BAB II
TINJAUAN PUSTAKA

2.1 Prototype

Menurut Merriam Webster Dictionary *Prototype* pertama kali digunakan pada tahun 1552 di Prancis dan berasal dari bahasa Yunani *prototypon*. *Prototype* merupakan sebuah metode pengembangan software yang banyak digunakan pengembang agar dapat saling berinteraksi dengan user selama proses pembuatan sistem. *Prototype* mewakili model produk yang akan dibangun atau mensimulasikan struktur, fungsional, dan operasi sitem (Adi Fitra Andikos dan Yesi Gusteti, 2016).

Metode *prototype* merupakan sebuah paradigm baru dalam suatu metode pengembangan perangkat lunak. Metode ini tidak hanya sekedar evolusi dalam hal pengembangan perangkat lunak, tetapi memliki bentuk untuk merevolusi metode pengembangan perangkat lunak yang lama yaitu sistem sekuensial yang biasa dikenal dengan nama SDLC atau *waterfall development* model. Berikut ini adalah tahapan-tahapan dari metode *prototype*.

1. *Initial Requirement*

   Proses ini adalah tahapan awal dari metode prototype yang menjelaskan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. *Design*

   Design merupakan tahapan kedua dalam membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian terhadap pelanggan.

3. *Prototyping*

   Prototyping merupakan tahapan dari gambaran suatu sistem.

4. *Customer Evaluation*

   Evaluasi ini dilakukan oleh pelanggan, apakah *prototyping* yang sudah dibangun telah sesuai dengan yang diinginkan pelanggan atau tidak, apabila sudah sesuai maka langkah berikutnya akan diambil.

Namun jika tidak, maka *prototyping* akan direvisi dengan cara mengulang kembali langkah-langkahnya.

5. *Development*

   Dalam tahap ini *prototyping* yang sudah disepakati akan diterjemahkan ke dalam bahasa pemrograman yang sesuai.

6. *Test*

   Setelah sistem sudah menjadi suatu perangkat lunak yang siap untuk dipakai, kemudian dilakukan proses pengujian, pengujian ini dilakukan dengan *black box, white box, Basis path*, pengujian arsitektur, dll.

7. *Maintain*

   Perangkat lunak yang sudah diuji dan diterima pelanggan untuk siap digunakan.

2.2 Technique For Orders Reference by Similarity to Ideal Solution (TOPSIS)

TOPSIS atau kepanjangan dari Technique For Orders Reference by Similarity to Ideal Solution pertama kalinya diperkenalkan oleh Yoon dan Hwang pada tahun 1981. TOPSIS adalah salah satu metode untuk penyelesaian dalam hal permasalahan pengambilan keputusan yang terdiri dari banyak kriteria, metode TOPSIS juga dapat melakukan perengkingan terhadap alternatif terpilih dengan didasarkan pada konsep bahwasanya alternatif terpilih yang terbaik bukan hanya mempunyai jarak terpendek dari suatu solusi ideal positif tapi juga mempunyai jarak terpanjang dari suatu solusi ideal negatif. Solusi ideal positif bisa diartikan solusi yang mana memaksimalkan keuntungan *(profit)* dan meminimalkan suatu biaya *(cost)* (Kristiana, 2018). Sedangkan kalau solusi ideal negatif dapat diartikan dengan suatu solusi yang meminimalkan keuntungan dan memaksimalkan biaya. Alternatif-alternatif yang sudah dirangking selanjutnya dijadikan untuk referensi saat pengambilan keputusan sebagai bahan untuk memilih solusi terbaik yang diharapkan. Secara umum, prosedur TOPSIS mengikuti langkah-langkah sebagai berikut :

1. Membuat matriks keputusan yang ternormalisasi.
2. Membuat matriks keputusan yang ternormalisasi berbobot.
3. Menentukan matriks solusi ideal positif dan solusi ideal negatif.

4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan solusi ideal negatif.
5. Menentukan nilai prefensi untuk setiap alternatif.

