BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini akan membahas mengenai teori dasar yang diperlukan untuk penelitian dan pengembangan perangkat lunak. Dasar teori yang dijadikan landasan dalam penelitian ini meliputi pengertian *Crowdsourcing*, Location *Based Service* (LBS), Google Maps API, *Web Service*, *Javascript Object Notation* (JSON), *Firebase Cloud Messaging* (FCM), Formula *Harversine*, konsep dasar *Unified Modelling Languague* yang dipakai pada rekayasa perangkat lunak dan konsep pengujian yang dipakai pada perangkat lunak.

2.1 Kajian Pustaka

Kajian pustaka pada penelitian ini berisi tentang penelitian sebelumnya mengenai sistem pelaporan kejadian dan *crowd sourcing* pada *smartphone*.

Penelitian terdahulu yang berjudul *"Rancang Bangun Sistem Pelaporan Kepolisian Berbasis Android Menggunakan SMS Gateway dan Websocket"* yang dilakukan oleh Irfan Septiadi Putra pada tahun 2015 berlokasi di Universitas Brawijaya Malang. Penelitian tersebut membahas mengenai pelaporan kejadian kepada pihak kepolisian yang menggunakan *smartphone* Android sebagai media pelaporannya.

Dalam melakukan pelaporan pengguna sebelumnya harus terdaftar sebagai pelapor di dalam system tersebut, dan saat akan melakukan pelaporan maka aplikasi pada *smartphone* Android akan mendapatkan lokasi pelapor dan menampilkan jenis kategori laporan yang akan disampaikan. Laporan tersebut dikirimkan ke *server* dan di hitung lokasi polsek terdekat menggunakan Google Maps API dan diteruskan menuju polsek tersebut (Putra, 2015).

Selain dari konsep penilitian yang di lakukan Irfan Septiadi Putra di Indonesia di kota Malang sendiri telah di terapkan konsep yang kurang lebih sama yakni pelaporan kejadian kepada polsek melalui *smartphone* Android yang memanfaatkan lokasi dari pelapor yang, namun terdapat perbedaan yakni pengiriman data laporan tidak menggunakan SMS melainkan *Webservice* atau koneksi internet (FILKOM UB, 2015). Berbeda dengan kota Malang yang konsep dari tombol panik terfokus pada polsek saja, di Bandung sendiri yang lebih dahulu menerapkan konsep serupa dalam proses penanganannya langsung di tangani oleh pusat teknologi Bandung atau yang lebih dikenal Bandung Command Center (Nurmatari, 2015).

Penelitian terdahulu "*Crowdsourcing with Smartphone*" yang ditulis oleh Georgios Chatmilioudis dkk membahas mengenai bagaimana memanfaatkan *smartphone* untuk melakukan *crowdourcing* yang memungkinkan pengguna berkontribusi untuk menyelesaikan suatu permasalahan yang kompleks. Salah satu aplikasi pada *smartphone* android yang dibahas bernama *Crowdcast* menggunakan lokasi dari pengguna untuk dapat menghubungkan dengan pengguna disekitarnya, sehingga saat pengguna mengajukan pertanyaan maupun

meminta bantuan mengenai sesuatu akan dicarikan pengguna disekitarnya untuk menjawab atau meresponnya (Chatmilioudis, et al., 2012).

Implementasi aplikasi *mobile* untuk mencari dan memberikan bantuan terhadap permasalahan kendaraan berdasarkan lokasi terdekat yang memiliki konsep dasar serupa dengan konsep tombol panik dan aplikasi *Crowdcast* diharapkan dapat membantu masyarakat mendapatkan bantuan lebih cepat dengan melibatkan orang di sekitarnya.

2.2 Landasan Teori

Landasan teori yang digunakan dalam penelitian ini berasal dari buku yang mendukung dan jurnal – jurnal penelitian sebelumnya. Landasan – landasan teori yang diambil yaitu berkaitan dengan Crowdsourcing, Location *Based Service* (LBS), Google Maps API, *Web Service*, *Javascript Object Notation* (JSON), *Firebase Cloud Messaging* (FCM), konsep dasar *Unified Modelling Languague* atau UML dan konsep pengujian perangkat lunak.

