BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini akan digunakan lima tinjauan pustaka yang nantinya dapat mendukung penelitian, berikut ini merupakan tinjauan pustaka yang diambil yaitu pada tabel 2.1:

Tabel 2.1 Tinjauan Pustaka

1	Penulis	Latif dan Suwarjono (2018)
	Judul	Sistem Informasi Geografis Pendataan Bangunan Berdasarkan Izin Mendirikan Bangunan Di Kabupaten Merauke
	Masalah	Proses pendataan yang dilakukan masih secara manual dengan tertulis dan direkap pada aplikasi office
	Metode	*Waterfall*
	Hasil	Dengan diterapkan sistem informasi geografis untuk pendataan bangunan mempermudah mengetahui lokasi bangunan yang akan dibangun serta kerusakan bangunan yang harus diperbaikai.
2	Penulis	Wulandari, Thamrin, Dan Budiawan (2016)
	Judul	Aplikasi Informasi Lokasi Jalan Rusak BerbasisWeb dan Android
	Masalah	Saat ini mengadukan hal seperti itu cukup sulit, karena kurangnya media yang khusus menyediakan pengaduan jalan rusak. Oleh karena itu, perlu adanya suatu media yang berfungsi untuk menyampaikan pengaduan jalan rusak yang kemudian akan menjadi sebuah informasi bagi pengguna jalan
	Metode	*Waterfall*
	Hasil	Aplikasi Informasi Lokasi Jalan Rusak Berbasis Web dan Android (BrokenRoads.app) adalah aplikasi media pelaporan dan media informasi. Media pengaduan terletak di aplikasi android. Tidak hanya pengaduan, user juga dapat melakukan pelaporan jalan rusak melalui aplikasi android ini. Sedangkan aplikasi web berperan sebagai media informasi dan pengelolaan data untuk admin. Lokasi jalan rusak sendiri ditampilkan dalam bentuk map, sehingga dapat memudahkan pengguna jalan mengetahui titik pasti jalan rusak, kelebihan yang dimiliki yaitu proses pelaporan dari masyarakat menjadi lebih mudah.

Tabel 2.1 Tinjauan Pustaka (Lanjutan)

3	Penulis	Ibrahim dan maita (2017)
	Judul	Sistem Informasi Pelayanan Publik Berbasis Web Pada Dinas Pekerjaan Umum Kabupaten Kampar
	Masalah	Adapun kinerja sistem dalam pelayanan pengaduan yang sedang berjalan pada Dinas Pekerjaan Umum Kabupaten Kampar masih belum optimal karena pengelolaannya masih manual.
	Metode	*Object Oriented Analysis Design* (OOAD)
	Hasil	Sistem Pelayanan Publik Berbasis Web ini mempermudah masyarakat dalam menyampaikan pengaduan dan permohonan pembangunan atau perbaikan jalan dan jembatan di Kabupaten Kampar, serta mempercepat pihak Dinas PU untuk merespon setiap pengaduan dan mempermudah dalam mengelola pengaduan yang disampaikan oleh masyarakat
4	Penulis	Maulidiansyah, Rakhman dan Ramdhani (2017)
	Judul	Aplikasi Pelaporan Kerusakan Jalan Tol Menggunakan Layanan Web Service Berbasis android
	Masalah	Jika jalan dalam keadaan tidak baik untuk dilalui oleh pengguna jalan seperti adanya jalan berlubang maka harus diadakan perbaikan jalan untuk memastikan keamanan dan kenyamanan pengguna jalan. Namun karena proses pelaporannya masih bersifat manual dengan menggunakan form isian, ini menyebabkan lama waktu pelaporan setidaknya memakan waktu satu hari kerja karena setelah inspektor melakukan inspeksi jalan, inspektor harus menyalin laporan hasil inspeksinya kedalam komputer terlebih dahulu.
	Metode	REST (*Representational State Transfer*)
	Hasil	Penelitian ini menghasilkan sebuah aplikasi pelaporan kerusakan jalan tol dalam perangkat mobile berbasis android dengan menggunakan layanan web service. Pelaporan kerusakan jalan tol ini digunakan untuk melaporkan beberapa hal diantaranya: jenis kerusakan jalan, lokasi kerusakan jalan, ukuran kerusakan jalan, dsb. Proses pelaporan kerusakan jalan dilakukan oleh inspektor. Tujuan dari penelitian ini adalah untuk mengimplementasikan layanan web service dengan metode REST (Representational State Transfer) pada perangkat mobile berbasis android.

