mendalam. Dengan melibatkan para *client* yang bersangkutan selama proses perancangan akan memberikan hasil yang tepat sesuai keinginan *client* yang bersangkutan. Proses analisa kebutuhan menggunakan teknik pengumpulan data yaitu studi pustaka dan studi lapangan.

2. *Quick Plan*

Tahap kedua adalah perencanaan awal perangkat lunak dengan cepat sesuai dengan spesifikasi kebutuhan pengguna atau *client* berdasarkan data yang telah dikumpulkan pada tahap *communication*. Adapun perancangan yang dibuat seperti perencanaan desain antarmuka, perencanaan basisdata, dan kebutuhan pendukung lainnya pada proses ini.

3. *Modeling Quick Design*

Tahap ketiga berfokus pada pembuatan rancangan *prototype* awal disesuaikan dengan kebutuhan pengguna yaitu; *input*, proses dan *output*, serta mencakup fitur dan fungsionalitas dasar yang relevan untuk mencapai tujuan utama sistem yang telah direncanakan pada tahap sebelumnya. Model cepat ini kemudian dievaluasi oleh pengguna atau *client* untuk memastikan bahwa pengembang telah memahami dengan benar kebutuhan dan persyaratan yang direncanakan.

4. *Construction of Prototype*

Pada tahap ini, *prototype* yang telah dievaluasi dan disetujui akan dikembangkan lebih lengkap dan diperluas untuk mencakup semua fitur dan fungsionalitas yang diharapkan. Pengembang membangunan perangkat lunak berfokus terhadap aspek utama perangkat lunak dengan maksud agar proses selanjutnya pengembang bisa dengan cepat mendapatkan *feedback* dari *client* tentang perangkat lunak yang dibuat.

5. *Deployment Delivery & Feedback*

Tahap terakhir adalah penyebaran perangkat lunak yang telah selesai dikembangkan. Setelah perangkat lunak dianggap siap dan telah melewati fase pengujian, maka perangkat lunak akan diperkenalkan dan

digunakan oleh pengguna sesuai dengan kebutuhan. Tahap ini juga mencakup pelatihan bagi pengguna untuk memastikan bahwa mereka dapat menggunakan sistem dengan efektif. Perangkat lunak yang telah diserahkan mendapatkan *feedback* sebagai landasan untuk memperbaiki perangkat lunak agar sesuai dengan spesifikasi kebutuhan.

2.2.3 Sistem Pendukung Keputusan

SPK merupakan sistem informasi khusus yang ditujukan untuk membantu manajemen dalam proses pengambilan keputusan yang berhubungan dengan permasalahan yang bersifat semi terstruktur secara efektif dan efisien, serta tidak menggantikan fungsi pengambil keputusan dalam membuat suatu keputusan. Sistem Pendukung Keputusan (SPK) memiliki tujuan yaitu untuk meningkatkan efektivitas dan efisiensi dalam pengambilan keputusan[9].

Carter et. al. menyatakan sistem pendukung keputusan memiliki tiga komponen utama atau subsistem utama dalam menentukan kapabilitas teknis SPK, antara lain subsistem data, subsistem model dan subsistem dialog, sebagai berikut:

a. Subsistem Data (*Data Subsystem*)

Komponen SPK yang bertugas menyediakan data yang dibutuhkan oleh sistem adalah subsistem data. Data tersebut disimpan dalam data base yang dikelola oleh suatu sistem yang disebut DBMS *(Database Management System)* gunanya untuk mempercepat proses ekstraksi saat data diperlukan.

b. Subsistem Model (*Model Subsystem*)

Komponen SPK yang bertugas menghasilkan suatu pemecahan atau hasil yang diinginkan dengan mengolah data yang di simpan di DBMS dengan model model yang dibuat adalah subsistem model[9].

c. SubSistem Dialog (*User System Interface*)

Pada subsistem ini adalah komponen yang bertugas untuk mengimplementasikan SPK yang dibuat sehingga *user* atau pemakai dapat berkomunikasi dengan sistem yang dirancang secara interaktif.