2.2.1 Crowdsourcing

Crowdsourcing merupakan sebuah konsep yang dapat diartikan secara singkat sebagai jenis aktivitas kolaboratif berbasis internet. Dimana lebih lanjut dapat diartikan sebagai aktivitas *online* dimana sekelompok individu dengan berbagai pengetahuan dengan sukarela melakukan tugas yang diberikan melalui sebuah panggilan terbuka. Orang yang melakukan tugas tersebut akan menerima kepuasan baik secara ekomomi, pengakuan sosial, harga diri, atau pengembangan keterampilan. Sedangkan orang yang membuat tugas akan mendapatkan keuntungan dari hasil tugas yang dituntaskan oleh pelaksana tugas (Estellés-Arolas & González-Ladrón-de-Guevara, 2012).

2.2.2 *Location Based Services* (LBS)

Location Based Services yang selanjutnya disebut dengan LBS atau layanan berbasis lokasi adalah suatu istilah umum yang digunakan untuk menentukan lokasi dari sebuah perangkat bergerak. LBS merupakan suatu layanan yang bereaksi aktif terhadap perubahan entitas posisi sehingga mampu mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut (Anwar, et al., 2014). LBS pada penelitian ini digunakan untuk mendapatkan lokasi pengguna saat ini menggunakan GPS saat hendak mencari bantuan dan lokasi terkini dari pengguna lainnya. Terdapat tiga unsur utama LBS yaitu (Anwar, et al., 2014):

1. *Location Manager* (*API Maps*), memberikan source untuk LBS. Aplication Programming Interface (API) Maps menyediakan fasilitas untuk menampilkan atau memanipulasi peta.
2. *Location Providers* (*API Location*), memberikan teknologi pencarian terhadap lokasi yang digunakan oleh perangkat.

3. *API Location* berhubungan dengan data dari GPS (*Global Positioning System*) dan data lokasi secara real-time. Lokasi, perpindahan, serta kedekatan dengan lokasi tertentu dapat ditentukan melalui *Location Manager*.

2.2.3 Google Maps API

Google Maps adalah fitur yang pada dasarnya disediakan gratis oleh *google*, dan bukan standar fitur dari perangkat android. Fitur *Google Maps* ini dapat digunakan secara *embeded* dalam beberapa bahasa pemrograman termasuk android melalui sebuah jembatan aplikasi yang disebut *API (Application Programming Interface)*. *Google* menyediakan *API* untuk pengembang mengintegrasikan aplikasinya dengan fitur *Google Maps*. Dengan menggunakan *Google Maps API*, peta *google* akan dapat diintegrasikan dengan berbagai platform pemrograman, sehingga peta yang disediakan oleh google dapat tampil di halaman aplikasi yang dibuat (Adam & Hayuhardika, 2012).

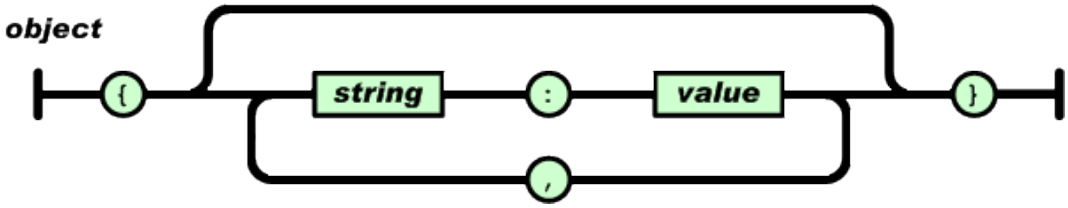
2.2.4 *Web Service*

Web service adalah suatu aplikasi basis data (*database*), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara remote oleh berbagai perangkat dengan sebuah perantara tertentu. Secara umum, *web service* dapat diidentifikasi dengan menggunakan URL seperti *website* pada umumnya. Namun yang membedakan *web service* dengan *website* pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL *website*, URL *web service* hanya mengandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berfungsi untuk membangun sebuah fungsi-fungsi tertentu aplikasi. *Web service* dapat diartikan juga sebuah metode pertukaran data, tanpa memperhatikan dimana sebuah basis data disimpan, dibuat dalam bahasa apa sebuah aplikasi yang mengkonsumsi data, dan di platform apa sebuah data tersebut dikonsumsi. *Web service* mampu menunjang interopabilitas, sehingga mampu menjadi sebuah jembatan penghubung antara berbagai sistem yang ada (Nolan, et al., 2013).

