

BAB II

LANDASAN TEORI

2.1. Kursus Mengemudi

Lembaga Kursus adalah salah satu penyelenggara pendidikan diluar sekolah resmi (non-formal) untuk mengembangkan kemampuan dan keterampilan diri (Mahdy et al., 2021). Kursus Mengemudi secara spesifik dapat diartikan suatu pendidikan untuk mengembangkan kemampuan dan keterampilan diri dalam mengemudikan kendaraan khususnya mobil. KBBI sendiri mendeskripsikan kursus sebagai pelajaran tentang suatu pengetahuan atau keterampilan, yang diberikan dalam waktu singkat. Atau bisa juga diartikan sebagai lembaga di luar sekolah yang memberikan pelajaran serta pengetahuan atau keterampilan yang diberikan dalam waktu singkat.

2.2. Sistem Informasi Berbasis Web

Sistem Informasi menurut Richard Vidgen adalah sebuah kumpulan komponen-komponen yang berinteraksi, komponen yang dimaksud adalah manusia, prosedur-prosedur, dan teknologi-teknologi yang ada, dimana komponen-komponen tersebut secara bersamaan mengumpulkan, memproses, menyimpan, dan mendistribusikan informasi untuk mendukung pengontrolan, pembuatan keputusan dan pengelolaan organisasi/perusahaan. Sistem informasi sendiri berisi informasi tentang organisasi/perusahaan terkait, contohnya, kondisi tentang operasional internal mereka, dan tentang lingkungan di dalam perusahaan tersebut, sebagai contoh informasi tentang para pelanggan, para supplier, dan kompetitor-kompetitor yang ada. Tanpa adanya sistem informasi, sebuah organisasi sulit untuk bertahan. Namun, bukan berarti bahwa sistem informasi harus menggunakan teknologi-teknologi informasi yang berbentuk komputer-komputer atau jaringan internet dan komunikasi, karena banyak sekali bentuk sistem informasi. Organisasi atau perusahaan sangat bergantung pada sistem informasi, meskipun aspek-aspek

formal dari sistem informasi ini masih menggunakan sistem pengarsipan berupa kertas pada era sebelum adanya teknologi informasi itu sendiri.

Menurut (Manullang, A. H., et al., 2021), *website* pada dasarnya adalah kumpulan halaman yang saling terhubung melalui jaringan internet yang biasanya berisi bermacam-macam media, media yang dimaksud diantaranya adalah teks, gambar, audio, video, dan lain lain. Secara analogi, *website* dapat diibaratkan sebagai sebuah perpustakaan digital tanpa batas halaman. Setiap halaman dalam *website* berpotensi berisi informasi yang beragam, dan pengguna dapat dengan mudah berpindah antar halaman hanya dengan mengklik tautan.

2.2.1. *Hyper Text Markup Language (HTML)*

HTML adalah salah satu istilah pemrograman yang paling dikenal oleh masyarakat umum, namun, banyak perdebatan yang mengatakan bahwa sebenarnya HTML bukanlah sebuah bahasa pemrograman. Ada juga beberapa orang yang beranggapan bahwa karena dalam menulis HTML diperlukan setidaknya pemahaman dasar tentang pemrograman, maka HTML dianggap sebagai bahasa pemrograman. Mengacu dari jurnal yang ditulis oleh (Sari et al., 2022) HTML merupakan salah satu bahasa pemrograman yang digunakan untuk membuat website. HTML biasa ditulis untuk membantu perancangan struktur dasar halaman website atau bisa juga dianggap sebagai pondasi awal untuk menyusun kerangka halaman website secara lebih terstruktur sebelum masuk ke tahap desain dan fungsionalitas.

2.2.2. *Cascading Style Sheet (CSS)*

Pada era modern saat ini, hampir tidak bisa kita temui rangkaian kode HTML tanpa dilengkapi CSS. Menurut (Sari et al., 2022) CSS adalah bahasa pemrograman yang ditujukan untuk memberikan modifikasi tampilan elemen-elemen web seperti *font*, *outline*, *background*, menyesuaikan tampilan website dengan ukuran layar, dan sebagainya. Jika HTML digunakan untuk menempatkan konten-konten apa saja yang ingin ditampilkan pada sebuah halaman web, CSS digunakan untuk

memberikan pemahaman kepada mesin untuk melakukan modifikasi terhadap tampilan elemen dan penataan tata letak lebih lanjut.