Tabel 2.1 Tinjauan Pustaka (Lanjutan)

5	Penulis	Wirnanda, Anggraini dan Isya (2018)
	Judul	Analisis Tingkat Kerusakan Jalan Dan Pengarunya Terhadap Kecepatan Kendaraan (Studi Kasus: Jalan Blang Bintang Lama Dan Jalan Teungku Hasan Dibakoi)
	Masalah	Perencanaan yang tidak tepat, pengawasan yang kurang baik dan pelaksanaan yang tidak sesuai dengan rencana yang ada, selain itu minimnya biaya pemeliharaan, keterlambatan pengeluaran anggaran serta prioritas penanganan yang kurang tepat juga menjadi penyebabnya.
	Metode	Analisis Regresi
	Hasil	Kerusakan jalan yang terjadi di beberapa ruas jalan menimbulkan kerugian yang sangat besar terutama bagi pengguna jalan seperti waktu tempuh yang lama, kemacetan, kecelakaan, dan lain-lain. Secara umum penyebab kerusakan jalan ada berbagai sebab yaitu umur rencana jalan yang telah dilewati, genangan air pada permukaan jalan yang tidak dapat mengalir akibat drainase yang kurang baik, beban lalu lintas berulang yang berlebihan (overloaded) yang menyebabkan umur pakai jalan lebih pendek dari perencanaan, kesimpulanya semakin tinggi kecepatan kendaraan maka kerusakan jalan akak semakin cepat.

Berdasarkan uraian tinjauan pustaka diatas, maka pembahasan yang dilakukan belum pernah dilakukan oleh peneliti-peneliti sebelumnya terutama pada dinas bina marga Bandar Lampung, serta dapat dilihat perbedaannya yaitu :

1. Sistem yang akan dibuat berbasis *mobile*
2. Menggunakan sistem pemetaan Map
3. Masyarakat yang melaporkan kerusakan jalan menggunakan data KTP

2.2 Jenis Kerusakan Perkerasan Jalan

Jjenis kerusakan jalan yaitu Setiap struktur perkerasan jalan akan mengalami proses pengerusakan secara progresif sejak jalan pertama kali dibuka untuk lalu lintas. Kerusakan jalan dapat dibedakan menjadi dua, yaitu kerusakan struktural mencakup kegagalan perkerasan atau kerusakan dari satu atau

lebih komponen perkerasan yang mengakibatkan perkerasan tidak dapat lagi menanggung beban lalu lintas sehingga harus diperbaiki dengan membangun ulang perkerasan tersebut dan kerusakan fungsional adalah suatu kondisi kerusakan dimana keamanan dan kenyamanan pengguna jalan menjadi terganggu sehingga biaya operasi kendaraan semakin meningkat (Yudaningrum and Ikhwanuddin, 2017).

