BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Pada penelitian ini, penulis menggunakan beberapa penelitian sebelumnya dan serupa, sebagai pendukung penelitian yang sedang dilakukan. Dibawah ini merupakan tinjauan pustaka yang sudah diteliti sebelumnya dan serupa :

Tabel 2. 1 Tinjauan Pustaka

Nomor	**Detail Jurnal**	
1	Judul	Rancang Bangun Sistem Informasi Staycation Berbasis Web Dengan Implementasi Teknologi Mern Stack
	Tahun Terbit	2021
	Penulis	(Solahudin and Dwi Nuryana, 2021)
	Metodologi Penelitian	*Extreme Programming*
2	Judul	Penerapan Teknologi Stack MERN pada Aplikasi Service Manajemen Bengkel Berbasis Web
	Tahun Terbit	2022
	Penulis	(Maulana and Santoso, 2022)
	Metodologi Penelitian	WDLC (*Web Development Life Cycle*)
3	Judul	Rancang Bangun Aplikasi Reparasi Panggilan " Repair Calling '' Berbasis Android
	Tahun Terbit	2020
	Penulis	(Ronaldo and Sanjaya, 2021)
	Metodologi Penelitian	*Rapid Aplication Development (RAD)*
4	Judul	Rancang Bangun Aplikasi Online Sistem Pemesanan Jasa Tukang Bangunan Berbasis Lokasi
	Tahun Terbit	2020
	Penulis	(Janis *et al.*, 2020)
	Metodologi Penelitian	*Rapid Application Development*
5	Judul	Perancangan Aplikasi Home Service Menggunakan Progressive Web Application
	Tahun Terbit	2019
	Penulis	(Atikah and Huda, 2019)
	Metodologi Penelitian	*Waterfall*

Tabel 2. 2 Tinjauan Pustaka Lanjutan

6	Judul	Rancang Bangun Sistem Pelayanan Home Service Pada Yoyena Optic Berbasis Web
	Tahun Terbit	2020
	Penulis	(Hernanda and Nurmiati, 2021)
	Metodologi Penelitian	*Prototype*

Penelitian diatas adalah dasar-dasar penelitian sebelumnya yang dapat dijadikan tinjauan pustaka pada penelitian ini. Berikut merupakan penjelasan tinjauan pustaka diatas, diantaranya:

1. Tinjauan Pustaka 1

Penelitian yang dilakukan oleh (Solahudin and Dwi Nuryana, 2021) membuat sistem informasi pemesanan tempat menginap, kesulitan mencari tempat penginapan yang mudah untuk mengakses destinasi wisata yang diinginkan menjadi fokus utama dalam penelitian ini, dengan menerapkan Metodologi *Extreme Programming* dan teknologi MERN Stack untuk mempermudah pencarian penginapan yang dekat dengan destinasi wisata saat berlibur menghasilkan Rancang Bangun Sistem Informasi Staycation Berbasis Web Dengan Implementasi Teknologi Mern Stack. Di sisi lain, kelebihan menggunakan teknologi ini adalah dapat mempercepat *performance* sistem dengan dipisahnya antara *client-side* dan *server-side*. Hasil dari penelitian ini adalah dapat mempermudah pengguna mendapatkan informasi tempat wisata yang terdapat penginapannya serta bisa memesannya untuk berlibur, aplikasi ini mirip seperti website Dinas Pariwisata Surabaya yang membedakannya adalah, pada aplikasi ini bisa memesan tiket wisata sekaligus tempat menginapnya, sementara website dinas pariwisata kota Surabaya hanya bisa membaca informasi wisata saja.

2. Tinjauan Pustaka 2

Penelitian yang dilakukan oleh (Maulana and Santoso, 2022) transaksi pencatatan penjualan service ataupun produk masih manual, rekapitulasi bulanan yang masih harus dihitung secara manual setiap minggu ataupun bulan. Pendataan produk spare part yang masih manual akan membuat data yang tertulis tidak konsisten, dengan tidak konsistennya data di dalam buku besar akan membuat seluruh data bengkel tidak akurat dan valid, dengan menerapkan teknologi *stack* MERN pada Aplikasi *Service* Manajemen Bengkel.

Sistem ini dapat melakukan pencatatan data *service*, *order*, barang yang hanya dilakukan sekali atau dua kali aksi saja dengan menerapkan metode WDLC *(Web Development Life Cycle)*. Kelebihan menggunakan teknologi *stack* MERN yaitu sistem yang dibuat menjadi lebih cepat, serta meningkatkan *performance* ramah SEO, dan mempercepat dan mempermudah pengolahan data. Hasil dari penelitian ini dapat membantu dan melihat perkembangan penjualan setiap minggu atau bulanan, serta data yang ditampilkan valid dan juga konsisten.

3. Tinjauan Pustaka 3

(Ronaldo and Sanjaya, 2021) melakukan penelitian pada reparasi alat-alat atau perabotan elektronik rumah tangga seperti TV, Kulkas, Radio, Mesin cuci, dan sebagainya. Namun di desa atau dusun jarang sekali ada yang bisa memperbaiki alat-alat elektronik tersebut dan warga harus membawanya ke toko reparasi yang ada di kota. Dengan demikian perlu adanya tukang reparasi panggilan yang bisa datang langsung ke rumah warga untuk memperbaiki alat-alat elektronik yang rusak.

Menerapkan pendekatan metodologi *Rapid Application Development* (RAD) penulis membuat Aplikasi Reparasi panggilan yang dapat memesan perangkat elektronik yang ingin di perbaiki setelah itu pekerja reparasi akan datang ke rumah untuk memperbaiki perangkat atau alat-alat elekronik yang rusak tersebut, dengan menggunakan teknologi Android Studio untuk membuat aplikasi yang berjalan di *mobile device,* Kelebihan menggunakan teknologi ini yaitu pengguna bisa dengan mudah mengaksesnya melalui *smartphone* mereka. Hasil dari penelitian ini adalah mempermudah masyarakat desa dalam memperbaiki barang elektronik yang rusak dengan memanggil tukang service elektronik melalui aplikasi tanpa harus pergi ke kota.

4. Tinjauan Pustaka 4

Penelitian yang dilakukan oleh (Janis *et al.,* 2020) mengetahui kebutuhan masyarakat akan seseorang yang memiliki kemampuan khusus dalam masalah pembangunan masih sulit untuk dicari, dan juga beberapa pemilik jasa tukang bangunan memiliki keterbatasan area atau jarak dalam melakukan pekerjaan mereka dikarenakan tidak adanya media yang efisien yang membantu mereka menemukan pekerjaan dengan mudah.