Karena sejatinya, HTML tidak dirancang untuk menentukan aspek visual pada sebuah desain website. Sebab, fokus utama dari HTML adalah membagi struktur sebuah halaman website. Oleh karena itu, dikenalkan skrip “pendamping” untuk memperindah tag-tag HTML yaitu CSS. Selain itu, tujuan penggunaan CSS adalah untuk memberikan kesan konsisten di seluruh website.

2.2.3. Tailwind CSS *Framework* (CSS)

(Somi, M., 2023) dalam jurnalnya menjelaskan bahwa *Framework Tailwind CSS* adalah *framework* CSS yang mengutamakan penggunaan kelas utilitas yang paling populer dan bertujuan untuk membangun tampilan antarmuka khusus dengan cepat dan mudah. Maksudnya, berbeda dengan *Bootstrap* yang tergolong *framework UI kits*, *tailwind* tidak menyediakan komponen-komponen siap pakai. *Tailwind* tidak mempunyai tema bawaan. Dengan *tailwind*, kita memberikan *style* dengan mengetikkan kelas-kelas yang sudah ditentukan sebelumnya ke kodingan HTML yang kita kerjakan.

Lebih lanjut lagi, (Somi, M., 2023) menjelaskan bahwa dengan menggunakan *tailwind*, memberikan kita kemampuan untuk mempercepat proses pemrograman tanpa kita harus menulis kode CSS di *file* lain melainkan, menulisnya secara bersamaan di kode markup HTML. Selain itu, *tailwind* memberikan kita kemampuan untuk melakukan kustomisasi secara penuh sesuai keinginan kita. Efek samping dari menggunakan *tailwind* adalah kode HTML yang dihasilkan akan jauh lebih panjang

2.2.4. Javascript

Halaman website yang dihasilkan dari hanya menggunakan bahasa HTML & CSS cenderung statis dan kurang menarik. Untuk membuat tampilan yang lebih dinamis diciptakan sebuah bahasa pemrograman baru demi mengatasi kekurangan ini, yakni *Javascript*. Sebagai referensi, (Noviantoro et al., 2022) menjelaskan

bahwa *Javascript* adalah salah satu bahasa pemrograman yang digunakan untuk dapat berjalan di web browser. Pada awal pengembangan bahasa pemrograman ini sempat disebut dengan nama *Mocha*, kemudian berubah penamaannya menjadi *Live-Script*, dimana ketika masa rilis, diubah lagi menjadi *Javascript*. Lebih jauh lagi dijelaskan bahwa *Javascript* adalah *script* program berbasis *client* yang dieksekusi oleh browser sehingga membuat halaman web melakukan tugas-tugas tambahan yang tidak bisa dilakukan hanya dengan memanfaatkan HTML biasa. Selain alasan yang kami sebut diatas, beberapa interaksi yang ingin kami munculkan pada aplikasi ini tidak dapat diselesaikan hanya dengan menggunakan HTML dan CSS.

2.2.5. jQuery

(Sahrudin A. et al., 2023) menyebutkan pada jurnalnya menjelaskan jQuery sebagai berikut. Pada tahun 2006, John Resig memperkenalkan jQuery, sebuah *library* Javascript yang revolusioner. Sebelum hadirnya jQuery, para pengembang perangkat lunak harus menulis kode Javascript yang kompatibel dengan berbagai macam *browser*. jQuery hadir sebagai terobosan, memberikan para pengembang perangkat lunak kemampuan untuk menulis kode Javascript yang ringkas, mudah dipahami, serta dapat dieksekusi di berbagai *browser*.