Kerusakan jalan dapat dilihat pada Tabel 2.2 di bawah ini :

Tabel 2.2 Jenis Kerusakan pada Perkerasan Jalan

No.	Jenis Kerusakan
1	Retak kulit buaya
2	Retak kotak-kotak
3	Retak pinggir
4	Retak sambungan
5	Retak memanjang/ melintang
6	Patah/slip
7	Keriting
8	Amblas
9	Alur
10	Sungkur
11	Mengembang/jembul
12	Lubang
13	Cekungan
14	Pelepasan butir
15	Pengausan agregat
16	Kegemukan
17	Pinggir turun vertikal
18	Tambalan
19	Perlintasan jalan rel

2.3 *Mobile*

Aplikasi *mobile* yaitu program siap pakai yang direkap untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain dan dapat digunakan oleh

sasaran yang dituju sedangkan *mobile* dapat di artikan sebagai perpindahan dari suatu tempat ke tempat yang lain (Gunawan *et al.*, 2017).

Maka aplikasi *mobile* dapat di artikan sebuah program aplikasi yang dapat dijalankan atau digunakan walaupun pengguna berpindah-pindah dari satu tempat ke tempat yang lain serta mempunyai ukuran yang kecil.

2.3.1 *Mobile Web*

Mobile web adalah Halaman HTML berbasis *browser* yang diakses menggunakan perangkat *portable* (*smartphone atau tablet*) melalui jaringan seluler seperti 3G, 4G maupun Wifi. *Mobile Web* dirancang untuk menampilkan data seperti teks, gambar, dan video dari sebuah *website* ke dalam sebuah tampilan yang lebih kecil yakni perangkat *mobile* (Gunawan and Saputro, 2018).

Mobile Web bertujuan membuat *web* dapat diakses dari sebuah perangkat *mobile* secara sederhana seperti mengakses *web* dari sebuah komputer *dekstop*. Dalam membuat sebuah *mobile web* membutuhkan implementasi untuk perbaikan dari segi *interoperability*, *usability* dan *accessibility* pada sebuah *mobile web*. *Mobile Web* umumnya berukuran ringan disetiap halamannya yang ditulis dengan *Extensible Hypertext Markup Language* (XHTML) atau *Wireless Markup Language* (WML) untuk mengirimkan konten ke perangkat *mobile*. Selain itu beberapa teknik seperti dengan menggunakan *Adobe Flash Lite* atau *Sun* J2ME yang memungkinkan untuk membuat perangkat *mobile* yang lebih bervariasi.

2.3.2 *Web Based*

Web based adalah Aplikasi yang dibuat berbasis *web* yang membutuhkan *web server* dan *browser* untuk menjalankannya (Urbieta *et al.*, 2019).

Dengan membuat sistem berbasis *web based* ada beberapa hal yang penting dan harus kita pikirkan sebelum membangun sistem tersebut, di antaranya:

1. Tidak membutuhkan *hardware* dengan spesifikasi yang tangguh untuk menjalankan aplikasinya.
2. Server yang dibutuhkan cukup di instalkan *tools* pendukung saja agar klien mudah menjalankan aplikasi
3. Infrastruktur jaringan yang dibutuhkan juga cukup besar karena aplikasi yang dibuat dapat diakses dari jaringan luar (internet).
4. Aplikasi berbasis *web based* dapat diakses dari berbagai perangkat dengan syarat menggunakan *web browser* saja sudah dapat mengaksesnya.
5. Jika aplikasi yang sudah jadi ingin di *update*, sangat mudah untuk melakukannya karena tidak membutuhkan membuka keseluruhan aplikasi.

2.3.3 *Web View*

Web view adalah Sebuah *class* pada *Android* yang berfungsi sebagai semacam *sendbox* untuk menampilkan dan menjalankan aplikasi *mobile* yang berbasiskan *web*, seperti HTML5, JQuery *Mobile*, dan sebagainya (Srijaya *et al.*, 2015).

Cara mudahnya *WebView* itu seperti *Android browser*, tetapi tidak mempunyai address bar tempat memasukkan alamat URL. Karena URL yang mau kita buka sudah didefinisikan di aplikasi dan tidak bisa kita ganti.