Pendekatan metode yang digunakan adalah *Rapid Application Development*, penulis membuat aplikasi online sistem pemesanan jasa tukang bangunan berbasis lokasi, sehingga dapat mempermudah *user* selaku *costumer* sebagai pelanggan untuk mencari tahu bagaimana dapat menemukan pekerja jasa tukang bangunan serta dapat melakukan pemesanan dan *worker* yang selaku pekerja dalam menerima pesanan hanya melalui *smartphone* yang terkoneksi dengan internet. Teknologi yang digunakan adalah Android Studio, Java dan Firebase, kelebihan menggunakan

teknologi tersebut yaitu mendukung *native method*, berorientasi pada objek, dinamis, NoSQL sehingga cepat dan responsif. Hasil dari penelitian ini berupa aplikasi *mobile device* yang dapat digunakan masyarakat selaku *costumer* sebagai pelanggan untuk mencari jasa tukang bangunan dan *worker* selaku penyedia jasa untuk menawarkan jasanya kepada pelanggan, aplikasi ini hampir mirip dengan aplikasi yang trend pada saat ini, yaitu Gojek dan Grab dimana pemesan melakukan pemesanan terhadap kendaraan yang akan ditumpangi dari lokasi pemesanan sampai ditempat tujuan pemesan.

Perbedaannya yaitu aplikasi ini hanya melakukan pemesanan terhadap jasa tukang bangunan dan jasa tukang bangunan itu sendiri yang akan pergi ke lokasi pemesan tersebut

5. Tinjauan Pustaka 5

Penelitian yang dilakukan oleh (Atikah and Huda, 2019) dibidang layanan bersih-bersih, saat ini pola kehidupan masyarakat berubah-ubah, baik laki-laki dan perempuan dituntut menghabiskan waktunya pergi ke luar rumah untuk bekerja, aktivitas di rumah yang semakin padat namun pelayanan tidak berubah mengakibatkan ketidakseimbangan, sehingga diperlukan orang untuk melakukan bersih-bersih dirumah. Waktu dulu penghuni rumah dapat mencari orang yang mau mengerjakan pekerjaan rumah di biro penyedia jasa tenaga kerja atau dengan menyewa orang yang bekerja sesuai bidangnya seperti menyewa tukang potong rumput. Sekarang penghuni rumah tidak memiliki waktu untuk pergi menyewa orang yang akan mengerjakan pekerjaan rumah sehingga penghuni rumah ingin menyewa orang secara praktis dan tidak menghabiskan banyak waktu.

Pendekatan yang digunakan adalah *Software Development Lyfe Cycle* (SDLC) metodologi *waterfall* penulis membuat aplikasi *Home Service* yang merupakan perantara penghuni rumah dengan orang yang mau melakukan pekerjaan rumah, aplikasi tersebut mempermudah pekerjaan rumah tangga, dibangun dengan teknologi *framework* yii2, dan *progressive web application* dan dibuat *responsive* sehingga bisa digunakan di *smartphone* maupun *browser personal computer*. Kelebihan teknologi tersebut yaitu cepat untuk pengembangan sistem, optimasi SEO yang baik, instalasi cepat daripada *native mobile*, membutuhkan sedikit ruang penyimpanan, *multiplatform*. Hasil dari penelitian ini berupa aplikasi *Home Service* yang memberikan layanan bersih-bersih di dalam maupun di luar rumah bagi mereka yang sibuk dengan kerjaannya, dapat dipesan dimanapun dan kapanpun.

6. Tinjauan Pustaka 6

(Hernanda and Nurmiati, 2021) melakukan penelitian terhadap kesehatan mata masyarakat Indonesia yang masih abai, *Optic* sebagai penyedia jasa pelayanan kesehatan mata masyarakat belum dimanfaatkan secara optimal. Pandemi Covid-19 yang terjadi di Indonesia memperburuk keadaan dan berdampak pada penyedia jasa pemeriksaan mata. Masyarakat menjadi enggan untuk memerikasakan kesehatan matanya karena sulit untuk mendapatkan jasa dari para penyedia *optic*. Dengan metode *Prototype* penulis merancang dan membangun aplikasi web untuk mempermudah konsumen dalam memperoleh jasa pemeriksaan kesahatan mata melalui aplikasi yang dibuat menggunakan teknologi PHP dan MySQL. Kelebihan dalam penggunaan teknologi tersebut ialah banyaknya library support, kecepatan pemuatan yang cepat meskipun koneksi internet lambat, lebih stabil, program yang ringkas dan *open-source*

Hasil dari penelitian ini berupa sistem pelayanan *home service* berbasis web yang memanfaatkan pelayanan *home service*, Sehingga antara konsumen dan pihak penyedia jasa saling menguntungkan satu sama lain. Konsumen mendapatkan kemudahan memperoleh jasa pemeriksaan mata, sedangkan pihak penyedia jasa dapat dengan mudah melayani para konsumen dengan lebih efektif tanpa harus khawatir akan terjadinya kerumunan.

2.2 Keaslian Penelitian

Hal yang menjadi pembeda antara penelitian yang dilakukan penulis dengan penelitian yang telah dilakukan sebelumnya sebagaimana terlampir pada tabel tinjauan pustaka, diantaranya adalah:

1) Penelitian ini membuat sebuah sistem marketplace untuk pelayanan jasa pangkas rambut
2) Pendekatan yang digunakan pada penelitian menggunakan metode *Extreme Programming* (XP)
3) *Platform* yang digunakan dalam perancangan aplikasi merupakan *platform* berbasis *website*
4) Teknologi yang digunakan adalah *MERN Stack*
5) Pengujian sistem penulis menggunakan pengujian *ISO 25010* dalam aspek *functionality* dan *usability*

2.3 Konsep E-commerce dan Marketplace

Menurut (Harmayani *et al*., 2020) *E-commerce* adalah penyebaran, pembelian, penjualan, pemasaran barang dan jasa melalui sarana elektronik seperti internet atau televisi, www, atau jaringan komputer lainnya. *E-commerce* melibatkan transfer data elektronik, pertukaran data elektronik, sistem manajemen inventori otomatis,

dan sistem pengumpulan data otomatis. E-commerce merupakan bagian dari *e-business*, dimana cakupan *e-business* lebih luas, tidak hanya sekadar perniagaan tetapi mencakup juga pengkolaborasian mitra bisnis, pelayanan nasabah, lowongan pekerjaan dan lain lain. Selain teknologi www, *e-commerce* juga memerlukan teknologi basisdata (*database*), surat elektronik (*e-mail*), dan bentuk teknologi komputer yang lain seperti halnya sistem pengiriman barang, dan alat pembayaran.