TOPSIS memerlukan rating kerja setiap alternatif A_i di setiap kriteria C_j yang ternormalisasi yang dapat dilihat pada persamaan (1).

$$r_{ij=} \frac{X_{ij}}{\sqrt{\sum_{i=1}^{m} X_{ij^2}}}$$

dengan

i=1,2,….m; dan j=1,2,….n

dimana :

$r_{ij=}$ matriks keputusan ternormalisasi

$x_{ij=}$ bobot kriteria ke j pada alternatif ke i

Solusi ideal positif A^+ dan solusi ideal negatif A^- dapat ditentukan dengan berdasarkan rating bobot ternormalisasi (y_{ij}) dan dapat dilihat pada persamaan (2) :

$y_{ij}= w_i . r_{ij}$;

Dengan i= 1,2,…..,m dan j=1,2,….,n.

$A^+ = (y_1^+, y_2^+,…., y_n^+)$;

$A^- = (y_1^-, y_2^-,…., y_n^-)$;

Dimana y_{ij} merupakan matriks ternormalisasi terbobot [i][j] dan w_i yaitu vektor bobot [i]. Supaya dapat menghitung nilai solusi ideal maka langkah awalnya harus menentukan terlebih dahulu apakah bersifat keuntungan (*benefit*) dan atau bersifat biaya (*cost*). Dimana apabila j merupakan atribut keuntungan (*benefit*) maka y_j^+ adalah *max* y_{ij} dan untuk y_j adalah min y_{ij}. Sedangkan sebaliknya jika j merupakan atribut biaya (*cost*) maka y_j^+ adalah min y_{ij} dan y_i^- adalah max y_{ij}.

Rumus jarak antara alternatif A_i dengan solusi ideal *positif* yang dapat dilihat pada persamaan (3).

$$D_i^{+} = \sqrt{\sum_{i=1}^{m} (y_i^{+} - y_{ij})^2} \quad ; i = 1,2,\ldots,m$$

Dimana D_i^{+} merupakan jarak alternatif A_i dengan solusi ideal *positif*, y_i^{+} adalah solusi ideal *positif* [i] dan y_{ij} merupakan matriks normalisasi terbobot [i][j]. Sedangkan rumus jarak antara alternatif A_i dengan solusi ideal *negatif* yang dapat dilihat pada persamaan (4).

$$D_i^{-} = \sqrt{\sum_{i=1}^{m} (y_{ij}^{-} - y_i^{-})^2}\,; \; i = 1,2,\ldots,m$$

Dimana D_i^{-} merupakan jarak alternatif A_i dengan solusi ideal negatif, y_i^{-} merupakan solusi ideal positif [i] dan untuk y_{ij} merupakan matriks normalisasi terbobot [i][j].

Nilai prefensi pada tiap alternatif V_i dapat dilihat pada rumus di persamaan (5).

$$V_i = \frac{D_i^{-}}{D_i^{-} + D_i^{+}}; \; i = 1,2,\ldots,m$$

Dimana V_i adalah kedekatan tiap alternatif terhadap solusi ideal, untuk D_i^{+} merupakan jarak alternatif A_i dengan solusi ideal positif dan D_i^{-} yaitu jarak alternatif dengan solusi ideal negatif. Nilai V_i yang lebih besar memperlihatkan bahwa alternatif A_i lebih dipilih (Murnawan & Shiddiq, 2012).

2.3 Object Oriented Programming (OOP)

Pemrograman berorientasi objek (OOP) merupakan salah satu hal yang penting didalam ilmu komputer. Pemrograman beriorientasi objek saat ini adalah teknik pemrograman yang sangat popular dan banyak dijumpai digunakan oleh para programmer untuk menggantikan teknik pemrograman berbasis prosedur. Pemrograman berorientasi objek saat ini telah menerapkan

paradigma beriorientasi objek dalam Bahasa pemrograman. Paradigma berorientasi objek merupakan sebuah metode berpikir untuk menciptakan suatu model abstrak dari dunia fisik. Pemrograman berorientasi objek menerapkan entitas dalam dunia nyata seperti halnya objek, kelas, penyembunyian data (enkapsulasi), pewarisan, polimorfisme, abstraksi di dalam Bahasa pemrograman. Dalam OOP, program dirancang dengan membuat obyek yang saling berinteraksi satu sama lain.