Website pada umumnya digunakan untuk melakukan request dan response yang dilakukan antara *client* dengan *server*. Sebagai contoh, seorang pengguna layanan web tertentu mengetikkan alamat *url website* untuk membentuk sebuah *request*. Maka *request* akan sampai pada *server*, kemudian diolah dan disajikan dalam bentuk sebuah *response*. Dengan singkat kata terjadilah hubungan client-*server* secara sederhana. Sedangkan pada *web service* hubungan antara client dan *server* tidak terjadi secara langsung. Hubungan antara client dan *server* dijembatani oleh data *web service* dalam format tertentu. Sehingga akses terhadap basis data tidak akan ditangani secara langsung oleh client, melainkan melalui perantara yang disebut *web service* (Nolan, et al., 2013). Pada penelitian ini menggunakan *web service* sebagai media pertukaran informasi data antara *server* dan *client*.

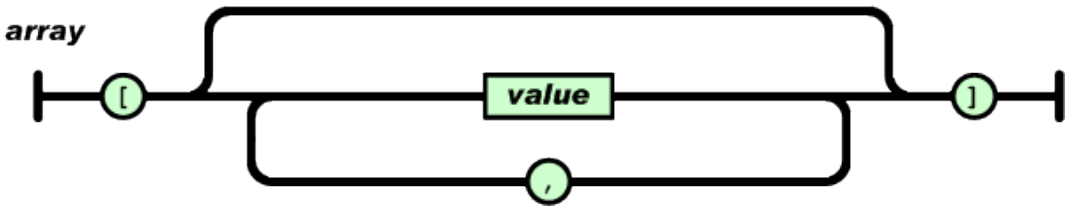
2.2.5 Javascript Object Nation (JSON)

JSON merupakan sebuah format data yang digunakan untuk melakukan pertukaran data. Pertukaran data bisa dilakukan dalam satu komputer ataupun berbeda komputer dan platform. Hal ini karena JSON mudah untuk ditulis dan dibaca oleh manusia serta mudah diterjemahkan (decode) ataupun dibuat (encode) oleh komputer (JSON ORG, 2016). JSON terbuat dari dua buah struktur yang pertama nama atau nilai dan yang kedua nilai terurutkan. Untuk struktur yang pertama biasa disebut dengan JSON Object dan sedangankan untuk yang kedua biasa disebut dengan JSON Array. Untuk lebih jelas mengenai bentuk notasi pada JSON Object dapat dilihat pada Gambar 2.1 dan untuk bentuk notasi pada JSON Array dapat dilihat pada Gambar 2.2.



Gambar 2.1 Notasi JSON Object

Sumber: JSON ORG (2016)



Gambar 2.2 Notasi JSON Array

Sumber: JSON ORG (2016)

2.2.6 Firebase Cloud Messagng (FCM)

Firebase Cloud Messaging atau yang selanjutnya disebut dengan FCM adalah sebuah layanan gratis yang disediakan oleh Google untuk memungkinkan para pengembang aplikasi dapat mengirim pesan antara aplikasi yang terserdia pada *server* dan aplikasi yang tesedia pada klien. Pengiriman pesan ini termasuk pesan *downstream* dari *server* ke aplikasi klien, dan pesan *upstream* dari aplikasi client ke *server* (Google Developers, 2017).

Misalnya, pesan *downstream* bisa menginformasikan ke aplikasi yang tersedia pada klien bahwa terdapat data baru yang dapat diambil dari *server*, seperti pemberitahuan "terdapat email baru". Untuk kasus penggunaan seperti pesan singkat, pesan FCM dapat mentransfer hingga 4KB *payload* (isi data) ke aplikasi klien. Layanan FCM menangani semua aspek antrian pesan dan pengiriman ke dan dari aplikasi klien.

Gambar 2.3 Arsitektur FCM

Sumber: (Google Developers, 2017)

Pada Gambar 2.3 dijelaskan arsitektur atau bagaimana interaksi antara komponen dengan FCM dimana Firebase Cloud Messaging *Server* menerima *downstream* dari aplikasi *server* maupun Notification Console GUI dan meneruskannya kepada aplikasi klien yang dapat berupa perangkat Android iOS dan Web.