2.2.6. MySQL

MySQL merupakan sistem manajemen basis data relasional (RDBMS) yang populer dan banyak digunakan di berbagai platform (Hermiati R. et al., 2021). Dikenal dengan keunggulannya dalam mendukung skrip PHP, MySQL menawarkan kemudahan integrasi dengan bahasa pemrograman web yang populer ini. Salah satu keunggulan utama MySQL adalah sintaks kuerinya yang sederhana dan mudah dipahami. Bahasa *Structured Query Language* (SQL) yang digunakan MySQL dirancang dengan intuitif, memungkinkan pengguna untuk mengelola data dengan efektif dan efisien. Selain itu, MySQL terkenal dengan kecepatannya yang luar biasa dalam memproses dan mengakses data. Kecepatan ini menjadikannya

pilihan ideal untuk aplikasi web yang membutuhkan performa tinggi dan skalabilitas yang baik.

2.2.7. *Hypertext Preprocessor (PHP)*

PHP atau *Hypertext Preprocessor* sebagaimana yang dijelaskan (Adrianto S., 2021) adalah sebuah bahasa pemrograman untuk membuat web yang bersifat *server-side scripting*, PHP memungkinkan untuk membuat halaman web bersifat dinamis. Selain itu, PHP membutuhkan *Database Management System (DBMS)* untuk dijalankan secara bersamaan. DBMS yang paling populer di kalangan pelajar pemrograman salah satunya adalah MySQL, namun, PHP juga mendukung DBMS lain seperti Oracle, Microsoft Access, Interbase, D-Base, PostgreSQL, dan DBMS yang lainnya.

Satu hal yang mungkin menjadi pertanyaan adalah apabila PHP dikenal dengan *Hypertext Preprocessor*, lalu apa kepanjangan PHP? Menurut tulisan yang diterbitkan melalui web sekawanstudio.com yang ditulis oleh (Miranda R.A., 2023) Pada tahun 1994, ketika Rasmus Lerdorf pertama kali menemukan *hypertext preprocessor*, beliau menggunakannya untuk memantau jumlah pengunjung atau yang sering kita sebut dengan *traffic website* dari halaman web pribadi nya atau dalam bahasa inggris dikenal dengan *Personal Home Page*. Alasan tersebutlah yang menyebabkan bahasa pemrograman ini dijuluki sebagai PHP.

2.2.8. *Framework PHP Laravel*

Menurut penjelasan dari (Subecs, 2021) Laravel merupakan *framework* PHP yang paling sering digunakan untuk *programmer* pemula dan berpengalaman. Laravel dianggap mampu mengurangi durasi pengembangan sistem perangkat lunak serta mempersiapkan pasar dengan metode PHP berorientasi objek yang lebih modern. *Syntax-syntax* ekspresif dan *function-function* modern yang dimiliki Laravel disukai oleh para *programmer* yang ingin mengembangkan web atau aplikasi yang lebih kompleks. Dengan menggunakan *framework* ini diyakini dapat mempermudah proses pengembangan karena Laravel menggunakan sistem paket

modular dimana modul-modul yang disediakan saling terkait satu sama lain, dimana kita bisa mengembangkan sistem perangkat lunak yang lebih luas lagi. *Framework* ini memberikan kita jalan pintas yang memungkinkan *programmer* berkonsentrasi terhadap masalah-masalah yang lebih penting.

2.3. Rekayasa Perangkat Lunak

Secara bahasa, *Software Engineering* atau yang sering disebut dengan Rekayasa Perangkat Lunak dalam bahasa Indonesia (RPL) tersusun dari 2 kata, *Software* dan *Engineering*. *Software* atau perangkat lunak bukan hanya sebuah program seperti yang diasumsikan banyak orang. Program adalah kode komputer yang dapat dieksekusi, yang tujuannya untuk melakukan satu atau lebih komputasi tertentu. Sedangkan *software* adalah kumpulan dari kode pemrograman komputer yang dapat dieksekusi, terorganisir dan terdokumentasi. Lalu, *Engineering* sendiri adalah semua hal yang meliputi pengembangan produk (baik fisik maupun digital) dengan menerapkan dan memanfaatkan prinsip-prinsip dan metodologi ilmiah yang tersusun dengan baik.