Pemrograman berorientasi objek atau *Object Oriented Programming* (OOP) adalah cara baru dalam berpikir serta berlogika dalam menghadapi berbagai masalah yang akan dicoba dibatasi dengan bantuan Komputer. Cara berpikir berorientasi objek merupakan segala sesuatu yang dilihat sebagai suatu objek. Paradigma rekayasa perangkat lunak berorientasi objek berbasis pada tatanan berorientasi objek. Pemrograman berorientasi objek secara umum mempunyai empat keunggulan, yaitu sebagai berikut :

1. Modularitas yang memberikan cara termudah untuk memecahkan masalah.
2. Penggunaan kembali kode.
3. Fleksibilitas.
4. Yang paling alami dan pragmatis pendekatan yang membuatnya berguna pemecahan masalah.

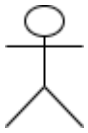
2.4 *Use Case Diagram*

Diagram *use case* memperlihatkan perilaku sistem, subsistem, atau kelas yang tampil pada sisi pengguna (Rumbaugh et al, 1999). *Use case* bermanfaat untuk mengetahui fungsi atau kegunaan apa saja yang ada di dalam sistem informasi dan siapa saja yang mempunyai hak untuk menggunakan fungsi-fungsi tersebut, dan membantu megkomunikasikan sebuah rancangan aplikasi dengan konsumen. Berikut adalah objek yang ada dalam *use case diagram*.

a. Aktor

Aktor merupakan pencitraan penggunaan eksternal, proses atau segala sesuatu yang saling berhubungan dengan sistem, subsistem atau kelas. Aktor biasanya memberikan informasi kepada sistem dan

memerintahkan sistem untuk melakukan sesuatu. Gambar dari aktor dapat dilihat pada gambar 2.1.



Gambar 2. 1 Aktor

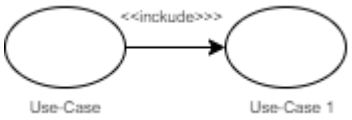
b. Use-Case

Use-Case merupakan suatu fungsi yang sudah disediakan oleh unit sistem dan diutarakan oleh satu atau beberapa aktor unit sistem. Tujuan dari *use-case* sendiri yaitu menentukan perilaku sistem tanpa mengungkap struktur internal sistem. Contoh Use-Case dapat dilihat pada gambar 2.2.

Gambar 2. 2 *Use Case*

c. Include Relationship

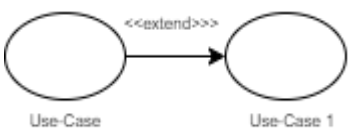
Include Relationship adalah relasi cakupan yang memungkinkan suatu *use-case* menggunakan fungsionalitas yang telah disediakan *use-case* lain. Contoh *Include Relationship* dapat dilihat pada gambar 2.3.



Gambar 2. 3 *Include Relationship*

d. Extend Relationship

Extend Relationship merupakan relasi yang memungkinkan *use-case* dapat memperluas fungsionalitas yang telah disediakan *use-case* lain. Contoh *Extend Relationship* dapat dilihat pada gambar 2.4.



Gambar 2. 4 *Extend Relationship*

2.5 Sequence Diagram

Sequence diagram yaitu UML yang mana menggambarkan interaksi di dalam dan sekitar sistem (termasuk pengguna, display, dan sebagainya) berbentuk pesan yang diletakkan diantara objek-objek didalam use case dan digambarkan dalam waktu tertentu. Sequence diagram terdiri dari dua dimensi, yaitu dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terpaut).

Sequence diagram biasanya digunakan sebagai suatu untuk menggambarkan skenario atau rangkaian langkah-langkah yang dijalankan sebagai jawaban dari sebuah *event* untuk meghasilkan *output* tertentu. Tujuan utama dari sequence diagram yaitu untuk mengetahui urutan kejadian yang menghasilkan output yang diinginkan.

Ada beberapa komponen utama dari sequence diagram yang sering digunakan, yaitu sebagai berikut :

a. Aktor

   Komponen yang pertama yaitu aktor. Aktor menggambarkan seorang pengguna atau *user* yang keberadaannya di luar sistem dan sedang berinteraksi dengan sistem. Biasanya digambarkan berupa symbol stick figure.

b. Objek

   Objek digunakan untuk mendokumentasikan perilaku sebuah objek didalam sistem. Komponen ini berbentuk kotak dan berisikan nama dari objek dengan garis bawah.

c. Activation Box

   Activation box merupakan komponen yang menyampaikan waktu yang dibutuhkan suatu objek untuk menyelesaikan tugasnya. Semakin lama waktu yang dibutuhkan , maka secara otomatis activation boxnya akan menjadi lebih panjang.

d. Lifeline

   Lifeline adalah sebuah garis yang menggambarkan masa hidup dari sebuah objek dalam suatu sequence diagram.

e. Message (Pesan)

Message menunjukan komunikasi antar objek. Message muncul secara berurutan pada lifeline.