2.3.4 *CodeIgniter*

CodeIgniter adalah *Framework* untuk bahasa pemprograman PHP, yang dibuat Rick Ellis pada tahun 2006. CodeIgniter memiliki banyak fitur yang membantu para pengembang PHP untuk dapat membuat aplikasi secara mudah dan cepat serta memiliki sifat yang fleksibel dapat mengembangkan dalam perangkat web, dekstop maupun mobile (Raharjo, 2018).

Sehingga *framework* tersebut sangat mudah diterapkan dan dikembangkan dengan kelebihan memiliki banyak refrensi yang dapat digunakan sebagai acuan pengembangan sistem.

2.3.5 MySql

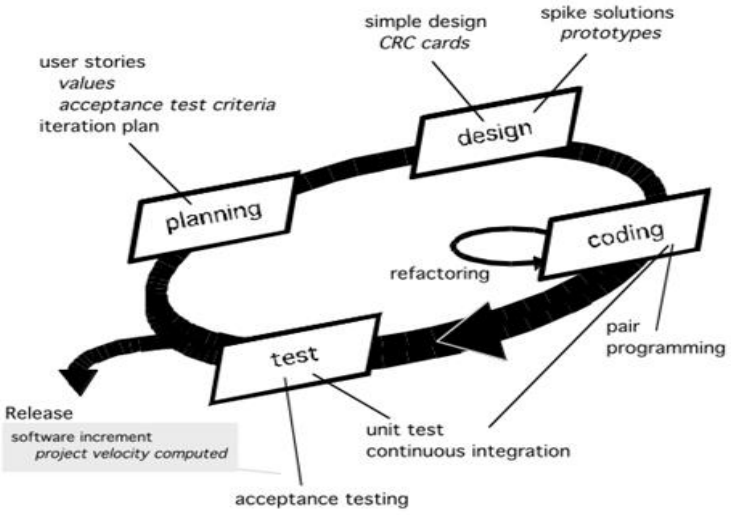
MySQL adalah basis data yang bersifat *open source* sehingga banyak di gunakan untuk media. Walaupun gratis, MySQL tetap berkualitas dan sudah cukup memberikan performance yang memadai. Penggunaan PHP MyAdmin lebih mudah digunakan karena menggunakan *interface* yang lebih mudah dipahami (Sabar *et al.*, 2019).

MySQL adalah nama sebuah *database server* yang menangani akses database yang selalu dalam bentuk pernyataan SQL (*Structured Query Language*) yaitu suatu bahasa yang digunakan untuk mengakses *database* relasional.

2.4 *Extreme Programming*

Extreme programming adalah pendekatan *extreme programming* merupakan suatu pendekatan berorientasi objek dan sebagai pengembangan perangkat lunak cepat sedikit lebih rinci dengan tujuan memberikan ulasan secara ringkas (Suryantara, 2017).

Paradigma yang diinginkan mencakup didalam seperangkat aturan dan praktik-praktik dalam empat konteks kegiatan kerangka yang dapat dilihat pada Gambar 2.1:



Gambar 2.1 *Extreme Programming*
Sumber: (Suryantara, 2017)

2.4.1 Tahapan Pengembangan *Extreme Programming*

Tahapan dalam penelitian sebagai langkah-langkah penelitian yang harus dikerjakan, berkut adalah tahapan penelitian *extreme progrmamming* menurut (Suryantara, 2017).

1. *Planning* (Perencanaan)

Peneliti atau pengembang memutuskan bagaimana hasil *story* dari pengguna dibangun dengan komitmen telah disepakati, adapun *story-story* yang dilakukan dengan cara :

a. Pengguna menceritakan apa permasalahan pada sistem yang digunakan dan sistem seperti apa yang akan dibangun. Menurut (Schwaber and Sutherland, 2017), *User Story* adalah menceritakan dari perspektif pengguna mengenai apa yang dia inginkan agar lebih seperti yang dilakukan oleh sistem.