Proses Perancangan Sistem Pendukung Keputusan (SPK) memiliki delapan tahapan antara lain [9]:

1) Perencanaan (*Planning*)

Tahap ini berkaitan dengan perumusan masalah serta penentuan tujuan dari SPK yang akan dibuat.

2) Penelitian (*Research*)

Penelitian berkaitan dengan pencarian data dan sumber daya yang tersedia.

3) Analisis (*Analysis*)

Tahap ini menjadi penentuan teknik perancangan serta pendekatan pengembangan sistem yang akan dilakukan serta sumber data yang dibutuhkan.

4) Perancangan (*Design*)

Tahap ini melakukan perancangan terhadap ketiga subsistem dari SPK yaitu subsistem data, subsistem model dan subsistem dialog.

5) Pembangunan (*Construction*)

Tahap ini merupakan sambungan dari tahap perancangan, di mana ketiga subsistem yang dirancang digabungkan menjadi suatu SPK. Pada tahap ini dimulai penulisan bahasa pemrograman bagi SPK.

6) Implementasi (*Implementation*)

Tahap ini merupakan pengimplementasian SPK yang dibangun, selain itu juga ada beberapa tugas yang harus dilakukan seperti *testing, evaluation, demonstration, orientation, training, dan deployment.*

7) Pemeliharaan (*Maintenance*)

Tahap ini melibatkan perencanaan dukungan yang harus dilakukan terus menerus untuk mempertahankan keandalan sistem.

8) Adaptasi (*Adaptation*)

Dalam tahap ini dilakukan pengulangan terhadap tahap-tahap di atas sebagai tanggapan atas perubahan kebutuhan *user*.

2.2.4 Metode *Simple Additive Weighting*

Metode penjumlahan terbobot juga dikenal dengan sebutan Metode *Simple Additive Weighting* (SAW). Metode SAW membutuhkan proses normalisasi matriks keputusan (X) ke suatu skala yang dapat diperbandingkan dengan semua rating alternatif yang ada. Metode ini merupakan salah satu metode yang bisa digunakan untuk menghadapi situasi *Multiple Attribute Decision Making* (MADM). MADM itu sendiri merupakan suatu metode yang digunakan untuk mencari alternative optimal dari sejumlah alternatif dengan kriteria tertentu. Metode SAW ini mengharuskan pembuat keputusan menentukan bobot bagi setiap atribut. Skor total untuk alternatif diperoleh dengan menjumlahkan seluruh hasil perkalian antara rating (yang dapat dibandingkan lintas atribut) dan bobot tiap atribut. Rating tiap atribut haruslah bebas dimensi dalam arti telah melewati proses normalisasi matriks sebelumnya[10].

Berikut adalah langkah-langkah metode SAW[5]:

a. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu C_i.

b. Memberikan nilai bobot dari masing-masing kriteria yang telah ditentukan.

c. Menentukan nilai rating kecocokan dapat diperoleh menggunakan persamaan berikut.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

d. Membuat matriks keputusan berdasarkan kriteria (C_i), kemudian melakukan normalisasi matriks berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisasi R. Dapat diperoleh menggunakan persamaan berikut.
Formula untuk melakukan normalisasi tersebut adalah:
Rumus pada Aribut *Benefit*:

$$r_{ij} = \left\{ \frac{x_{ij}}{Max\, x_{ij}} \right.$$

Rumus pada Aribut *Cost*:

$$r_{ij} = \left\{ \frac{Min\, x_{ij}}{x_{ij}} \right.$$

Dimana:
r_{ij} = rating kinerja ternormalisasi
Max x_{ij} = nilai maksimum dari setiap baris dan kolom
Min x_{ij} = nilai minimum dari setiap baris dan kolom
x_{ij} = baris dan kolom dari matriks
Dengan rij adalah rating kinerja ternormalisasi dari alternatif A_i pada atribut C_j; i = 1, 2, m dan j = 1, 2, …, n.

e. Hasil dari nilai kinerja ternormalisasi (rij) membentuk matriks ternormalisasi (R) pada persamaan 3.

$$R = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1j} \\ \vdots & & & \vdots \\ r_{i1} & r_{i2} & \cdots & r_{ij} \end{bmatrix}$$

f. Hasil akhir diperoleh dari proses perangkingan yaitu penjumlahan dari perkalian matriks ternormalisasi (R) dengan vektor bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif

terbaik (A_i) sebagai solusi. Dapat diperoleh menggunakan persamaan 4.