Pada saat ini, situs *e-commerce* banyak bermunculan di dunia, termasuk Indonesia. Umumnya ada beberapa jenis *e-commerce*. Berikut model *E-commerce* yang ada di Indonesia sebagai berikut:

1. Iklan baris, merupakan salah satu bentuk e-commerce yang tergolong sederhana, bisa dianggap sebagai evolusi dari iklan baris yang biasanya ditemui di koran-koran ke dalam dunia *online*. Penjual menggunakan media sosial atau forum untuk beriklan, biasanya tidak bisa langsung menyelesaikan transaksi pada website yang bersangkutan.
2. *Retail*, merupakan jenis *e-commerce* yang dimana semua proses jual-beli dilakukan melalui yang sudah diterapkan oleh situs *retail* yang bersangkutan. Oleh karena itu, kegiatan jual-beli di *retail* relatif aman, namun biasanya pilihan produk yang tersedia tidak terlalu banyak, atau hanya fokus ke satu-dua kategori produk.
3. *Marketplace*, bisa dianggap sebagai jasa *mall online*, namun yang berjualan bukan penyedia *website*, melainkan anggota-anggota yang mendaftar berjualan di website. *Marketplace* umumnya menyediakan lapisan keamanan tambahan untuk setiap transaksi yang terjadi, seperti sistem pembayaran *escrow* atau lebih umum dikenal sebagai rekening bersama.

Jadi setiap terjadi transaksi di dalam sistem marketplace tersebut, pihak *marketplace* akan menjadi pihak ketiga yang menerima pembayaran dan menjaganya hingga produk sudah dikirimkan oleh penjual dan diterima oleh pembeli. Setelah proses pengiriman selesai, barulah uang pembayaran teruskan ke pihak penjual.

Model bisnis pada *e-commerce* sebagai berikut:

1. Bisnis ke Bisnis (B2B)

   B2B melakukan transaksi elektronik antara satu perusahaan dan lainnya. Jenis model ini biasanya digunakan oleh produsen dan grosir atau grosir dan *reseller*

2. Bisnis ke Konsumen (B2C)

   B2C adalah kebalikan dari B2B. jenis model ini melakukan transaksi *online* antara produsen atau perusahaan dan konsumen akhir. Bisnis ini berhubungan langsung dengan konsumen atau kelompok individu dan bukan dengan perusahaan atau bisnis lain.

3. Konsumen ke Konsumen (C2C)

   C2C adalah model bisnis yang melibatkan transaksi antar konsumen. Kedua pihak tidak bertemu langsung, tetapi hanya melalui *platform online* pihak ketiga.

4. Konsumen ke Bisnis

   C2B adalah kebalikannya dari C2C. model bisnis ini melibatkan transaksi dari konsumen ke perusahaan. Konsumen akan menawarkan produk atau layanan kepada perusahaan yang membutuhkannya.

5. Bisnis ke Administrasi (B2A)

B2A mencakup aktivitas transaksi *online* yang terjadi antara perusahaan dan administrasi publik. Jenis ini melibatkan layanan pemerintah

6. Konsumen ke Administrasi (C2A)

   C2A memiliki model yang sama dengan B2A. sederhananya, C2A melibatkan transaksi antara konsumen atau individu dan administrasi publik.

7. Online ke Offline (O2O)

   O2O meluncurkan model bisnis untuk menarik pelanggan *online* untuk berbelanja di toko fisik. Pada dasarnya, konsep ini ingin menghubungkan saluran *online* dengan toko fisik.

Menurut (Wijaya, 2020) *Marketplace* pada dasarnya memiliki konsep yang mirip dengan pasar tradisional dalam perdagangan *offline*. Sebagaimana pasar tradisional, *marketplace* menyediakan tempat bagi orang-orang yang ingin berjualan. Hanya saja jika pasar tradisional merupakan pasar fisik yang mengharuskan penjual dan pembeli bertatap muka, maka marketplace merupakan pasar yang semua transaksinya dilakukan secara *online* tanpa penjual dan pembeli harus bertemu. Selain itu, jika berjualan di pasar tradisional, penjual tidak perlu membayar uang sewa. *Marketplace* sebenarnya merupakan sebuah *platform* yang menyediakan tempat bagi para pelaku bisnis yang ingin menjual produk-produk atau jasa mereka. Melalui *marketplace* para konsumen dapat menemukan berbagai jenis barang atau jasa yang mereka cari dari berbagai toko *online*. Pihak *marketplace* akan menampilkan produk yang dicari oleh pembeli produk atau jasa yang dicari oleh konsumen dari berbagai toko *online* yang terafiliasi dengan marketplace. Konsumen bisa memilih dan melakukan perbandingan produk atau

jasa dari suatu toko dengan toko yang lainnya, baik dari segi harga, kualitas maupun modelnya secara bersamaan dengan mudah.

2.4 Metode Pengembangan Sistem Agile

Menurut (Nyunando and Nasien, 2020) *agile* adalah sekelompok metodologi pengembangan perangkat lunak yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun. *Agile* memiliki pengertian bersifat cepat, ringan dan bebas bergerak, sehingga diperlukan inovasi dan *responsibility* yang baik antara tim pengembang dan klien agar kualitas dari perangkat lunak yang dihasilkan bagus Ada beberapa model pengembangan perangkat lunak yang termasuk metode pengembangan sistem *agile:*

1. *Extreme Programming*
2. SCRUM
3. *Dynamic Systems Development Method* (DSDM)

2.4.1 Metode Extreme Programming (XP)

Menurut (Pressman, 2010), Extreme programming (XP) Extreme Programming didasarkan pada nilai - nilai berikut:

a. Komunikasi

   Komunikasi yang efektif seharusnya terjadi antara perekayasa perangkat lunak dan para stakeholder lainnya (contoh: untuk membangun fitur - fitur dan fungsi - fungsi tertentu yang dibutuhkan bagi perangkat lunak). XP menekankan kolaborasi informal (namun bersifat lisan) antara pelanggan dan pengembang perangkat lunak, menekankan pentingnya pembentukan metafora-metafora yang efektif untuk mengkomunikasikan konsep -

konsep yang penting, menekankan pentingnya adaptasi terhadap umpan balik yang berkesinambungan, dan menekankan pentingnya dokumentasi yang produktif sebagai suatu media komunikasi.

b. Kesederhanaan

XP membatasi pengembang perangkat lunak melakukan perancangan hanya untuk kebutuhan – kebutuhan yang sifatnya mendesak alih - alih melakukan perancangan kebutuhan - kebutuhan yang diperlukan di masa depan. Tujuannya adalah untuk menciptakan rancangan yang sederhana yang dapat dengan mudah diimplementasikan dalam bentuk kode – kode program secara cepat. Jika rancangan tersebut selanjutnya harus ditingkatkan, rancangan yang bersangkutan dapat di refaktorisasi di waktu yang lain.

c. Umpan Balik

Umpan balik berasal dari tiga sumber: dari perangkat lunak yang diimplementasikan sendiri oleh si penulis perangkat lunak, dari para pelanggan, dan dari anggota tim perangkat lunak lain. Dengan melakukan perancangan dan kemudian menerapkan suatu strategi pengujian yang efektif, perangkat lunak (melalui langkah - langka pengujian) menyediakan bagi tim cepat umpan - umpan balik yang sangat bermanfaat. XP menggunakan unit pengujian sebagai taktik pengujian utama. Ketika masing - masing kelas dikembangkan, tim cepat mengembangkan suatu unit pengujian untuk menjalankan masing – masing operasi sesuai dengan fungsi yang berbeda - beda. Ketika suatu peningkatan dihantarkan ke pelanggan, user stories atau use cases yang diimplementasikan sebagai