Berdasarkan uraian tersebut, maka tampilan berubah dari produk ke pengguna sepenuhnya dan *User Stories* menjadi standar persyaratan di semua kerangka kerja sistem.

b. Berdasarkan hasil cerita pengguna maka peneliti menentukan poin pada bagian *value* untuk memutuskan apa saja yang akan dibangun.

c. Dari hasil kesepakatan tersebut maka peneliti menentukan *acceptence criteria test* yaitu menentukan kriteria-kriteria apa saja yang nantinya sebagai acuan terhadap sistem yang akan di uji.

d. Sehingga hasil peneliti menyimpulkan berapa kali akan dilakukan *realies* dan perbaikan pada tahap *interation plan* merencanakan berapa kali akan dilakukan uji terhadap sistem yang dibangun.

2. *Design* (Pemodelan)

Extreme programming pada proses pembuatan desain di lakukan untuk memberikan informasi gambaran sistem yang akan dibangun, berikut adalah beberapa desain yang akan dilakukan oleh peneliti:

a. CRC *Card* untuk mengenali dan mengatur *object oriented class* yang sesuai dengan pengembangan. Jika pada saat perancangan terdapat ketidak sesuaian maupun perbaikan maka akan dilakukan

b. *Spike Solution* yang dilakukan kepada pengguna untuk mendapatkan kesesuaian antara ke inginan pengguna dengan pengembangan yang dilakukan.

c. *Prototype* adalah bagian perancangan berupa *user interface* dalam bentuk *wireframing* untuk mempermudah pengguna melihat desain sistem.

3. *Coding* (Pengkodean)

Pada proses pengkodean peneliti menyesuiakan terhadap *story* pengguna sehihngga sistem yang dibangun sesuai, proses pengkodean yang diakukan yaitu:

a. *Pair Programming* merupakan tahap sistem dibangun dengan bahasa pemrograman dan media penyimpanan yang telah disepakati.
b. *Refactory* merupakan tahapan yang dilakukan ketika terjadi ketidak sesuaian kode program sehingga dilakukan perbaikan guna mendapatkan hasil yang sesuai.

4. *Testing* (Pengujian)

Tahap pengujian dilakukan oleh pengguna sebagai *user* dengan melakukan uji sesuai dengan *acceptance test* yang telah ditentukan dan disetujui. *Unit test* yang telah dibuat fokus pada keseluruhan fitur dan fungsional sistem. Sehingga sistem dapat disimpulkan telah sesuai dan dapat di *realies.*

Menurut Suryantara (2017) kelebihan pada XP dibangun dengan melakukan pemrograman berpasangan (*pair programming*). *Developer* didampingi oleh pihak klien dalam melakukan *coding* dan *unit testing* sehingga klien bisa terlibat langsung dalam pemrograman sambil berkomunikasi dengan *developer*. Selain itu perkiraan beban tugas juga diperhitungkan.

1. Menekankan pada kesederhanaan dalam pengkodean: "*What is the simplest thing that could possibly work*?" Lebih baik melakukan hal yang sederhana dan mengembangkannya besok jika diperlukan. Komunikasi yang lebih banyak mempermudah, dan rancangan yang sederhana mengurangi penjelasan.

2. Setiap *feed back* ditanggapi dengan melakukan tes, *unit test* atau *system integration* dan jangan menunda karena biaya akan membengkak (uang, tenaga, waktu).
3. Banyak ide baru dan berani mencobanya, berani mengerjakan kembali dan setiap kali kesalahan ditemukan, langsung diperbaiki.

Menurut Suryantara (2017) kelemahan model *Extreme Programming* yaitu:

1. *Developer* harus selalu siap dengan perubahan karena perubahan akan selalu diterima.
2. Tidak bisa membuat kode yang detail di awal (*prinsip simplicity* dan juga anjuran untuk melakukan apa yang diperlukan hari itu juga).

2.5 UML *(Unified Modelling Language)*

UML *(unified Modelling Language)* adalah bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Rosa and Shalahuddin, 2019).