Nilai preferensi untuk setiap alternatif (V_i) diberikan sebagai:

$$V_i = \sum_{j=1}^{n} w_j r_{ij}$$

Dimana:

V_i = Nilai akhir dari alternatif

w_j = Bobot yang telah ditentukan

r_{ij} = Normalisasi matriks

Nilai V_i yang lebih besar mengindikasikan bahwa alternative A_i lebih terpilih

2.2.5 Android

Android merupakan *software* untuk perangkat *mobile* dengan basis *linux*, yang mencakup sistem operasi *linux, middleware* dan aplikasi inti yang telah dirilis oleh *Google*[11]. Android adalah salah satu sistem operasi sekaligus platform pemrograman yang telah dikembangkan oleh *google* yang bisa digunakan di berbagai *smartphone* dan perangkat seluler lainnya. Tujuan dari android ialah meningkatkan inovasi di perangkat seluler sehingga pengguna dapat mencoba berbagai macam fitur yang tersedia[12].

Android telah menyediakan *platform* terbuka bagi para pengembang untuk membuat sebuah aplikasi. Selain itu juga disediakan *kit development* perangkat lunak untuk mempermudah pengembang dalam penulisan kode asli dan perakitan modul perangkat lunak. *Google* telah memberikan tawaran bagi pengembang agar lebih efisien dalam mengembangkan aplikasinya berupa lingkungan pengembangan terintegrasi (IDE) lengkap yang disebut android studio, dengan fitur lanjutan untuk pengembangan, *debug*, dan pemaketan aplikasi android[13].

2.2.6 Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan android dan bersifat *open source* atau gratis. Android studio sudah mendukung penggunaan beberapa bahasa pemrograman untuk mengembangkan aplikasi android serta mendukung IDE android studio antara lain yaitu Java, Kotlin, dan C++. *Toolkit* yang biasa digunakan dalam pengembangan aplikasi berbasis android biasanya adalah *Android Software Development Kit (SDK).* Ada berbagai alat yang disediakan di Android SDK seperti,

debugger, pustaka perangkat lunak, simulator, dokumentasi, kode sampel, dan tutorial[14]. *Google* telah mengumumkan peluncuran android studio pada 16 Mei 2013 pada saat *event Google I/O Conference* pada tahun 2013. Eclipse sebagai IDE resmi untuk mengembangkan aplikasi android sejak saat itu tergantikan oleh android studio[15].

2.2.7 Bahasa Pemrograman Kotlin

Kotlin adalah sebuah bahasa pemrograman yang telah dikembangkan oleh JetBrains, perusahaan pengembangan perangkat lunak, khususnya aplikasi perangkat lunak IDE. Kotlin dirilis sebagai bahasa pemrograman yang *open source* dan sekarang didukung *google* sebagai bahasa pemrograman dalam pengembangan aplikasi android. Selain untuk pengembangan aplikasi android Kotlin juga digunakan dalam pengembangan aplikasi berbasis *web*, *desktop* dan *backend*[16].

Kotlin merupakan bahasa pemrograman berbasis *Java Virtual Machine* (JVM). Kotlin disebut sebagai bahasa pemrograman yang pragmatis untuk android karena mengkombinasikan *object oriented* (OO) dan bahasa fungsional. Kotlin juga merupakan bahasa pemrograman yang *interoperabilitas* yang membuat bahasa ini dapat digabungkan dalam satu *project* dengan bahasa pemrograman Java[16]. Kotlin merupakan penyempurna dari bahasa pemrograman Java dalam pengembangan aplikasi android. Selain itu Kotlin juga disajikan sebagai bahasa pemrograman yang mudah dipahami karena sederhana dalam penulisan sintaknya. *Library* dan *framework* yang disediakan banyak dan memiliki fitur OOP (*object oriented programming*) digunakan di perusahaan besar[14].