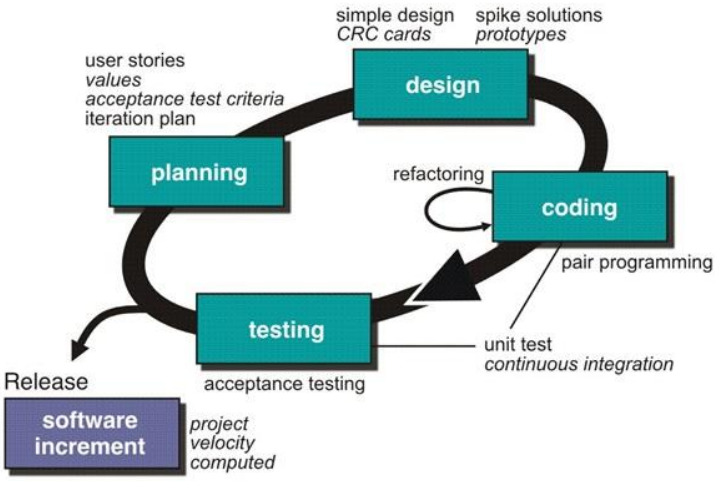
hasil peningkatan digunakan sebagai dasar untuk tes penerimaan. Tingkat dimana perangkat lunak menerapkan keluaran (output), fungsi, dan perilaku dari use case pada dasarnya (bagi tim cepat) merupakan suatu bentuk umpan balik yang sangat bermanfaat. Akhirnya, ketika kebutuhan – kebutuhan baru yang diperoleh sebagai bagian dari perancangan iteratif muncul, tim cepat menyediakan bagi pelanggan suatu umpan balik yang cepat mengenai dampak biaya dan jadwal atas kebutuhan – kebutuhan yang baru tersebut.

d. Keberanian

Kepatuhan ketat terhadap praktik - praktik XP tertentu menuntut keberanian. Sebuah istilah yang lebih tepat digunakan adalah disiplin. Misalnya, sering kali ada tekanan signifikan untuk melakukan perancangan demi kebutuhan di masa depan. Sebagian besar tim perangkat lunak cepat mengalah kemudian berdebat bahwa “melakukan perancangan untuk hari esok” akan menghemat waktu dan tenaga dalam jangka panjang. Sebuah tim XP yang cepat harus memiliki disiplin (keberanian) untuk melakukan perancangan hanya untuk saat ini dan mengakui bahwa kebutuhan - kebutuhan di masa depan dapat berubah secara drastis, sehingga menuntut pengerjaan ulang yang substansial atas rancangan - rancangan yang telah dibuat serta menuntut pengerjaan ulang yang substansial pula atas kode - kode program komputer yang telah diimplementasikan.

e. Rasa Hormat

Mengikuti masing-masing nilai penting tersebut, tim cepat menanamkan rasa hormat diantara para anggota tim perangkat lunak, diantara stakeholder lainnya dan anggota perangkat lunak, dan secara tidak langsung menanamkan rasa hormat untuk perangkat lunak itu sendiri. Ketika mereka berhasil mencapai penghantaran peningkatan perangkat lunak pada para pelanggan secara tepat waktu, tim cepat mengembangkan rasa hormat yang tumbuh pada proses XP.. Tahapan-tahapan XP dapat dilihat pada gambar 2.1 dibawah ini.



Gambar 2. 1 Proses Pengembangan Extreme Programming (XP)
Sumber (Borman, Priandika and Edison, 2020)

Pada fase dalam metodologi pengembangan sistem XP adalah sebagai berikut (dapat dilihat pada gambar 2.1 diatas):

1. Perencanaan (*Planning*)

   Pada tahapan ini merupakan tahapan yang diperlukan sebelum pengembang membuat sistem, tahapan ini penting karena dalam membuat

sebuah sistem harus direncanakan atau dianalisis kebutuhan-kebutuhan yang diperlukan *user*. Dengan cara mengidentifikasikan permasalahannya, kemudian menganalisis kebutuhan yang diperlukan, dan menetapkan jadwal untuk melaksanakan pembuatan sistem.

2. Perancangan (*Design*)

   Tahap selanjutnya adalah desain. Pada tahap ini pengembang merancang dengan membuat model yang diawali dengan pemodelan sistem, dilanjutkan dengan pemodelan arsitektur dan pemodelan basis data untuk memberikan gambaran tentang sistem yang akan dibangun.

   a. *Simple Design* adalah pengembang membangun perangkat lunak dengan desain yang sederhana. Dimulai dengan desain yang sederhana dilakukan menggunakan UML seperti *Use Case Diagram*, *Activity Diagram, Class Diagram,* dan *CRC Card*.
   b. *Prototype* adalah bagian perancangan berupa *user interface* dalam bentuk *wireframing* untuk mempermudah pengguna melihat desain sistem.

3. Pengkodean (*Coding*)

   Pada proses pengkodean yang dilakukan 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 ketidaksesuaian kode program sehingga dilakukan perbaikan guna mendapatkan hasil yang sesuai.

4. Pengujian (*Testing*)

   Tahapan ini merupakan tahapan terakhir setelah melakukan tahapan pengkodean. Pada tahapan ini pengembang sistem melakukan pengujian terhadap sistem yang sudah dibuat untuk mengetahui apakah ada kesalahan yang terdapat pada sistem saat sistem tersebut dijalankan, dan untuk mengecek sistem tersebut apakah sistem yang telah dibuat sudah sesuai dengan kebutuhan pengguna atau belum.

2.5 Analisis PIECES

Menurut (Dewi, 2018) , Metode PIECES adalah metode analisis sebagai dasar untuk memperoleh pokok-pokok permasalahan yang lebih spesifik. Dalam menganalisis sebuah sistem, biasanya akan dilakukan terhadap beberapa aspek antara lain analisis terhadap kinerja, informasi, ekonomi, pengendalian, efisiensi dan pelayanan, analisis ini disebut analisis PIECES (*Performance*, *Information*, *Economic*, *Control*, *Efficiency* and *Service*). Analisis PIECES ini sangat penting untuk dilakukan sebelum mengembangkan sebuah sistem informasi karena dalam analisis ini biasanya akan ditemukan beberapa masalah utama maupun masalah yang bersifat gejala dari masalah utama. Menurut (Ginting, 2020) metode ini menggunakan enam *variable* evaluasi yaitu :

a. *Performance* (Kinerja)