2.6 Activity Diagram

Activity diagram menggambarkan bagaimana alir aktivitas yang terjadi pada sistem yang akan dirancang, bagaimana masing-masing alir itu berawal**,** descision yang mana mungkin akan terjadi, dan bagaimana mereka berakhir. Activity diagram juga bisa menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

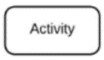
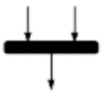
Activity diagram berfungsi membantu orang-orang di sisi bisnis dan pengembangan organisasi untuk memahami proses dan perilaku yang sama, Jadi sangat berguna bagi beberapa pemangku kepentingan. Dalam activity diagram akan digunakan beberapa simbol khusus termasuk yang digunakan untuk memulai, mengakhiri, menggabungkan, atau menerima langkah dalam alur untuk membuat activity diagram.

Dalam penggunaanya activity diagram mempunyai manfaat, Berikut beberapa manfaat activity diagram yaitu :

a. Mendemonstrasikan logika suatu algoritma.
b. Menjelaskan langkah-langkah yang dilakukan dalam kasus penggunaan UML.
c. Menggambarkan proses bisnis atau alur kerja antara pengguna dan sistem.
d. Sederhanakan dan tingkatkan proses apapun dengan mengklarifikasi kasus penggunaan yang rumit.
e. Memodelkan elemen arsitektur perangkat lunak, seperti metode, fungsi, dan operasi.

Selain beberapa manfaat yang dituliskan diatas terdapat bentuk dan simbol activity diagram, ada beberapa jenis paling umum yang ditemukan dalam diagram UML. Dibawah ini pada tabel 2.1 adalah gambar jenis simbol activity diagram yang paling umum.

Tabel 2. 1 Gambar jenis simbol *activity diagram*

No.	Gambar	Nama	Keterangan
1		Start Symbol	Awal dari proses atau alur kerja dalam activity diagram. Ini bisa digunakan dengan sendirinya atau dengan simbol catatan yang menjelaskan titik awal.
2		Activity	Merupakan aktivitas yang menunjukkan pembentukan proses yang dimodelkan. Simbol ini mencakup bentuk deskripsi singkat.
3		Connector	Menunjukkan aliran terarah, atau aliran control dari aktivitas. Panah masuk menggambarkan memulai langkah aktivitas, setelah langkah selesai aliran akan berlanjut dengan panah keluar.
4		Joint symbol/ Synchronization bar	Menggabungkan dua aktivitas bersamaan dan memperkenalkannya kembali ke aliran di mana hanya satu aktivitas yang terjadi pada satu aktivitas.
5		Decision	Digunakan mewakili keputusan dan memiliki setidaknya dua jalur yang bercabang dengan teks kondisi untuk memungkinkan pengguna melihat opsi.
6		Note	Untuk mengkomunikasikan pesan tambahan yang tidak sesuai dengan diagram itu sendiri.
7		Send signal	Menunjukkan bahwa sinyal sedang dikirim ke aktivitas penerima
8		Receive signal	Menerima suatu peristiwa, setelah event diterima, alur yang bersumber dari action ini maka selesai.
9		Shallow history pseudostate	Merupakan transisi yang memanggil status aktif terakhir.

No.	Gambar	Nama	Keterangan
10		Option Loop	Untuk memodelkan urutan berulang dalam simbol loop opsi.
11		Condition text	Ditempatkan disisi sebelah penanda keputusan untuk memberitahu dalam kondisi apa aliran aktivitas harus dipisahkan kearah itu.
12		End Symbol	Penanda status akhir dari rangkaian akhir proses.