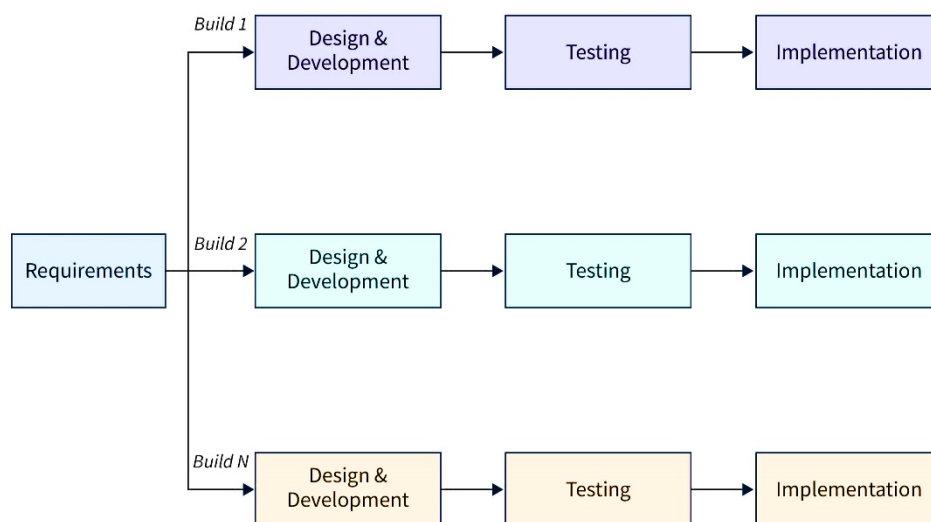
Sehingga, *Software Engineering* merupakan salah satu bagian dari *Engineering* yang berurusan dengan rekayasa perangkat lunak yang disusun secara baik dan menerapkan prinsip-prinsip, prosedur dan metodologi ilmiah. Hasil akhir dari RPL tentunya adalah perangkat lunak yang efisien, stabil dan bebas dari *bug*. Namun, apabila kita mengacu kepada pengertian menurut organisasi IEEE, RPL adalah (1) Penerapan dari sebuah pendekatan yang sistematis, disiplin, dan terukur untuk mengembangkan, mengoperasikan dan memelihara sebuah perangkat lunak; dalam hal ini penerapan rekayasa perangkat lunak. (2) Studi tentang pendekatan-pendekatan dari pernyataan diatas.

Selain IEEE, Fritz Bauer, ilmuwan komputer asal Jerman, mendefinisikan RPL sebagai "...pembentukan dan pemanfaatan prinsip teknik suara agar tercipta perangkat lunak yang ekonomis, yang stabil dan bekerja secara efisien di mesin sesungguhnya.". Mengutip dari modul pembelajaran Rekayasa Perangkat Lunak – Pendekatan Terstruktur & Berorientasi Objek karya (Bahar et al., n.d.) Rekayasa

perangkat lunak (*software engineering*) merupakan suatu disiplin ilmu yang membahas semua aspek produksi perangkat lunak (*software*), mulai dari tahap awal kajian spesifikasi / kebutuhan sistem sampai pemeliharaan sistem setelah digunakan (Sommerville, 2016). Pada definisi ini, ada dua istilah kunci: yang Pertama ‘Disiplin rekayasa’, yang berarti bahwa teknisi RPL membuat suatu alat bekerja. Mereka menerapkan teori, metode, dan alat bantu yang sesuai, selain itu mereka menggunakannya dengan selektif dan selalu mencoba mencari solusi terhadap permasalahan, walaupun tidak ada teori atau metode yang mendukung.

2.3.1. *Incremental Model*

Model pengembangan perangkat lunak *incremental model* dilakukan dengan membagi proyek perangkat lunak menjadi bagian-bagian yang lebih kecil atau dalam istilah *incremental model* disebut dengan *iterative build*. Gambar 2.1 dibawah ini adalah siklus atau skema pengembangan perangkat lunak untuk *incremental model*.