   Performance merupakan *variable* pertama dalam metode analisis PIECES. Dimana memiliki peran penting untuk menilai apakah proses atau prosedur yang ada masih mungkin ditingkatkan kinerjanya, dan melihat sejauh mana dan seberapa handalkah suatu sistem dalam berproses untuk menghasilkan tujuan yang diinginkan. Dalam hal ini kinerja diukur dari:

1. *Throughput*, yaitu jumlah pekerjaan yang dapat dilakukan atau dihasilkan pada saat tertentu.
2. *Response time*, yaitu waktu yang dibutuhkan untuk menyelesaikan serangkaian kegiatan untuk menghasilkan *output* tertentu.

b. *Information* (Informasi)

Informasi menjadi landasan untuk menilai apakah prosedur yang ada saat ini masih dapat diperbaiki sehingga kualitas informasi yang dihasilkan menjadi semakin baik. Informasi yang disajikan haruslah benar-benar mempunyai nilai yang berguna.

c. *Economy* (Ekonomi)

Analisis ini menilai apakah sistem yang ada saat ini masih dapat ditingkatkan manfaatnya atau diturunkan biaya pengeluarannya, Penggunaan sistem diharapkan dapat meningkatkan keuntungan atau penurunan biaya pengeluaran

d. *Control* (Kontrol)

Kontrol dilakukan untuk memastikan kinerja suatu sistem, diperlukan sebuah kontrol maupun pengawasan yang baik. Analisis ini digunakan untuk mengetahui seberapa besar pengendalian dan pengamanan yang dilakukan agar sistem yang digunakan tetap berjalan dengan seharusnya.

e. *Efficiency* (Efisiensi)

Hal utama yang perlu dipertanyakan dari suatu sistem yang digunakan adalah efektivitas dan efisiensi kinerjanya. Selain itu, juga harus diperhatikan alasan dari sistem itu dibuat dan digunakan. Suatu sistem harus mampu membantu permasalahan, khususnya dalam hal otomasi sistem dan

harus lebih unggul dari pada sistem manual. Analisis ini dilakukan untuk mengetahui efisiensi dari penggunaan sistem tersebut, sistem dikatakan efisien atau berhasil ketika dapat mencapai tujuan yang diinginkan sehingga tidak mengeluarkan banyak waktu dan tenaga yang berlebihan.

f. *Service* (Pelayanan)

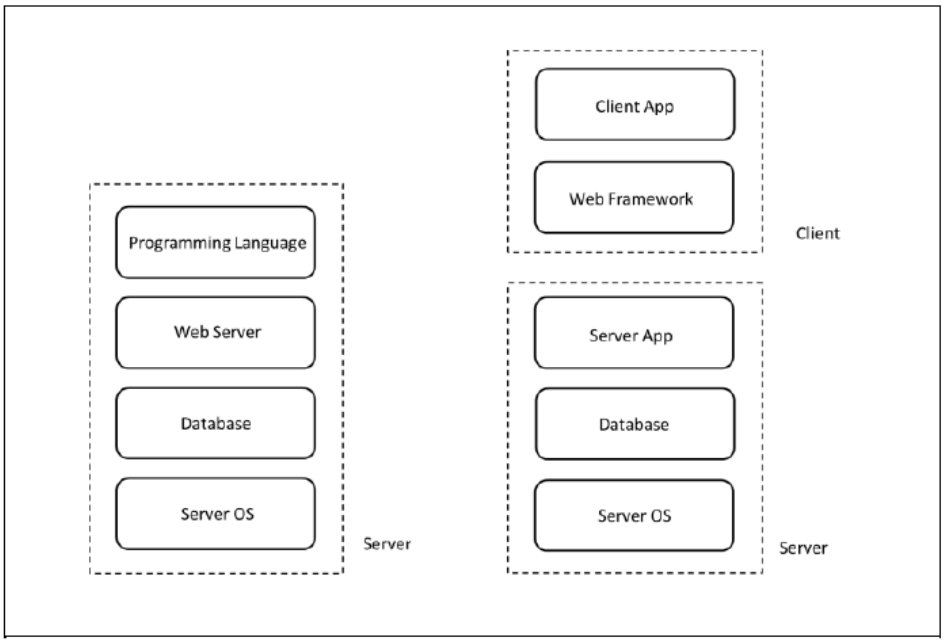
Sistem dapat dinyatakan berjalan dengan baik apabila diimbangi dengan pelayanan yang baik pula. Analisis ini dilakukan untuk mengetahui permasalahan yang ada pada pelaksanaan pelayanan dan bagaimana pelayanan tersebut berjalan, kualitas pelayanan sistem yang baik adalah dengan membuat layanan yang sangat *user friendly* untuk *end-user* (pengguna).

2.6 Fullstack Development

Menurut (Shropshire *et al.*, 2018) *Fullstack development* adalah konsep yang sedang tren dalam pengembangan perangkat lunak. *Fullstack development* diharapkan untuk merancang, membangun, dan mengimplementasikan solusi teknologi ujung ke ujung yang memenuhi kebutuhan bisnis. Mereka membantu dengan memastikan bahwa berbagai komponen kolektif membentuk solusi yang efektif dan terukur. Mereka juga dapat memenuhi peran manajer produk karena mereka dapat menerjemahkan kebutuhan bisnis ke dalam desain sistem ujung ke ujung.

Tujuan dari bagian ini adalah untuk memperkenalkan konsep tumpukan teknologi. Sebuah "*tech stack*" atau tumpukan teknologi juga dapat disebut sebagai tumpukan perangkat lunak atau hanya disebut tumpukan. Secara tradisional, sebagian besar tumpukan teknologi terdiri dari sistem operasi *server*, *server web*,