2.2.7 Formula Haversine

Formula *haversine* merupakan sebuah persamaan yang digunakan untuk mencari jarak dalam sebuah lingkaran dari dua titik koordinat berdasarkan *latitude* dan *longitude*. Dimana formula yang digunakan untuk menghitung jarak dua buah koordinat pada permukaan bumi berdasarkan *latitude* dan *longitude* dirumuskan dalam Persamaan 2.1 (Chopde & Nichat, 2013):

$$d = 2r \sin^{-1}(\sqrt{\sin^2\left(\frac{\phi_2-\phi_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\sin^2(\frac{\psi_2-\psi_1}{2})}) \qquad (2.1)$$

Dimana d adalah jarak antara dua titik koordinat dengan ϕ sebagai posisi latitude dan ψ sebagai posisi longitude dengan r sebagai *radius* dari bumi. Dalam penerapan kedalam bahasa pemrograman didapatkan *SQL Statement* yang ditunjukan pada Kode 2.1.

```
1  3956 * 2 *ASIN (SQRT(POWER(SIN((orig.lat - dest.lat)*pi()/180/2),
   2) +COS(orig.lat*pi()/180)
   *COS(dest.lat*pi()/180)*POWER(SIN((orig.lon - dest.lon) *
   pi()/180/2), 2)) ) AS DISTANCE
```

Kode 2.1 *SQL Statement* fomula haversine

2.2.8 UML

Bahasa pemodelan yang digunakan dalam proses perancangan aplikasi pencarian dan pemberian bantuan pada permasalahan kendaraan bermotor adalah UML. UML merupakan suatu bahasa pemodelan yang bertujuan untuk melakukan penjelasan spesifikasi, visualisasi, konstruksi dan dokumentasi artifak dari pencarian dan pemberian bantuan pada permasalahan kendaraan bermotor. UML digunakan untuk mempermudah dalam memahami, merancang, menelusuri, mengatur, memelihara, dan mengontrol informasi mengenai sistem yang akan dibangun. UML menangkap informasi mengenai struktur statis dan perilaku

dinamis dari suatu sistem. Struktur statis menentukan jenis objek yang penting untuk sistem dan implementasinya, serta hubungan antar objek tersebut. Perilaku dinamis mendefinisikan sejarah suatu objek dari waktu ke waktu dan komunikasi antar objek untuk mencapai tujuan (Rumbaugh & Booch, 1999). UML digambarkan dengan menggunakan berbagai macam diagram yang masing-masing memiliki fungsi dan makna tersendiri. Pada rancangan aplikasi ini digunakan 5 diagram yakni, *use case diagram, activity diagram, class diagram, sequence diagram* dan *entity relationship diagram*.

2.2.8.1 *Use Case* Diagram

Use case diagram digunakan untuk menggambarkan fungsionalitas yang dibutuhkan dari aplikasi pencarian bantuan ini. Diagram ini yang ditekankan adalah "apa" yang dapat diperbuat sistem beserta aktornya, bukan "bagaimana" sistem berjalan. Sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* membantu dalam menyusun *requirement* pencarian bantuan ini dan merancang kasus pengujian untuk semua fitur yang ada pada sistem dalam sebuah skenario *use case* (Dharwiyanti, 2003).

2.2.8.2 *Activity* Diagram

Activity Diagram digunakan untuk menggambarkan alur aktivitas dalam aplikasi pencarian bantuan yang dirancang, bagaimana aktivitas diawali, kondisi yang mungkin akan terjadi, dan bagaimana aktivitas tersebut akan diakhiri. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Diagram *activity* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi dilakukan *trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan bagaimana perilaku internal sebuah sistem (interaksi antar subsistem), tetapi lebih menggambarkan berbagai proses dan jalur aktivitas dari level atas secara umum (Dharwiyanti, 2003).

2.2.8.3 *Class* Diagram

Aplikasi pencarian bantuan merupakan sistem yang berorientasi objek baik pada sisi *client* maupun *server*, oleh sebab itu pada proses perancangan diperlukan *Class diagram* untuk memodelkan tampilan desain statis dari suatu sistem. *Class diagram* menunjukkan satu set dari *class*, *interface*, dan kolaborasi beserta hubungan antar *class*. *Class* adalah sebuah cetakan dari sekumpulan objek yang berbagi atribut, operasi, hubungan, dan semantik yang sama (Booch & Jacobson, 1998). *Class* dapat berbentuk suatu implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki kumpulan metode saja. *Interface* tidak dapat langsung diinstansiasikan, tetapi terlebih dahulu harus diimplementasikan menjadi sebuah *class* (Dharwiyanti, 2003).