2.2.8 UML (*Unified Modeling Language*)

UML merupakan salah satu alat bantu yang digunakan sebagai standarisasi bahasa pemodelan dalam pembangunan sistem yang dibangun dengan menggunakan teknik pemrograman berorientasi objek. Suatu sistem perangkat lunak membutuhkan sebuah pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan mendokumentasikan perangkat lunak tersebut. Hal tersebut dapat dilakukan menggunakan UML, karena merupakan bahasa visual sebagai pemodelan dan komunikasi terkait sebuah sistem dengan menggunakan diagram dan teks pendukung lainya[17].

2.2.9 *Use Case Diagram*

Use case diagram adalah pemodelan untuk perilaku sistem informasi yang akan dibuat dengan sistem yang diharapkan pengguna. *Use case diagram* menggambarkan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dengan kata lain *use case diagram* digunakan untuk mengetahui fungsi yang ada di dalam sebuah sistem informasi dan siapa yang berhak menggunakan fungsi tersebut[16].

Dua hal utama dalam *use case diagram* yaitu aktor dan *use case*. Adapun syarat penamaan pada *use case* yaitu nama didefinisikan sesimpel mungkin dan dapat dipahami. Aktor ialah orang, proses, maupun sistem yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, sedangkan *use case* ialah fungsionalitas yang disediakan sistem sebagai unit yang saling bertukar pesan baik antar unit maupun antar aktor. Simbol yang ada pada *use case diagram* dapat dilihat pada Tabel 2.1[17].

Tabel 2. 1 Simbol *Use Case Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		*Actor*	Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		*Use Case*	*Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Dinyatakan menggunakan kata kerja di awal *frase* nama *use case.*
3		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau memiliki interaksi dengan aktor
4		*Generalization*	Hubungan generalisasi dan spesialisasi antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

5		*Include*	Relasi tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.
6		*Extend*	Relasi tambahan ke sebuah *use case* dimana yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan.

2.2.10 *Sequence Diagram*

Sequence diagram merupakan salah satu jenis *diagram* pada UML yang digunakan untuk menggambarkan interaksi objek pada *use case* dengan menjelaskan berdasarkan urutan waktu hidup objek dan pesan yang dikirim dan diterima antar objek. *Sequence diagram* menggambarkan tahapan atau urutan yang harus dilakukan agar menghasilkan sesuatu seperti pada *use case diagram*[16].

Dalam membuat *sequence diagram* harus mengetahui objek yang terlibat dan metode yang dimiliki kelas yang diinstansiasi menjadi objek karena membutuhkan skenario yang ada pada *use case*. Banyaknya *sequence diagram* yang harus dibuat tergantung dari banyaknya *use case* yang didefinisikan[17]. Berikut beberapa simbol yang ada pada *sequence diagram* dapat dilihat pada Tabel 2.2[18]:

Tabel 2. 2 Simbol *Sequence Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		*Actor*	Aktor, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		*Boundary*	Menggambarkan sebuah *form*.
3		*Control Class*	Menghubungkan *boundary* dengan tabel.
4		*Entity Class*	Menggambarkan hubungan kegiatan yang akan dilakukan.
5		*Lifeline*	*Lifeline, object entity*, antarmuka yang saling berinteraksi. Menyatakan kehidupan suatu objek.

6		*Message*	Spesifikasi dari komunikasi antar *objek* yang memuat informasi-informasi tentang aktivitas yang terjadi.

2.2.11 *Flowchart*

Flowchart merupakan salah satu cara penyajian dari suatu algoritma berupa grafik dari bagan-bagan arus kerja suatu program atau sistem yang menggambarkan langkah-langkah dan urutan prosedur dalam menyelesaikan suatu masalah secara mendetail serta menggambarkan hubungan antara suatu proses dengan proses dalam suatu program. *Flowchart* menolong analis dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian[19]. Simbol-simbol *flowchart* yang dipakai sebagai alat bantu menggambarkan proses di dalam program dapat dilihat pada Tabel 2.3[20]:

Tabel 2. 3 Simbol *Flowchart*

No.	Simbol	Nama simbol	Keterangan
1		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalannya arus kerja dalam suatu proses. Simbol arus ini sering disebut juga dengan *connecting line*.
2		*Terminal*	Menyatakan permulaan atau akhir suatu program.
3		*Proses*	Proses perhitungan atau proses pengolahan data yang dilakukan oleh komputer.
4		*Manual Operation*	Menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
5		*Decision*	Keputusan dalan program yang menunjukan suatu kondisi tertentu yang akan menghasilkan data kemungkinan jawaban, ya atau tidak.
6		*Input - Output*	Menyatakan proses *input* dan *output* data yang diproses atau informasi tanpa tergantung dengan jenis peralatannya.