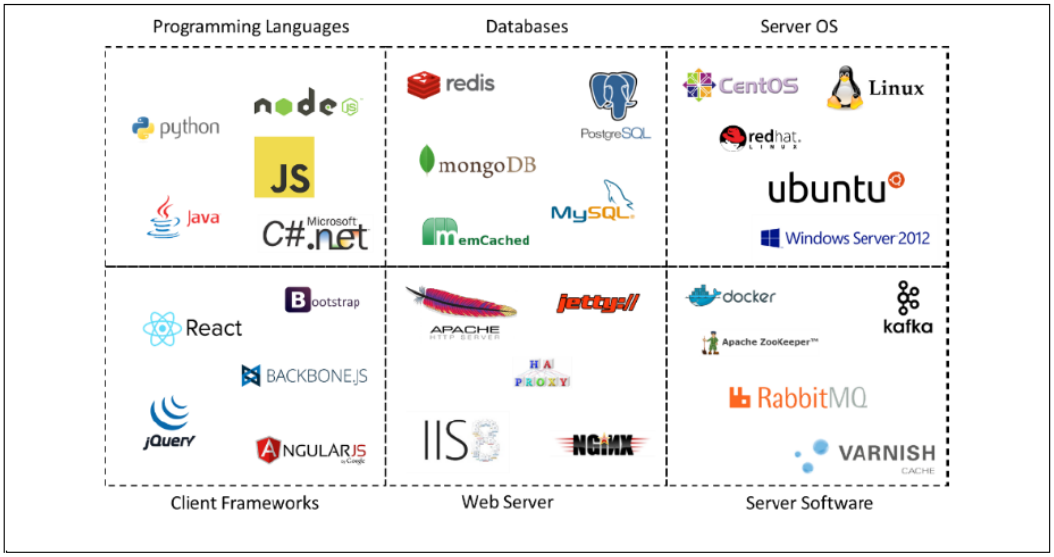
server basis data, dan bahasa pemrograman. Mereka dapat diperluas untuk memasukkan komponen tambahan seperti kerangka kerja pengembangan dan perangkat lunak sisi klien. Perlu dicatat bahwa sebagian besar komponen *stack* akan terintegrasi secara vertikal. Tujuan dari *tech stack* adalah untuk membuat layanan jaringan ke klien yang terdistribusi secara geografis. Layanan biasanya disampaikan melalui beberapa kombinasi aplikasi yang memberikan output dinamis kepada klien. *Server web* dan *database* dianggap sebagai bagian integral dari sebagian besar *tech stack*. Sebagian besar pemrosesan data dapat dilakukan di sisi *server*, sisi klien, atau didistribusikan secara merata di antara kedua titik.



Gambar 2. 2 Sisi Server Inti dan Arsitektur Stack Terdistribusi
Sumber (Shropshire *et al.*, 2018)

Pada beberapa organisasi mungkin mengandalkan *tech stack* tunggal untuk memberikan berbagai layanan perusahaan. Lainnya menerapkan beberapa *stack*, menggunakan untuk memberikan layanan yang berbeda. Meskipun ada sejumlah *tech stack* standar, tidak perlu mengikuti konvensi apapun saat memilih perangkat

lunak untuk setiap lapisan. Hal ini dilakukan untuk memilih perangkat lunak berdasarkan kebutuhan yang diharapkan, keahlian internal, dan masukan klien. Faktanya, beberapa organisasi membuat *tech stack* yang sangat disesuaikan untuk memenuhi kebutuhan mereka sendiri. Perangkat lunak yang paling populer untuk setiap lapisan bervariasi dengan waktu. perangkat lunak yang sedang tren untuk berbagai lapisan *tech stack* dapat dilihat pada gambar 2.3 dibawah ini



Gambar 2. 3 Komponen Stack Mainstream dan Emerging
Sumber (Shropshire *et al.*, 2018)

2.6.1 *Frontend*

Menurut (Mardan, 2018) *front-end* adalah istilah untuk aplikasi *browser*. *Browser* disebut klien karena dalam jaringan kita menggunakan komunikasi *client-server*. Pengguna berinteraksi dengan klien untuk membuat permintaan ke *server*, yang mengirimkan kembali tanggapan. Jadi *front-end* mengacu pada *browser* atau aplikasi klien. Seorang klien dapat menjadi aplikasi seluler juga. *"front-end"* digunakan untuk mendefinisikan aplikasi klien. contoh teknologi yang digunakan antara lain *Cascading Style Sheets* (CSS), *Hypertext Markup Language* (HTML), *Extensible Markup Language* (XML), JavaScript (JS), *JavaScript Object Notation*

(JSON), *Uniform Resource Identifier* (URI), *Hypertext Transfer Protocol* (HTTP), dan banyak teknologi lainnya.

2.6.2 *Backend*

Back-end merupakan sisi *server* atau *server side* dari sebuah *website* atau aplikasi. Untuk merancang atau mengembangkan *software* di sisi *server* yang berkaitan dengan logika serta *database*. Ini termasuk *platform* server seperti PHP, Python, Java, Ruby, Node.js, dan teknologi lainnya. Tujuannya adalah untuk memastikan bahwa aplikasi/*website* dapat tampil dan berguna dengan sebagaimana mestinya (Binaracademy.com, 2020).

2.7 MERN Stack

Menurut (Subramanian, 2019) Setiap aplikasi *web* dibangun menggunakan beberapa teknologi. Kombinasi dari teknologi ini disebut sebuah "*stack*," dipopulerkan oleh *LAMP stack*, yang merupakan akronim untuk *Linux, Apache, MySQL, PHP*, yang semua perangkat lunak *open source*. Saat pengembangan web semakin matang dan interaktivitasnya muncul, *Single Page Application* (SPA) menjadi lebih populer. SPA adalah paradigma aplikasi *web* yang menghindari pengambilan isi seluruh halaman *web* dari *server* untuk menampilkan konten baru. Alih-alih menggunakan panggilan ringan ke *server* untuk mendapatkan beberapa data atau cuplikan dan mengubah halaman *web*. Hasilnya terlihat cukup bagus dibandingkan ke cara lama memuat ulang halaman sepenuhnya. Ini membawa peningkatan kerangka kerja *front-end*, karena banyak dari pekerjaan dilakukan di bagian depan. Pada waktu yang hampir bersamaan, meskipun sama sekali tidak berhubungan, NoSQL *database* juga mulai mendapatkan popularitas.

MEAN *Stack* (MongoDB, Express, AngularJS, Node.js) adalah salah satu *stack open source* awal yang melambangkan pergeseran menuju SPA dan adopsi *NoSQL*. AngularJS, kerangka kerja *front-end* berdasarkan pola desain *Model View Controller* (MVC), menambatkan *stack* ini. MongoDB, NoSQL yang sangat populer *database*, digunakan untuk penyimpanan data persisten. Node.js, lingkungan *runtime* JavaScript sisi *server*, dan Express, *server web* yang dibangun di atas Node.js, membentuk *middle-tier*, atau *server web*. *Stack* ini bisa dibilang tumpukan paling populer untuk aplikasi *web* baru apapun hingga beberapa tahun yang lalu. Tidak benar-benar bersaing, tetapi React, teknologi *front-end* alternatif yang dibuat oleh Facebook, telah mendapatkan popularitas dan menawarkan alternatif untuk AngularJS. Dengan demikian menggantikan "A" dengan "R" di MEAN, untuk memberi kita *Stack* MERN. dikatakan "tidak persis" karena React bukanlah *framework* MVC yang lengkap. Ini adalah sebuah *library* JavaScript untuk membangun antarmuka pengguna.

2.7.1 Javascript

Menurut (Sahi, 2020), Javascript diperkenalkan pertama kali oleh Netscape pada tahun 1995. Pada awalnya bahasa ini dinamakan “LiveScript” yang berfungsi sebagai bahasa sederhana untuk *browser* Netscape Navigator 2. Javascript adalah bahasa yang berbentuk kumpulan skrip yang pada fungsinya berjalan pada suatu dokumen HTML, sepanjang sejarah internet bahasa ini adalah bahasa skrip pertama untuk web. Javascript ini adalah bahasa pemrograman yang digunakan dalam pengembangan website agar lebih dinamis dan interaktif, JavaScript dapat meningkatkan fungsionalitas pada halaman web. Bahkan dengan JavaScript ini bisa membuat aplikasi, *tools*, atau bahkan game pada web.