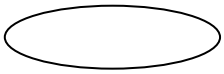
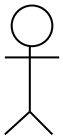
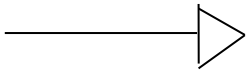
Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada UML *(Unified Modelling Language).*

2.5.1 *Use Case Diagram*

Use case yaiut mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Rosa and Shalahuddin, 2019).

Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada tabel 2.3.

Tabel 2.3 Simbol *Use Case Diagram*

No	Simbol	Deskripsi
1.		*Usecase* Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case.*
2.		Aktor Aktor seseorang/sesuatu yang berinteraksi dengan yang akan dibuat. diluar sistem informasi. Biasanya dinyatakan menggunakan kata benda
3.		Asosiasi/association merupakan komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor.
4.		Generalisasi *(generalization)* merupakan hubungan (umum – khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum
5.	<< *Include* >> 	Include berarti use case yang ditambahkan akan dipanggil saat use case tambahan dijalankan.
6.	<<*Extend*>> 	Ekstensi *(extend)* merupakan use case tambahan ke sebuah use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu.

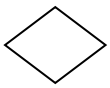
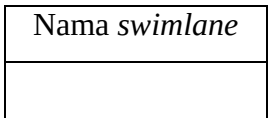
Sumber : (Rosa and Shalahuddin, 2019)

2.5.2 *Activity Diagram*

Activity diagram adalah menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa and Shalahuddin, 2019).

Berikut simbol-simbol yang akan digunakan dalam menggambarkan *activity diagram* dapat dilihat pada tabel 2.4 berikut ini :

Tabel 2.4 Simbol *Activity* Diagram

No.	Simbol	Keterangan
1.		Status awal aktivitas sitem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Percabangan *(Decision)* merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan *(Join)* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		*Swimlane* Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

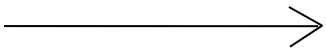
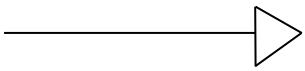
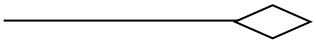
Sumber : (Rosa and Shalahuddin, 2019)

2.5.3 *Class Diagram*

Class diagram adalah mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa and Shalahuddin, 2019).

Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada tabel 2.5.

Tabel 2.5 Simbol *Class* Diagram

No.	Simbol	Deskripsi
1.	Nama_kelas *+Atribute* +Operasi	Kelas pada struktur sistem**.**
2.	Antar Muka/*Interface*  Nama_*Interface*	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.	Asosiasi / *Asociation* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan symbol
4.	Asosiasi Berarah / *Directed Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan symbol.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Agregasi / *aggregation* 	Relasi antar kelas dengan maksna semua bagian *(whole-part)*

Sumber: (Rosa and Shalahuddin, 2019)

2.6 Pengujian ISO 25010

ISO/IEC 25010 yaitu merupakan model kualitas sistem dan perangkat lunak yang menggantikan ISO/IEC 9126 tentang software engineering. Product quality ini juga digunakan untuk tiga model kualitas yang berbeda untuk produk perangkat lunak antara lain kualitas dalam model penggunaan, model kualitas produk, dan data model kualitas (ISO, 2011).

Model kualitas produk terdiri dari delapan karakteristik yang berhubungan dengan sifat statis perangkat lunak dan sifat dinamis dari sistem komputer. Model ini berlaku untuk sistem komputer dan produk perangkat lunak. Karakteristik yang didefinisikan oleh kedua model tersebut relevan untuk semua produk perangkat lunak dan sistem komputer. Karakteristik dan sub karakteristik memberikan terminologi yang konsisten untuk menentukan, mengukur dan mengevaluasi kualitas sistem dan perangkat lunak. Mereka juga menyediakan seperangkat karakteristik kualitas yang sesuai dengan persyaratan kualitas yang dapat dibandingkan untuk kelengkapan.