2.7 Class Diagram

Class diagram merupakan struktur sistem dari sebuah pendefinisian yang mana apabila diintansiasi akan menghasikan suatu objek dan itu merupakan inti dari pengembangan dan desain berorientasi objek. *Class diagram* menjelaskan model dalam perancangan atribut dan fungsi-fungsi yang akan digunakan untuk membangun sebuah sistem baru. Di dalam *class diagram* juga menggambarkan struktur dan deskripsi *class*, *package*, dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain sebagainya. Class memiliki tiga area pokok, yaitu sebagai berikut :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Class dapat disebut implementasi dari interface, yaitu class abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinistansiasikan, tetapi harus diimplementasi menjadi sebuah website, maka hal ini *interface* mendukung resolusi metoda pada saat *run-time*. Selanjutnya di bawah ini akan dijelaskan bagaimana hubungan antar kelas, yaitu sebagai berikut :

a. Asosiasi, merupakan hubungan statis antar *class*. Biasanya menggambarkan *class* yang memiliki atribut berupa class lain, atau *class* yang harus mengetahui eksitensi *class* lain.
b. Agregasi, merupakan hubungan yang menyatakan bagian.
c. Pewarisan, yaitu hubungan *hirarkis* antar *class*.

d. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

2.8 Website

Website merupakan kumpulan-kumpulan dari beberapa halaman situs, yang terangkum dalam sebuah domain atau subdomain, dimana tempatnya ada di dalam *World Wide Web (WWW)* yang ada dalam internet. Website juga merupakan halaman yang berisi data, baik data text, gambar, suara atau lainnya yang dapat diakses secara online menggunakan protocol HTTP *(hyper text transfer protocol)* dan dalam pengaksesannya menggunakan perangkat lunak yang disebut browser. Ada banyak jenis browser yang popular saat ini diantaranya, yaitu *Internet exploler* yang diproduksi oleh *Microsoft, Mozila Firefox, Opera* dan *Safari* yang diproduksi oleh Apple.

Browser (Perambah) merupakan aplikasi yang mampu menjalankan dokumen-dokumen web dengan cara diterjemahkan. Prosesnya dilakukan oleh komponen yang ada dalam aplikasi browser yang biasa disebut web engine. Semua dokumen web ditampilkan dengan cara diterjemahkan (M. Rudyanto Arief, 2011, Hal : 7).

2.9 PHP (Hypertext Processor)

PHP merupakan singkatan dari PHP *Hypertext Processor* yang digunakan sebagai Bahasa script server-side dalam pengembangan web yang disisipkan pada dokumen HTML. Menurut Andi (2007 : 5), PHP (hypertext Preprocessor) merupakan salah satu bahasa pemrograman yang berjalan dalam sebuah *web server* dan memiliki fungsi sebagai pengolah data pada sebuah server. Data yang dikirim oleh *user client* akan diolah dan disimpan pada database web server dan dapat ditampilkan kembali apabila diakses. Kode-kode program PHP dijalankan dengan mengupload file kedalam server.

Dalam penggunaan PHP memungkinkan web dapat dibuat dinamis sehingga *maintenance* situs web tersebut menjadi lebih mudah dan efisien. PHP ditulis dengan menggunakan Bahasa C. PHP adalah bahasa pemrograman script server-side yang didesain untuk pengembangan web. PHP merupakan

software *Open-Source* yang disebarkan dan dilisensikan secara gratis dan dapat didownload secara bebas melalui situs resminya yaitu https://www.php.net/.

Sintaks Program/Script dalam penulisannya yaitu diapit tanda khusus PHP. Terdapat empat macam pasangan tag PHP yang digunakan untuk menandai blok script PHP.

1. <?php……..?>
2. <script language=”PHP”>……</script>
3. <?.....?>
4. <%.....%>

(Kasiman Peranginangin, 2006, Hal:2)

2.10 *CodeIgniter*

CodeIgniter atau CI merupakan salah satu *framework* dari PHP. CodeIgniter dikembangkan oleh Rick Ellis. Tujuan pembuatan *framework* CodeIgniter menurut user manualnya yaitu untuk menghasilkan *framework* yang akan dapat digunakan untuk pengembangan proyek pembuatan situs web yang lebih cepat dibandingkan dengan pembuatan situs web dengan cara koding secara manual, dengan menyediakan banyak sekali pustaka yang dibutuhkan dalam pembuatan situs web dengan antarmuka yang sederhana dan struktur logika untuk mengakses pustaka yang dibutuhkan.