2.7.2 React JS

Menurut (Socs. Binus, 2019), *React JS* adalah sebuah pustaka/*library* javascript yang bersifat *open source* untuk membangun *User Interface* yang dibuat oleh Facebook. React JS hanya mengurusi semua hal yang berkaitan dengan tampilan dan logika di sekitarnya. *React JS* dapat mendesain tampilan sederhana untuk setiap level dalam aplikasi, sehingga dapat digunakan untuk membuat dan mengembangkan pembuatan aplikasi berbasis web.

Fitur yang dapat diunggulkan oleh React JS yaitu :

1. *Declarative*

   React dapat membuat UI (*User Interfaces*) yang interaktif, sehingga dapat dengan mudah membuat desain yang simple untuk di setiap state di dalam aplikasi. *Declarative views* dapat membuat kode lebih mudah untuk di prediksi dan lebih mudah untuk di *debug*.

2. *Component – Based*

   Dapat membuat *Encapsulated Component* yang dapat mengelola statusnya sendiri, lalu menyusunnya untuk membuat UI yang kompleks.

3. *Learn Once, Write Anywhere*

   Developer dapat men-*develop* fitur baru menggunakan react tanpa mengubah kode sebelumnya, react juga dapat bekerja menggunakan *Node JS* dan mobile apps menggunakan *React Native*.

2.7.3 Node JS

Menurut (Mardan, 2018), Node.js merupakan *runtime environment* untuk JavaScript yang bersifat *open-source* dan *cross-platform*. Dengan Node.js kita dapat menjalankan kode JavaScript di mana pun, tidak hanya terbatas pada

lingkungan *browser*. Node.js memiliki performa yang tinggi dengan teknologi *event-driven asynchronous I/O* untuk membangun *server* yang *scalable* dan efisien. Node.js menjalankan V8 JavaScript *engine* (yang juga merupakan inti dari Google Chrome) di luar *browser* dan banyak modul C++.

Node.js memiliki ciri khas yang sangat unik yaitu kemudahan dimana web server dan kode program nya tidak terpisah seperti pada bahasa pemrograman web lainnya. Pada node.js pengembang dapat mengaktifkan *web server* memodifikasinya dan menampilkan *content* dalam satu *file* tanpa harus mengubah banyak konfigurasi (Arhandi, Asmara dan Firdausi, 2019).

2.7.4 Express JS

Menurut (Yuliana, Rigustama and Zahra, 2021), Express.js adalah sebuah *framework* atau kerangka kerja yang terdapat dalam Node.js yang mudah dikembangkan oleh TJ Holowaychuk pada tahun 2010 lalu, untuk pengembangan aplikasi *web*, *service* API, *routing*, maupun *security*. Dikembangkannya Express.js akan berguna pada penggunaan *design pattern* yang dapat disesuaikan dengan arsitektur apapun sehingga sangat powerfull dan fleksible. Dengan pembuatan arsitektur dengan Express.js, API yang digunakan juga sangat ringan dan tidak memakan *resource* yang sangat banyak sehingga sangat memangkas biaya yang digunakan untuk pengembangan *website* selanjutnya.

Express JS juga punya arsitektur MVC (*Model View Controller*). Dengan begitu, setiap data diolah pada *Model*, dihubungkan melalui *Controller*, lalu ditampilkan menjadi informasi melalui *View* (Niagahoster, 2021).

2.7.5 Mongo DB

Menurut (Laksono and Amin, 2019), MongoDB adalah salah satu jenis dari perangkat lunak pengelola *database* jenis NoSQL yang merupakan *database* non-relasional. Metode yang dipakai oleh MongoDB adalah *document oriented database* yang semua data dirangkum dan disimpan dalam bentuk dokumen seperti *JavaScript Object Notation* (JSON).

2.8 Tailwind CSS

Menurut (Tailwindcss, 2019), *Tailwind CSS* merupakan kerangka kerja yang dikembangkan oleh Adam Wathan, *Tailwind CSS* bersifat *utility-first* untuk membangun desain *interface* dengan cepat. Maksud dari *utility-first* adalah pengembang tidak perlu lagi memikirkan panjang nama kelas untuk komponen HTML nya. Konsep yang digunakan oleh Tailwind CSS adalah :

1. *Utility-first*
2. *Responsive Design*
3. *Dark Mode*
4. *Reusing Style*
5. *Adding Custom Style*
6. *Functions & Directive*

2.9 UML (*Unified Modeling Language*)

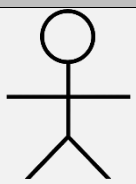
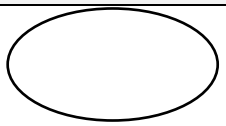
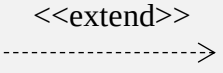
Menurut (Rosa dan Salahuddin, 2018), *Unified Modeling Language* (UML) adalah "bahasa" yang telah menjadi standar industri untuk visualisasi, merancang dan mendokumentasikan sistem perangkat lunak. UML menawarkan sebuah

standar untuk merancang model sebuah sistem. Terdapat diagram-diagram dalam UML, sebagai berikut:

2.9.1 *Use Case Diagram*

Menurut (Rosa dan Salahuddin, 2018), *Use case* diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu sama lain atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan apa saja yang berhak menggunakan fungsi-fungsi itu. Berikut adalah simbol-simbol yang digunakan dalam use case diagram dapat dilihat di tabel 2.3

Tabel 2. 3 Simbol Diagram *Use Case*

No	Simbol	Nama	Keterangan
1		Aktor	Orang, proses, sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi itu sendiri.
2		*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
3		*Association*	Berkomunikasi antara aktor dengan *use case* atau *use case* memiliki interaksi dengan aktor.
4		*Generalization*	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
5		*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.

Tabel 2. 4 Simbol Diagram *Use Case*

6	<<include>> ---------->	*Include*	Relasi *use case* tambahan sebuah *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini

Sumber (Rosa and Salahuddin, 2018)

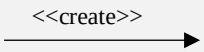
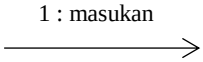
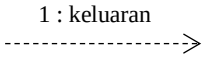
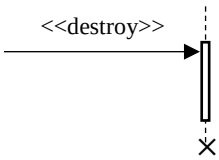
2.9.2 *Sequence Diagram*

Menurut (Rosa and Salahuddin, 2018), diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Adapun beberapa simbol yang terdapat pada sekuen diagram dapat dilihat pada tabel 2.3 di bawah ini.