2.6.1 *Functional Suitability*

Sejauh mana perangkat lunak mampu menyediakan fungsi yang memenuhi kebutuhan yang dapat digunakan dalam kondisi tertentu. Karakteristik ini dibagi menjadi beberapa karakteristik yaitu.

1. *Functional completeness*, sejauh mana fungsi yang disediakan mencakup semua tugas dan tujuan pengguna secara spesifik.
2. *Functional correctness*, sejauh mana produk atau sistem menyediakan hasil yang benar sesuai kebutuhan.
3. *Functional appropriateness*, sejauh mana fungsi yang disediakan mampu memfasilitasi penyelesaian tugas dan tujuan tertentu.

2.6.2 *Compatibility*

Sejauh mana sebuah produk, sistem atau komponen dapat bertukar informasi dengan produk, sistem atau komponen dan/atau menjalankan fungsi lain yang diperlukan secara bersamaan ketika berbagi perangkat keras dan

environment perangkat lunak yang sama. Karakteristik ini dibagi menjadi 2 karakteristik yaitu.

1. *Co-existence*, sejauh mana produk atau sistem dapat menjalankan fungsi yang dibutuhkan secara efisien sementara berbagi sumber daya dengan produk atau sistem yang lain tanpa merugikan produk atau sistem tersebut.
2. *Interoperability*, sejauh mana dua atau lebih produk, sistem atau komponendapat bertukar informasi dan menggunakan informasi tersebut.

2.6.3 *Usability*

Sejauh mana sebuah produk atau sistem dapat digunakan oleh user tertentu untuk mencapai tujuan dengan efektif, eficiency, dan kepuasan tertentu dalam konteks penggunaan. Karakteristik ini terbagi menjadi beberapa karakteristik yaitu.

1. *Appropriateness recognizability*, sejauh mana pengguna dapat mengetahui apakah sistem atau produk sesuai kebutuhan mereka.
2. *Learnability*, sejauh mana produk atau sistem dapat digunakan oleh pengguna untuk mencapai tujuan tertentu yang belajar menggunakan sistem atau produk dengan efisien, efektif, kebebasan dari resiko dan kepuasan dalam konteks tertentu.
3. *Operability*, sejauh mana produk atau sistem mudah dioperasikan dan dikontrol.
4. *User error protection*, sejauh mana produk atau sistem melindungi pengguna terhadap membuat kesalahan.

5. *User interface aesthetics*, sejauh mana antarmuka pengguna dari produk atau sistem memungkinkan interaksi yang menyenangkan dan memuaskan pengguna.
6. *Accessibility*, sejauh mana produk atau sistem dapat digunakan oleh semua kalangan untuk mencapai tujuan tertentu sesuai konteks penggunaan.

2.6.4 *Reliability*

Sejauh mana sebuah sistem, produk atau komponen dapat menjalankan fungsi tertentu dalam kondisi tertentu selama jangka waktu yang ditentukan. Karakteristik ini terbagi menjadi beberapa subkarakteristik yaitu.

1. *Maturity*, sejauh mana produk atau sistem mampu memenuhi kebutuhan secara handal di bawah keadaan normal.
2. *Availability*, sejauh mana produk atau sistem siap beroperasi dan dapat diakses saat perlu digunakan.
3. *Fault tolerance*, sejauh mana produk atau sistem tetap berjalan sebagaimana yang dimaksud meskipun terjadi kesalahan pada perangkat keras atau perangkat lunak.
4. *Recoverability*, sejauh mana produk atau sistem mampu dapat memulihkan data yang terkena dampak secara langsung dan menata ulang kondisi system seperti yang diinginkan ketika terjadi gangguan.

2.6.5 *Security*

Sejauh mana sebuah produk atau sistem melindungi informasi dan data sehingga seseorang atau sistem lain dapat mengakses data sesuai dengan jenis dan

level otorisasi yang dimiliki. Karakteristik ini terbagi menjadi beberapa karakteristik yaitu.