Gambar 2. 1 Siklus Pengembangan Perangkat Lunak *Incremental Model*
(Sumber: Sachan D., 2024)

Fitur utama dari model ini adalah proses iteratifnya, di mana setiap iterasi yang dilakukan selanjutnya dikembangkan, diuji, dan diintegrasikan satu per satu. Dengan membagi proyek ke bagian-bagian kecil, tim pengembang dapat

memprioritaskan fitur dan mengadaptasi perubahan-perubahan kecil dengan lebih efisien. Untuk lebih jelasnya kami akan menjelaskan setiap tahapan-tahapan di dalam *incremental model* :

- 1) **Requirements** merupakan tahap awal yang krusial dalam pengembangan perangkat lunak. Pada tahap ini, tim pengembang berfokus pada pengumpulan dan analisis kebutuhan perangkat lunak secara menyeluruh. Kebutuhan yang didapatkan berfungsi sebagai fondasi yang kokoh untuk memandu fase-fase pengembangan selanjutnya. Hasil akhir dari Requirements Gathering adalah dokumen yang berisi uraian rinci mengenai kebutuhan perangkat lunak. Dokumen ini harus ditulis dengan jelas, terstruktur, dan mudah dipahami oleh semua pihak yang terlibat dalam proyek.
- 2) **Design & Development** merupakan fase krusial dalam pengembangan perangkat lunak. Pada tahap ini, tim proyek akan melakukan pemrosesan hasil analisa kebutuhan di tahapan sebelumnya menjadi rancangan yang berfungsi sebagai acuan teknis untuk membangun perangkat lunak yang sesuai dengan kebutuhan yang telah ditetapkan sebelumnya. Desain yang dimaksud adalah desain arsitektur atau struktur dari perangkat lunak, desain interaksi perangkat lunak, dan terakhir desain tampilan perangkat lunak. Selanjutnya, tim pengembang akan berfokus pada pembangunan masing-masing fungsionalitas secara bertahap, sesuai model pengembangan yang dipilih, dalam hal ini model pengembangan perangkat lunak yang digunakan adalah *incremental model*. Sehingga proses implementasi dilakukan secara iteratif.
- 3) **Testing** merupakan salah satu fase yang wajib ada dalam pengembangan perangkat lunak untuk memastikan kualitas dan fungsionalitas yang optimal. Pengujian yang efektif dilakukan secara berkelanjutan sepanjang siklus pengembangan perangkat lunak, tidak hanya pada tahap akhir proyek. Pengujian yang dilakukan dibagi menjadi tiga, pengujian fungsional berfokus pada setiap iterasi berfungsi sesuai dengan spesifikasi. Selanjutnya, pengujian regresi yang berfokus untuk memastikan bahwa

fungsionalitas yang telah dibangun sebelumnya tidak terganggu oleh penambahan fungsional baru. Dan terakhir, pengujian integrasi yang ditujukan untuk memastikan bahwa setiap fungsionalitas baru dapat bekerja sama secara harmonis dengan fungsionalitas yang dibangun pada iterasi sebelumnya.

- 4) **Implementation** proses ini dilakukan untuk memastikan bahwa semua komponen yang dibangun pada perangkat lunak sudah diimplementasikan dan dapat dioperasikan secara bersamaan tanpa gangguan atau kesalahan. Sehingga, pada fase ini pula, tim pengembang harus memastikan bahwa fitur-fitur yang dibangun di setiap iterasi dapat saling bekerja dengan baik dengan fitur-fitur dari iterasi lain.

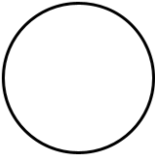
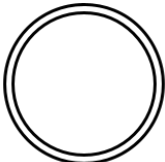
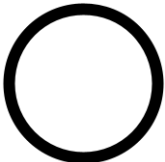

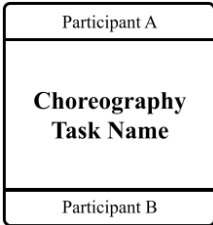
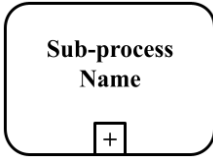
Dengan mengimplementasikan *incremental model* tim pengembang dapat melakukan perbaikan dan penyempurnaan secara berkelanjutan. Selain itu, aplikasi yang awalnya dirasa kompleks, karena dikembangkan secara iteratif, hal ini memberikan tim pengembangan pemahaman yang lebih dalam dan memudahkan pengelolaan terkait sistem secara intrinsik.