2.2.8.4 *Sequence* Diagram

Sequence diagram digunakan untuk menggambarkan interaksi antar objek di dalam dan di sekitar aplikasi pencarian bantuan (termasuk pengguna, *display*, dan basisdata) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan [DHA - 03]. Diagram *sequence* menampilkan interaksi berupa grafik dua dimensi. Dimana dimensi vertikal menunjukkan sumbu waktu dan dimensi horizontal menunjukkan peran *classifier* yang mewakili objek individu dalam berkolaborasi. Setiap peran *classifier* diwakili oleh kolom vertikal. Selama waktu untuk suatu objek masih ada, maka suatu peran ditunjukkan dengan garis putus-putus. Selama waktu aktivasi prosedur pada objek yang aktif, maka digambarkan sebagai garis ganda (Rumbaugh & Booch, 1999).

2.2.9 Pengujian Perangkat Lunak

Diperlukannya pengujian sistem yang telah dirancang dan diimplementasikan agar mengetahui kesalahan dan segala jenis kemungkinan yang dapat menimbulkan kesalahan sesuai dengan spesifikasi aplikasi yang telah ditentukan. Berdasarkan standar IEEE, pengujian perangkat lunak meliputi pengertian aktivitas yang dilakukan dan mengevaluasi kualitas produk dan untuk mengembangkannya dengan melakkan indentifikasi kelemahan dan permasalahan yang terjadi (Simamarta, 2010).

Strategi untuk pengujian perangkat lunak mengintegrasikan metode desain kasus uji perangkat lunak kedalam serangkaian langkah yang disusun dengan baik dan hasilnya adalah konstruksi perangkat bergerak yang berhasil. Strategi pengujian dapat dilakukan melalui pengujian tingkat rendah yaitu pengujian yang dilakukan pada kode program untuk membuktikan bahwa segmen kode sumber yang kecil telah diimplementasikan dengan tepat. Selain itu terdapat pengujian tingkat tinggi yang memvalidasi fungsi-fungsi sistem mayor yang tidak sesuai dengan kebutuhan pengguna (Pressman, 2001).

Pada aplikasi pencarian dan pemberian bantuan pada permasalahan kendaraan bermotor ini dilakukan pengujian dengan menggunakan pengujian validasi untuk memvalidasi fungsi – fungsi aplikasi mayor yang tidak sesuai dengan kebutuhan pengguna, serta dilakukan pengujian *usability* untuk mengukur kepuasan akan kemudahan penggunaan dan kebermanfaatan yang diberikan oleh sistem. Berikut merupakan penjelasan dari pengujian validasi dan pengujian *usability.*

2.2.9.1 Pengujian Validasi

Pengujian validasi aplikasi pencarian dan pemberian bantuan pada permasalahan kendaraan bermotor dicapai melalui sederetan pengujian *blackbox* yang menampilkan kesesuaian sistem dengan persyaratan kebutuhan yang telah

didefinisikan pada *use case diagram* dan skenario *use case.* Pengujian menguraikan kelas – kelas pengujian yang akan digunakan untuk mengungkap kesalahan ketika terjadi penyesuaian sistem dengan persyaratan. Pengujian dan skenario uji keduanya didesain untuk memastikan apakah semua persyaratan fungsional terpenuhi pada sistem, semua persyaratan kinerja tercapai, dan dokumentasi dibuat dengan benar (Pressman, 2001).

2.2.9.2 Pengujian *Usability*

Pada penelitian ini, pengujian *usability* dilakukan untuk mengkaji dan menilai seberapa mudah penggunaan aplikasi yang dibuat. Artinya kata "*usability*" sendiri mengacu pada suatu metode untuk melakukan improvisasi terhadap *ease-of use* selama proses perancangan. Pengujian *usability* mencakup 5 komponen yaitu (Nielsen, 2012):

1. *Leranability*

   Semudah apa pengguna dapat mempelajari penggunaan produk tersebut pada pertama kali penggunaan.

2. *Efficiency*

   Secepat apa pengguna dapat melakukan tugasnya.

3. *Memorability*

   Sejauh mana pengguna dapat mengingat langkah atau proses yang perlu dilakukan untuk menyelesaikan tujuannya.