1. *Confidentiality*, sejauh mana produk atau perangkat lunak memastikan data hanya bisa diakses oleh mereka yang berwenang untuk memiliki akses.
2. *Integrity*, sejauh mana produk atau perangkat lunak mampu mencegah akses yang tidak sah untuk memodifikasi data.
3. *Non-repudiation*, sejauh mana peristiwa atau tindakan dapat dibuktikan telah terjadi, sehingga tidak ada penolakan terhadap peristiwa atau tindakan tersebut.
4. *Accountability*, sejauh mana tindakan dari suatu entitas dapat ditelusuri secara unik untuk entitas.
5. *Authenticity*, sejauh mana identitas subjek atau sumber daya dapat terbukti menjadi salah satu yang diklaim.

2.6.6 *Portability*

Sejauh mana keefektifan dan efisiensi sebuah sistem, produk atau komponen dapat dipindahkan dari satu perangkat keras, perangkat lunak atau digunakan pada lingkungan yang berbeda. Karakteristik ini dibagi menjadi beberapa karakteristik yaitu.

1. *Adaptability*, sejauh mana produk atau sistem dapat secara efektif dan efisien disesuaikan pada perangkat lunak, perangkat keras dan lingkungan yang berbeda.
2. *Installability*, sejauh mana produk atau sistem dapat berhasil dipasang atau dihapus dalam lingkungan tertentu.

3. *Replaceability*, sejauh mana produk atau sistem dapat menggantikan produk atau sistem lain yang ditentukan untuk tujuan yang sama pada lingkungan yang sama.

2.6.7 *Performance Efficiency*

Kinerja relatif terhadap sumber daya yang digunakan dalam kondisi tertentu. Karakteristik ini terbagi menjadi beberapa subkarakteristik yaitu.

1. *Time behaviour*, sejauh mana respon dan pengolahan waktu produk atau sistem dapat memenuhi persyaratan ketika menjalankan fungsi.
2. *Resource utilization*, sejauh mana jumlah dan jenis sumber daya yang digunakan oleh produk atau sistem dapat memenuhi persyaratan ketika menjalankan fungsi.
3. *Capacity*, sejauh mana batas maksimum parameter produk atau sistem dapat memenuhi persyaratan.

2.6.8 *Maintainability*

Sejauh mana keefektifan dan efisiensi dari sebuah produk atau sistem dapat dirawat. Karakteristik ini terbagi menjadi beberapa subkarakteristik yaitu.

1. *Modularity*, sejauh mana sistem terdiri dari komponen terpisah sehingga perubahan atau modifikasi pada salah satu komponen tersebut memiliki dampak yang kecil terhadap komponen yang lain.
2. *Reusability*, sejauh mana aset dapat digunakan lebih oleh satu sistem atau digunakan untuk membangun aset lain.
3. *Analyzability*, tingkat efektivitas dan efisiensi untuk mengkaji dampak perubahan pada satu atau lebih bagian-bagian produk atau sistem, untuk

mendiagnosis kekurangan atau penyebab kegagalan produk, untuk mengidentifikasi bagian yang akan diubah.

4. *Modifiability*, sejauh mana produk atau sistem dapat dimodifikasi secara efektif dan efisien tanpa menurunkan kualitas produk yang ada.
5. *Testability*, tingkat efektivitas dan efisiensi untuk membentuk kriteria uji dari produk, sistem atau komponen dan uji dapat dilakukan untuk menentukan apakah kriteria tersebut telah terpenuhi

Berdasarkan tahapan-tahapan ISO 25010 tersebut maka peneliti menggunakan pengujian terhadap kualitas perangkat lunak berupa aplikasi *web* dapat dinilai dari tiga aspek, yaitu kemudahan penggunaan, fungsional dan efisiensi kinerja..