2.4. **Business Process Model and Notation (BPMN)**

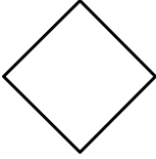



Business Process Model and Notation adalah standar khusus untuk digunakan sebagai benchmark untuk pemodelan proses bisnis yang menghasilkan notasi grafis untuk memvisualisasikan proses bisnis (Firdaus et al., 2022). *Business Process Model and Notation* (BPMN) merupakan standar global yang diakui untuk pemodelan proses bisnis. Dirancang untuk memvisualisasikan dan menganalisis proses bisnis secara jelas dan mudah dipahami, BPMN telah menjadi alat penting bagi para pemangku kepentingan dalam berbagai organisasi. BPMN menyediakan notasi grafis yang terstruktur dan konsisten untuk merepresentasikan langkah-langkah, keputusan, dan aliran data dalam suatu proses bisnis. Notasi ini memungkinkan para pelaku bisnis, analis, dan pengembang perangkat lunak untuk memahami proses bisnis dengan cara yang sama, sehingga meningkatkan

komunikasi dan kolaborasi di antara mereka. Notasi-notasi BPMN akan dijelaskan melalui tabel 2.1 dibawah ini.

Tabel 2. 1 Notasi-notasi BPMN
(Sumber : OMG, 2011)

No.	Notasi	Nama	Deskripsi
1		<i>Start Event</i>	Sesuai dengan nama notasinya, notasi ini mengindikasikan dimana sebuah rangkaian proses dimulai.
2		<i>Intermediate Event</i>	Notasi ini digunakan ketika terjadi sebuah proses diantara rangkaian proses.
3		<i>End Event</i>	Sebagaimana nama dari notasi disamping, notasi ini mengindikasikan akhir dari rangkaian proses.
4		<i>Task</i>	Sebuah tugas adalah aktivitas atomik yang dimasukkan di rangkaian proses bisnis. Sebuah tugas digunakan ketika sebuah pekerjaan dalam proses bisnis sudah tidak bisa lagi dipecah menjadi tugas yang lebih kecil lagi.
5		<i>Choreography Task</i>	Notasi ini merepresentasikan sebuah tugas yang melibatkan dua partisipan untuk menyelesaikannya.
6		<i>Sub-process</i>	Sebuah sub-proses adalah gabungan aktivitas yang dimasukkan pada sebuah proses.

Tabel 2. 1 Notasi-notasi BPMN (Lanjutan-1)
(Sumber : OMG, 2011)

No.	Notasi	Nama	Deskripsi
7		<i>Gateway</i>	Sebuah gateway tidak sama dengan notasi kondisional pada flowchart, notasi ini digunakan untuk mengontrol pemecahan dan pertemuan pada sebuah rangkaian proses bisnis.
8		<i>Sequence Flow</i>	Notasi ini mengindikasikan sebuah alur yang tidak dimulai dari <i>Intermediate Event</i> yang terhubung ke sebuah tugas / sub-proses.
9		<i>Message Flow</i>	Notasi ini digunakan untuk menunjukkan alur informasi antara dua partisipan yang bersiap untuk mengirim dan menerima informasi tersebut.
10		<i>Association</i>	Notasi ini digunakan untuk menghubungkan informasi dan artifak terkait teks anotasi dan artifak lain yang terasosiasi dengan elemen grafik lainnya.

2.5. Unified Modeling Language (UML)

Menurut buku dengan judul “*The Unified Modeling Language Reference Manual*” UML adalah tujuan umum dari bahasa pemodelan visual yang digunakan untuk menspesifikasi, memvisualisasikan, menyusun, dan mendokumentasi hal-hal yang terkait sistem perangkat lunak. UML mencatat semua keputusan dan pemahaman mengenai sistem-sistem yang wajib dibangun nantinya. Selain itu, UML digunakan untuk memahami, merancang, mengeksplorasi, mengkonfigurasi, memelihara, dan mengontrol informasi terkait sistem yang dikembangkan. Namun, UML bukan merupakan bahasa pemrograman, melainkan sebuah *tools* yang dapat

menginspirasi pembuatan program yang selanjutnya bisa dikembangkan menggunakan bahasa pemrograman lainnya.