CodeIgniter merupakan *framework* yang bersifat *open source* dan menggunakan metode MVC (*Model, View, Controller*) sehingga memudahkan developer atau programmer dalam membangun sebuah aplikasi berbasis web tanpa harus membuatnya dari awal, sehingga aplikasi web tersebut lebih terstruktur dan lebih sederhana. Secara sederhana konsep MVC terdiri dari tiga bagian yaitu *model, view*, dan *controller*. Berikut ini adalah penjelasan mengenai tiga bagian tersebut.

1. Model

   Menjelaskan struktur data dari situs web yang bisa berupa basis data maupun data lain. Misalnya dalam bentuk *file* teks atau *file* xml. Didalam view biasanya terdapat class dan fungsi untuk mengambil, melakukan *update* dan menghapus data situs web. Bagian model

biasanya berhubungan dengan perintah-perintah *query* SQL karena situs web membutuhkan tempat untuk menyimpan data yaitu dengan basis data atau *database*.

2. *View*

   View adalah suatu informasi yang ditampilkan kepada pengunjung situs web. Bisa dikatakan *view* adalah halaman situs web yang dibuat menggunakan HTML dengan bantuan CSS atau JavaScript. Di dalam *view* jangan pernah ada kode untuk melakukan koneksi ke basis data, dikarenakan *view* hanya khusus untuk menampilkan data-data hasil dari model dan *controller*.

3. *Controller*

   Controller adalah penghubung antara model dan view. Di dalam *controller* inilah terdapat *class* dan fungsi-fungsi yang memproses permintaan dari *view* ke dalam struktur data didalam model. Tugas *controller* yaitu menyediakan berbagai variabel yang akan ditampilkan di *view*, memanggil model untuk melakukan akses ke basis data, menyajikan penanganan waktu *error*, mengerjakan proses logika dari aplikasi serta melakukan validasi atau cek terhadap *input*. Jadi untuk urutan sebuah *request* yaitu, *user* berhubungan dengan *view*, didalam view semua informasi akan ditampilkan. Ketika *user* melakukan *request*, misalnya klik tombol maka permintaan tersebut selanjutnya akan diproses oleh controller. Kemudian controller akan meminta Model untuk menyelesaikan permintaan tersebut, baik itu melakukan *query* atau lainnya. Setelah dari Model, data akan dikirimkan kembali untuk diproses lebih lanjut di dalam controller dan baru dari controller data akan ditampilkan ke dalam view.

2.11 Skala Likert

Skala *likert* adalah metode pengukuran yang digunakan untuk mengukur pendapat seseorang menggunakan kuesioner agar bisa mengetahui skala sikap terhadap suatu objek tertentu (Sugiyono, 2010). Skala likert merupakan suatu skala psikometrik yang umum digunakan dalam kuisioner, dan merupakan

skala yang paling banyak digunakan dalam riset berupa survey dan penelitian karena skala tersebut adalah yang paling mudah digunakan.

Skala *likert* adalah skala pengukuran yang telah dikembangan oleh Likert. Skala *likert* mempunyai empat atau lebih butir-butir pertanyaan yang dikombinasikan sehingga akan membentuk sebuah skor atau nilai yang memperesentasikan sifat individu, contohnya dapat meliputi pengetahuan, sikap dan perilaku. Dalam proses suatu analisa data, komposit skor, biasanya jumlah atau rataan, dari semua butir pertanyaan dapat digunakan (Indriani, Sakethi, & Syarif, 2020). Berikut ini adalah Langkah-langkah dalam perhitungan skala *likert*.

1. Penentuan skor

   Langkah pertama yaitu menentukan skor pada setiap jawaban dari beberapa pernyataan yang akan dijawab oleh responden.Seperti pada tabel 2.2 merupakan skala jawaban pada skala likert beserta skor yang berada dibawah ini.

Tabel 2. 2 Skala Jawaban Pada Skala *Likert*

Skala Jawaban	Skor	Range Skor
Sangat tidak setuju (STS)	1	1% - 20%
Kurang setuju (KS)	2	21% - 40%
Netral (N)	3	41% - 60%
Setuju (S)	4	61% - 80%
Sangat setuju (SS)	5	81% - 100%

   Jika dirasa pernyataan termasuk sulit untuk diberikan jawaban, maka bisa ditambahkan pilihan jawaban hingga7-9 skala pada setiap pernyataan.