4. *Errors*

   Sebanyak apa, sejauh mana dan semudah apa bagi pengguna jika berhubungan dengan kesalahan yang ada.

5. *Satisfaction*

   Bagaimana tanggapan pengguna terhadap rancangan produk secara keseluruhan.

- **Kuisioner USE**

Dalam penelitian ini, acuan kuesioner yang dipakai adalah kuesioner USE. Kuesioner USE merupakan media kuesioner untuk mengukur *usability* dengan memakai 3 parameter yaitu kegunaan (*usefullness*), kepuasan (*satisfaction*) dan kemudahan penggunaan (*ease of use*). *Ease of use* merupakan sebuah parameter yang dibagi menjadi 2 faktor yaitu kemudahan dalam penggunaan (*ease of use*) dan kemudahan dalam mempelajari aplikasi (*ease of learning*) (Aelani & Falahah, 2012).

Contoh beberapa pertanyaan dalam kuesioner USE adalah sebagai berikut:

1. *Usefulness*
   - Aplikasi ini dalam pengerjaanya memenuhi ekspektasi saya.

- Aplikasi ini membuat saya menjadi lebih produktif.
- Aplikasi ini sangat berguna.

2. *Ease of Use*
   - Aplikasi ini mudah digunakan.
   - Aplikasi ini *user-friendly*.
   - Aplikasi ini fleksibel.
3. *Ease of Learning*
   - Aplikasi ini dapat dengan mudah dan cepat saya pelajari.
   - Aplikasi ini mudak diingat dalam penggunaanya.
4. *Satisfaction*
   - Aplikasi ini menyenangkan untuk digunakan.
   - Saya merasa saya harus memiliki aplikasi ini.

- **Skala Likert**

Skala Likert dalam penelitian ini digunakan sebagai acuan penilaian dan analisis hasil dalam melakukan survei untuk keperluan pengujian *usability*. Metode Likert merupakan suatu metode penentuan skala dalam pernyataan sikap yang menggunakan distribusi respons sebagai dasar penentuan nilai skalanya. Nilai dari skala Likert tergantung dari suatu kebutuhan. Skala Likert menjabarkan variabel yang diukur menjadi indikator variabel dimana indikator tersebut kemudian dijadikan sebagai titik tolak untuk menyusun butir-butir instrumen yang dapat berupa pernyataan atau pernyataan (Risnita, 2014).

Jawaban setiap pertanyaan atau pernyataan yang menggunakan skala Likert mempunyai gradiasi dari sangat positif sampai sangat negatif yang dapat berupa kata-kata sebagai contoh:

a. Sangat setuju
b. Setuju
c. Netral
d. Tidak setuju
e. Sangat tidak setuju

Untuk keperluan analisis kuantitatif, maka jawaban tersebut dapat diberi skor yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 Penilaian jawaban kuantitatif

Jawaban	Skor
Sangat setuju	5
Setuju	4
Netral	3
Tidak setuju	2
Sangat tidak setuju	1

Perhitungan skala Likert diterapkan untuk mendapatkan indeks persentase pengujian *usability*. Persamaan 2.2 digunakan untuk menghitung Total Skor dengan *nilaiSTS* merupakan jumlah dari jawaban sangat tidak setuju, *nilaiTS* merupakan jumlah jawaban tidak setuju, *nilaiN* merupakan jumlah dari jawaban netral, *nilaiST* merupakan jumlah jawaban setuju dan *nilaiSS* merupakan jumlah jawaban sangat setuju.

Persamaan 2.3 digunakan untuk menghitung nilai Y yang didapatkan dari perkalian *SkorLikerTertinggi* dengan *JumlahResponden*. Dimana dalam penilaian jawaban kuantitatif pada Tabel 2.1 skor tertinggi adalah pada jawaban sangat setuju yang bernilai 5. Persamaan 2.4 digunakan untuk menghitung indeks persentase yang didapatkan setelah melakukan perhitungan *TotalSkor* dari persamaan 2.2 dan Y dari Persamaan 2.3.

$$TotalSkor=(nilaiSTS\times 1)+(nilaiTS\times 2)+(nilaiN\times 3)+(nilaiST\times 4)+(nilaiSS\times 5) \quad (2.2)$$

$$Y = SkorLikerTertinggi \times JumlahResponden \quad (2.3)$$

$$Index(\%) = (TotalSkor/Y) \times 100\% \quad (2.4)$$