Tujuan dari UML sendiri diantaranya adalah UML dibuat sebagai tujuan utama untuk bahasa pemodelan dimana semua pelaku pengembangan perangkat lunak bisa menggunakannya. Bertujuan untuk mencakup konsep-konsep metode ternama yang nantinya mereka gunakan sebagai bahasa pemodelan. UML tidak dimaksudkan untuk dijadikan metode pengembangan yang lengkap, sebab UML tidak memiliki langkah-langkah mendetail tentang proses pengembangan perangkat lunak. Sekali lagi ditegaskan oleh (Jacobson et al., 2021) UML mencakup konsep-konsep yang dianggap penting untuk mendukung sebuah proses iteratif yang modern berdasarkan penerapan dengan arsitektur yang kuat untuk menyelesaikan kebutuhan berdasarkan masalah-masalah dan kasus-kasus yang dialami pengguna.

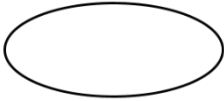
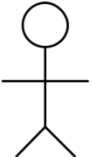
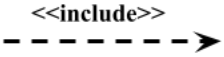
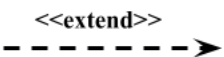
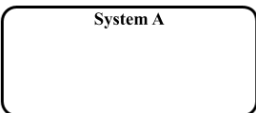
Yang terakhir, inti dari tujuan UML adalah pemodelan yang sesederhana mungkin selama masih mampu memenuhi syarat-syarat pemodelan sistem praktis secara penuh yang nantinya akan dibuat. Teknik-teknik pemodelan dari UML yang kita kenal diantaranya adalah *Use Case Diagram*, seperti yang kami jelaskan sebelumnya, kemudian terdapat *Activity Diagram*, *Class Diagram*, *Sequence Diagram*, kemudian banyak pemodelan-pemodelan lain yang kurang populer seperti *Statechart Diagram*, *Collaboration Diagram*, *Component Diagram*, *Deployment Diagram*, *Extensibility Construct*, dan pemodelan-pemodelan yang lain.

2.5.1. Use Case Diagram

Menurut (Simanullang et al., 2021) pada jurnalnya dengan judul Sistem Informasi Pemesanan Menu Makanan Pada Rm Sedep Roso Rantauprapat Berbasis Web, peneliti sempat sedikit menjelaskan tentang istilah ini. *Use Case Diagram* adalah suatu pola atau gambaran yang menunjukkan kelakuan atau kebiasaan sistem. Sedangkan (Setiawansyah et al., 2022) menjelaskan bahwa *Use Case Diagram* adalah sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di

dalam sebuah sistem. Tabel 2.2 berikut adalah notasi-notasi yang ada pada *use case diagram*.

Tabel 2. 2 Notasi *Use Case Diagram*
(Sumber : OMG, 2011)




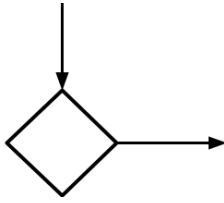
No.	Notasi	Nama	Deskripsi
1		<i>Use Cases</i>	Merepresentasikan fungsionalitas sebuah sistem, terkadang juga tujuan akhir dari aktor. Simbol ini selalu berada di dalam <i>Boundary Box</i> .
2		<i>Actors</i>	Aktor yang berinteraksi dengan sistem biasanya memicu <i>use case</i> . Simbol ini selalu berada di luar <i>Boundary Box</i> .
3	 	<i>Association</i>	<p>Arah panah digunakan untuk menandakan sebuah hubungan antara aktor dan <i>use case</i> atau antara dua <i>use case</i>.</p> <p>Panah dengan anotasi <<extend>> menandakan bahwa sebuah <i>use case</i> mungkin mengadopsi perilaku dari <i>use case</i> lain.</p> <p>Panah dengan anotasi <<include>> menandakan bahwa sebuah <i>use case</i> menggunakan fungsionalitas <i>use case</i> lain.</p>
4		<i>Boundary Box</i>	Simbol ini mengindikasikan batas lingkup sistem.