2. Skor Tertinggi

   Rumus untuk menghitung skor tertinggi adalah sebagai berikut :

   Y = Skor Tertinggi x Jumlah Responden

3. Jumlah Skor

Setelah menghitung dan diperoleh skor tertinggi, maka langkah berikutnya yaitu menghitung jumlah skor. Jumlah skor merupakan jumlah dari perkalian antara jumlah responden pada tiap-tiap jawaban dikalikan dengan nilai skor pada tiap-tiap jawaban. Dibawah ini adalah rumus dalam menghitung jumlah skor.

Jumlah Skor = T x Pn

T = Total jumlah responden yang memilih

Pn = Pilihan angka skor likert

4. Rumus Indeks

Apabila sudah diperoleh nilai skor tertinggi dan jumlah skor, maka selanjutnya yaitu menghitung rumus index dengan rumus seperti dibawah ini :

$$\textbf{Rumus Index (100\%)} = \frac{\textbf{Jumlah Skor}}{\textbf{Skor Jawaban Tertinggi}} \textbf{ x 100}$$

5. Interval

Pencarian interval digunakan untuk mengetahui interpretasi penilaian hasil akhir dari perhitungan rumus indeks. Rumus interval adalah sebagai berikut :

I = 100 / Jumlah Skor

2.12 Blackbox Testing

Blackbox Testing adalah suatu teknik pengujian perangkat lunak yang berfokus pada spesifikasi fungsional dari perangkat lunak. *Blackbox Testing* bekerja dengan cara mengabaikan struktur kontrol sehingga perhatiannya difokuskan terhadap informasi domain. *Black Box Testing* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program.

Keuntungan dalam penggunaan *Blackbox Testing*, yaitu penguji tidak perlu memiliki keahlian tentang pengetahuan bahasa pemrograman tertentu, dalam pengujian dilakukan dari sudut pandang pengguna, hal ini membantu

dalam mengungkapkan ambiguitas atau inkonsistensi dalam spesifikasi persyaratan, dan *programmer* dan *tester* saling bergantung diantara keduanya.

Blackbox Testing juga mempunyai kekurangan, yaitu Uji kasus sulit didesain tanpa adanya spesifikasi yang jelas, Kemungkinan memiliki pengulangan tes yang sudah dilakukan oleh *programmer,* dan bagian back end tidak diuji sama sekali.

2.13 ISO 9126

ISO 9126 adalah salah satu *framework* standart internasional yang berfungsi untuk menguji kualitas perangkat lunak, dibuat oleh *Internasional Organization for standardization* (ISO) dan *Internasional Electrotechnical Commision* (IEC). Untuk standart internasional ini mempunyai kemampuan dalam mendefinisikan suatu kualitas produk perangkat lunak, mutu, model, dan metrik terkait untuk mengevaluasi dan menetapkan kualitas dari produk perangkat lunak. Model ISO 9126 memiliki enam karakteristik antara lain:.

1. *Fungsionality* (Fungsionalitas)
2. *Reliability* (Kehandalan)
3. *Usability* (Kebergunaan)
4. *Efficiency* (Efisiensi)
5. *Maintainabilitiy* (Pemeliharaan)
6. *Portability* (Portabilitas)

Dalam penelitian ini peneliti menggunakan ISO 9126-3 faktor usability. Usability merupakan sebuah analisa kualitatif yang mana menentukan seberapa mudah user untuk menggunakan antar muka suatu aplikasi, aplikasi dapat dikatakan *usable* apabila fungsinya dapat dijalankan secara efektif, efisien, dan memuaskan. Suatu efektivitas sangat berhubungan dengan keberhasilan dalam mencapai sebuah proses perangkat lunak, sedangkan efisiensi berhubungan dengan kelancaran *user* atau pengguna dalam menggunakan perangkat *mobile* tersebut dan kepuasan berkaitan dengan penerimaan sikap dari *user* atau pengguna terhadap perangkat lunak.

Usability atau kegunaan perangkat lunak digunakan dalam uji kelayakan aplikasi dalam penelitian ini menggunakan lima pertanyaan sehubungan dengan factor usability / kegunaan, yaitu kemudahan aplikasi dipahami

(understandbility), dipelajari *(learnbility)*, dioperasikan *(operability)*, daya ketertarikan *(attractiveness)*, dan kesesuaian dengan tujuan *(usability compliance)*.