7		*Document*	*Input* atau *Output* dalam format yang di cetak.

2.2.12 Pengujian *Black Box*

Pengujian *black box* dikenal sebagai *behavioral testing* adalah metode pengujian *software* yang berfokus pada persyaratan fungsional perangkat lunak tanpa menguji kode program[16]. Dalam pengujian *black box* pengujian yang dilakukan bertujuan untuk mengetahui apakah fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan[21].

Pengujian menggunakan *Black Box* memiliki beberapa teknik antara lain, *Equivalence Partitions, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Performance Testing, Requirement Testing, Endurance Testing dan Cause-Effect Relationship Testing.* Salah satu teknik yang mudah digunakan adalah teknik *Boundary Value Analysis*. Cara kerja Teknik ini yaitu dengan menentukan batasan-batasan yang akan dimasukkan atau dilakukan pada aplikasi, selanjutnya hasil keluaran atau respon pada aplikasi akan diamati apakah sesuai dengan yang diharapkan. Estimasi banyaknya data uji dapat dihitung melalui banyaknya *field* data *entry* yang akan diuji[22].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut[22]:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

Tidak seperti pengujian *white box* yang dilakukan pada saat awal proses pengujian, pengujian *black box* cenderung diaplikasikan selama tahap akhir pengujian. Karena pengujian *black box* memperhatikan struktur *control* maka perhatikan berfokus pada domain informasi.

2.2.13 ERD (*Entity-Relationship Diagram*)

Entity-Relationship Diagram (ERD) merupakan salah satu teknik yang digunakan sebagai tahap dasar dalam pembuatan *database*. ERD merupakan salah satu teknik merancang *database* yang paling banyak digunakan. ERD merupakan representasi visual dari diagram yang

berbentuk notasi grafis untuk mendeskripsikan bagaimana entitas saling terkait antara satu dengan yang lainnya dalam *database*. Fungsi ERD adalah sebagai alat bantu dalam menganalisis pembuatan *database* dan memberikan gambaran bagaimana kerja *database* yang akan dibuat. Di dalam ERD terdapat 3 elemen dasar, yaitu entitas, atribut, dan relasi sebagai berikut[23]:

1. Entitas

Entitas merupakan objek yang akan menjadi perhatian dalam suatu *database*. Entitas dapat berupa manusia, tempat, benda, atau kondisi mengenai data yang dibutuhkan. Simbol dari entitas berbentuk persegi panjang.

2. Atribut

Atribut merupakan informasi yang terdapat dalam entitas. Sebuah entitas harus memiliki *primary key* sebagai ciri khas entitas dan atribut deskriptif. Atribut biasanya terletak dalam tabel entitas atau dapat juga terpisah dari tabel. Simbol dari atribut berbentuk elips.

3. Relasi

Relasi di dalam ERD merupakan hubungan antara dua atau lebih entitas. Simbol dari relasi berbentuk belahketupat[8]. Relasi yang dapat dimiliki oleh ERD ada beberapa macam, yaitu:

a. *One to One*

Satu anggota entitas dapat berelasi dengan satu anggota entitas lain

b. *One to Many*

Satu anggota entitas dapat berelasi dengan beberapa anggota entitas lain

c. *Many to Many*

Beberapa anggota entitas dapat berelasi dengan beberapa anggota entitas lain

Simbol-simbol yang digunakan pada ERD dapat dilihat pada Tabel 2.4[21].

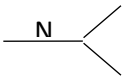
Tabel 2. 4 Simbol *Entity Relationship Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		Entitas	Merupakan data inti yang akan disimpan pada tabel basis data. Penamaan entitas biasanya lebih mengarah pada kata benda dan belum merupakan nama tabel.