Tabel 2. 5 Simbol *Sequence* Diagram

No	Simbol	Nama	Keterangan
1		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		Garis hidup*/lifeline*	Menyatakan kehidupan suatu objek
3	Nama objek : nama kelas	Objek	Menyatakan objek yang berinteraksi pesan

Tabel 2. 6 Simbol *Sequence* Diagram

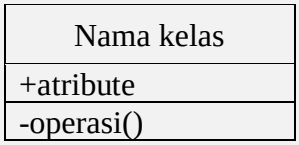
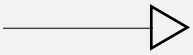
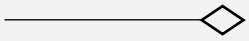
4		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
5		Pesan tipe create	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6		Pesan tipe call	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
7		Pesan tipe send	Menyatakan suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
8		Pesan tipe return	Menyatakan suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9		Pesan tipe destroy	Menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Sumber (Rosa and Salahuddin, 2018)

2.9.3 *Class Diagram*

Menurut (Rosa and Salahuddin, 2018), Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas memiliki atribut dan metode atau operasi. Atribut dan metode dapat memiliki salah satu sifat berikut:

Tabel 2. 7 Simbol *Class* Diagram

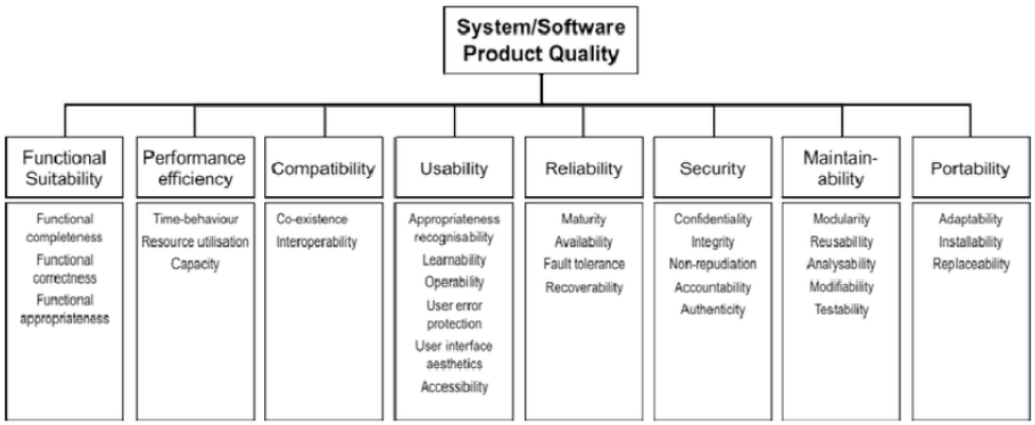
No	Simbol	Nama	Keterangan
1		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
2		Antarmuka / *interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3		*Class*	Kelas pada struktur sistem
4		*Directed association*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai *multiplicity*
5		Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
6		*Dependency*	Ketergantungan antarkelas
7		*Generalization*	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

Sumber (Rosa and Salahuddin, 2018)

2.10 Pengujian ISO 25010

(Lamada, Miru ands Amalia, 2020) mengungkapkan bahwa di antara berbagai standar pengujian, ISO 9126 dan ISO 25010 adalah standar internasional untuk pengujian perangkat lunak. Standar ISO 25010 dikembangkan sebagai alternatif standar ISO 9126 berdasarkan perkembangan ICT (Information and Communication Technology). Standar 25010 memiliki delapan karakteristik :

kompatibilitas fitur, keandalan, efisiensi, ketersediaan, keamanan, kompatibilitas, pemeliharaan, dan probabilitas.



Gambar 2. 4 Karakteristik Pengujian ISO 25010

1. *Functionality Suitability*, menggunakan alat penelitian berupa test case dengan skala Guttman. Skala Guttman digunakan untuk memberikan jawaban pasti atas masalah yang ingin anda angkat.
2. *Performance Efficiency*, dilakukan untuk menguji tingkat kinerja aplikasi yang dikembangkan.
3. *Compatibility*, merupakan kemampuan suatu komponen atau sistem untuk bertukar informasi.
4. *Usability*, Ini dilakukan dengan menganalisis umpan balik pengguna menggunakan skala 5 pilihan.
5. *Reliability*, untuk menguji keandalan atau keterpercayaan sistem.
6. *Security*, adalah sejauh mana sistem atau produk yang menyediakan layanan melindungi dari akses, penggunaan, modifikasi, gangguan, atau pengungkapan yang berbahaya.
7. *Maintainability*, menggunakan alat ukur yang telah diuji oleh peneliti langsung di lapangan kegiatan, sesuai dengan alat uji yang disebutkan Land,

pengujian ini terdiri dari 3 aspek, yaitu alat ukur, perhitungan konsistensi dan kesederhanaan.

8. *Portability*, adalah sejauh mana suatu sistem atau produk dapat dipindahkan dari satu ruang ke ruang lainnya.

Pada aspek *functionality suitability* Skala pengukuran dengan tipe guttman didapat jawaban yang tegas, yaitu "Ya" atau "Tidak" , Ya bernilai 1 dan Tidak bernilai 0 pada tiap item.

Tabel 2. 8 Bobot Jawaban *Functionality Suitability*

Jawaban	Ya	Tidak
Nilai	1	0

Sumber (Sugiyono, 2018)

Seluruh penilaiannya dihitung dengan menggunakan skala *likert,* sebagai berikut :

$$Klasifikasi\ Persentase = \frac{Bobot\ jawaban}{Bobot\ maksimal\ jawaban} = \times 100\%$$

Tabel 2. 9 Kriteria Persentase Hasil Uji

Persentase Kelayakan	**Kriteria**
$\geq 50\%$	Dapat diterima
$< 50\%$	Ditolak

Sumber (Sugiyono, 2018)

Pada aspek *Usability* pengujian dapat dilakukan dengan menghitung menggunakan skala *likert* menurut (Sugiyono, 2018). Skor untuk alternatif jawaban untuk setiap item sebagai berikut:

1. Skor 5 untuk jawaban sangat setuju,

2. Skor 4 untuk jawaban setuju,

3. Skor 3 untuk jawaban kurang setuju,

4. skor 2 untuk jawaban tidak setuju

5. Skor 1 untuk jawaban sangat tidak setuju.

Skor tersebut dihitung menggunakan rumus konversi ke persentase skor untuk mencari kriteria interpretasi skor hasil pengujian usability. Berikut ini adalah rumus konversi ke persentase skor.

$$Hasil = \frac{Skor\ Perolehan}{Skor\ Maksimal} \times 100\%$$

Hasil dari persentase tersebut kemudian dibandingkan dengan tabel kriteria interpretasi skor. Kriteria interpretasi skor dapat dilihat pada Tabel 2.7

Tabel 2. 10 Kriteria Interpretasi Skor

Persentase Skor	Keterangan
0%-20%	Sangat Tidak Baik
21%-40%	Tidak Baik
41%-60%	Netral
61%-80%	Baik
81%-100%	Sangat Baik

Sumber (Lamada, Miru and Amalia, 2020)