2		Atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
3		Relasi	Relasi yang menghubungkan antar entitas biasanya diawali kata kerja.
4		*Link*	Penghubung antara himpunan relasi dengan himpunan entitas dan himpunan entitas dengan atributnya

Simbol ERD juga memiliki kardinalitas untuk menunjukkan maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*), dan banyak ke banyak (*many to many*). Simbol-simbol kardinalitas ERD dapat dilihat pada Tabel 2.5 berikut.

Tabel 2. 5 Simbol Kardinalitas ERD

No.	**Simbol**	**Nama**	**Keterangan**
1		*Association* / Asosiasi	Penghubung antar relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian (kardinalitas).
2		Relasi Satu ke Satu (*One to One*)	Relasi yang menunjukkan bahwa setiap himpunan entitas berhubungan dengan tepat satu himpunan entitas lainnya.
3		Relasi Satu ke Banyak (*One to Many*)	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak tetapi tidak sebaliknya.

4		Relasi Banyak ke Banyak (*Many to Many*)	Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya.

2.2.14 DBMS (*Database Management System*)

Database Management System (DBMS) atau sering disebut sebagai sistem manajemen basis data merupakan suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola dan menampilkan data. Suatu sistem aplikasi dapat dikatakan merupakan DBMS apabila telah memenuhi persyaratan seperti menyediakan fasilitas untuk mengelola akses data, menangani integritas data, menangani akses data dan mampu menangani *backup* data. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda[17].

Tujuan pengolahan data dalam *database* adalah agar dapat memperoleh atau menemukan kembali data yang ingin dicari dengan cepat mudah selain itu juga pengolahan data dan tujuan-tujuan yang lainya. Berikut tujuan *database*: kecepatan dan kemudahan, efisien ruang penyimpanan, keakuratan, ketersediaan, kelengkapan, keamanan, kebersamaan pemakai[24].

2.2.15 MYSQL

MySQL ialah sebuah *software* sistem manajemen basis data SQL atau yang dikenal dengan DBMS (*Database Management System*) yang bersifat *multithread* dan *multi user*[19]. MySQL sangat cepat dalam proses mengirim dan menerima data. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public Licence* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk lomba-lomba yang bersifat khusus[19]. MySQL termasuk jenis RDBMS (*Relation Database management Sistem*). Itulah sebabnya istilah seperti tabel, baris, dan kolom digunakan pada MySQL. Pada MySQL sebuah *database* mengandung salah satu atau sejumlah tabel[25].

Ada empat instruksi dasar yang digunakan dalam SQL (*structured query language*), yaitu[21]:

a. *Select*

Digunakan untuk menampilkan data yang telah ada atau tersimpan.

b. Insert

Digunakan untuk menambahkan data yang baru kedalam *database*.

c. Update

Digunakan untuk mengubah data yang telah disimpan sebelumnya pada *database*.

d. Delete.

Digunakan untuk menghapus data yang telah ada pada *database*.

MySQL mempunyai beberapa keunggulan, yaitu[16]:

a. MySQL dapat menangani *database* relasional dan dapat dipakai untuk *client/server*.
b. *Software* MySQL adalah *open source*, artinya dapat mengambil, memakai, dan mengubah *source*-nya dengan bebas tanpa biaya.
c. MySQL sangat cepat. *Multithreaded* yaitu setiap *query* diperlakukan sebagai *thread-based* yang sangat cepat.
d. MySQL dapat diakses oleh *client* menggunakan protokol TCP/IP pada semua *platform*. Pada *windows, client* dapat mengakses menggunakan *named-pipe*. Sementara itu UNIX (Linux) dapat memakai domain *socket-file*.
e. MySQL dapat menangani *database* dengan stabil dan tangguh, fleksibel dengan berbagai pemrograman.
f. MySQL dapat memiliki keamanan penyimpanan data yang baik, selain itu juga mendapat dukungan dari banyak komunitas,
g. Mudahan dalam management *database*, serta mendukung transaksi dan perkembangan *software* yang cukup cepat.

2.2.16 *Framework* CodeIgniter

Framework CodeIgniter adalah aplikasi *opensource* yang merupakan sebuah *web application framework* untuk mengembangkan aplikasi menggunakan bahasa pemrograman PHP secara dinamis yang dibangun menggunakan konsep *model view controller development pattern.* Tujuannya untuk memudahkan dalam penulisan kode dasar atau terstruktur menjadi lebih cepat karena memiliki banyak *library* yang bisa digunakan dalam pengerjaan aplikasi[26].

Framework CodeIgniter memiliki beberapa versi yaitu CodeIgniter 1 merupakan versi CodeIgniter yang paling pertama dirilis pada 28 Januari 2006, CodeIgniter 2 dirilis pada 28 Januari 2011, CodeIgniter 3 adalah versi yang paling banyak digunakan dirilis pada 30

Maret 2015, dan CodeIgniter 4 adalah versi yang sedang dalam pengembangan untuk menggantikan CodeIgniter 3.

Framework Codeigniter menyediakan layanan untuk membangun *REST Server* atau menggunakan *library* yang disematkan dalam program, tujuannya untuk membangun *REST Server* yang memiliki otentikasi. *JSON Web Token* (JWT) adalah salah satu cara yang digunakan untuk melakukan otentikasi *REST Server*. *Library Chriskacerguis* merupakan *library* yang khusus digunakan untuk membangun *REST Server* (*RESTful API*) pada *framework* Codeigniter 3. Otentiakasi yang diberikan adalah pemberian key untuk akses *REST Server*, dan pembatasan (*limit*) akses *REST Server*[27].

2.2.17 REST API *(Representational State Transfer Application Programming Interface)*

Representional State Transfer (REST) merupakan sebuah gaya arsitektur metode komunikasi pada sistem sesuai standar antar sistem komputer di *web* yang menggunakan protokol HTTP *(Hypertext Transfer Protocol)* untuk pertukaran data, sedangkan *Application Programming Interface* (API) adalah sebuah antarmuka yang mampu untuk mengintegrasikan data dan menghubungkan sebuah aplikasi yang berjalan di berbagai *platform* sehingga dapat saling terhubung satu sama lain[28].

Kelebihan API antara lain fitur keamanan dan hak akses yang dapat diatur agar dapat digunakan bagi pengembang secara publik. tambahan *platform* baru pada sebuah sistem tidak akan mempengaruhi kinerja *platfrom* yang sudah tersedia, karena API yang telah digunakan sudah mempunyai standar dan aturan yang berlaku dalam melayani akses permintaan *multiplatform*. Mempercepat proses *development* dengan menyediakan *function* secara terpisah sehingga *developer* tidak perlu membuat fitur yang serupa[26].

Dalam pembuatan *RESTful API* banyak sekali bahasa pemrograman dan *framework* yang bisa digunakan. Pemilihan teknologi dalam pengembangan *RESTful API* sangat penting karena dapat mempengaruhi *performa* pada *server* baik secara *response time*, *cpu usage* maupun *memory usage*. Maka dari itu dalam pengembangan *RESTful API* perlu memilih bahasa pemograman dan *framework* yang tepat sehingga *server RESTful API* dapat menangani *request* dari *client* dengan baik[27].

2.2.18 *Class Diagram*

Class diagram adalah salah satu jenis diagram pada UML yang digunakan untuk menampilkan kelas-kelas maupun paket-paket yang ada pada suatu sistem yang akan digunakan. *Class diagram* menggambarkan struktur sistem maupun relasi-relasi dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem[16].

Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. *Class diagram* dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Berikut adalah simbol–simbol yang ada pada *class diagram*[21].

Tabel 2. 6 Simbol *Class Diagram*

No	Simbol	Nama Simbol	Keterangan
1.		Kelas	Menjelaskan tentang kelas pada struktur sistem yang akan dibuat.
2.		Asosiasi/ association	Menjelaskan tentang relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan multiplicity pada sistem.
3.		Asosiasi berarah/ directed association	Menjelaskan tentang relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dengan multiplicity pada sistem.
4.		Generalisasi	Menjelaskan tentang relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus) pada sistem.

5.		Kebergantungan / dependecy	Menjelaskan tentang relasi antar kelas dengan makna kebergantungan antar kelas pada sistem.
6.		Agregasi/ agregation	Menjelaskan tentang relasi antar kelas dengan makna semua bagian (whole-part) pada sistem.