2.5.2. *Activity Diagram*

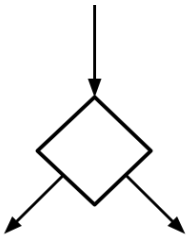
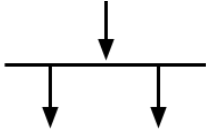
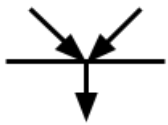

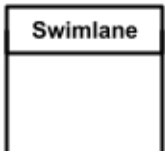

(Jacobson et al., 2021) menjelaskan *activity diagram* sebagai perwujudan khusus dari kondisi mesin yang ditujukan untuk memodel komputasi dan alur kerja sistem. Kondisi yang digambarkan pada *activity graph* mewakili kondisi eksekusi

komputasi yang dilakukan, bukan kondisi suatu objek secara spesifik. Umumnya, *activity graph* berasumsi bahwa komputasi yang terjadi tidak dipengaruhi oleh kejadian eksternal. *Activity diagram* biasanya berisi percabangan, lebih sering lagi percabangan kendali yang terbagi dua, dimana selanjutnya percabangan tersebut berjalan atau diproses bersamaan. Alur yang diproses secara bersamaan ini merepresentasikan aktivitas-aktivitas yang dapat dikerjakan oleh objek-objek atau orang-orang berbeda secara bersamaan pada sebuah organisasi. Seringkali, kejadian yang terjadi bersamaan ini muncul dari adanya agregasi, dimana objek memiliki proses yang harus dieksekusi secara bersamaan sendiri. Aktivitas yang terjadi secara bersamaan ini sebetulnya dapat dieksekusi secara bersamaan atau satu persatu. Tabel 2.3 dibawah ini menjelaskan arti dari setiap notasi untuk *activity diagram*.


Tabel 2. 3 Notasi-notasi *Activity Diagram*
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
1		Initial state	Sebuah rangkaian proses hanya membutuhkan satu kondisi awal kecuali kita menggambarkan aktivitas yang terjadi didalam aktivitas.
2		Action atau Activity State	Sebuah aktivitas merepresentasikan eksekusi dari sebuah aksi atau objek oleh objek.
3		Action Flow atau Control Flow	Alur aksi atau alur kontrol ini biasa digunakan untuk menunjukkan transisi dari satu kondisi aktivitas ke kondisi aktivitas lain
4		Decision Node dan Branching	Ketika kita akan membuat keputusan sebelum melanjutkan alur kontrol, kita bisa gunakan simpul keputusan.

Tabel 2. 3 Notasi-notasi *Activity Diagram* (Lanjutan-1)
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
5		<i>Guard</i>	Sebuah pelindung merujuk pada sebuah pernyataan yang ditulis disamping arah panah simpul keputusan. Pelindung membantu kita mengetahui batasan dan kondisi-kondisi yang menentukan alur sebuah proses
6		<i>Fork</i>	Simpul garpu digunakan untuk menggambarkan dua aktivitas yang terjadi secara bersamaan. Aktivitas yang berada dibawah simpul ini akan di eksekusi secara bersamaan.
7		<i>Join</i>	Simpul gabung digunakan untuk melebur dua aktivitas yang sebelumnya diproses secara bersamaan menjadi satu proses utuh kembali.
8		<i>Merge atau Merge Event</i>	Penggabungan aktivitas ini dapat dilakukan apabila kita ingin menggabungkan dua atau lebih aktivitas yang sebelumnya tidak diproses secara bersamaan, namun harus diselesaikan sebelum aktivitas selanjutnya dapat dieksekusi.
9		<i>Swimlanes</i>	<i>Swimlane</i> digunakan untuk mengelompokkan aktivitas-aktivitas yang dieksekusi oleh objek atau orang yang sama.
10		<i>Time Event</i>	Notasi ini merujuk pada aktivitas yang membutuhkan waktu lebih lama untuk diselesaikan.

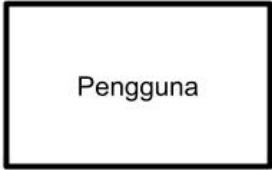

Tabel 2. 3 Notasi-notasi *Activity Diagram* (Lanjutan-2)
(Sumber: GeeksforGeeks, 2024)

No.	Notasi	Nama	Deskripsi
11		<i>Final State</i> atau <i>End State</i>	Kondisi akhir digunakan untuk menandakan akhir dari rangkaian proses.

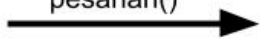

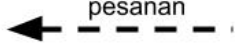
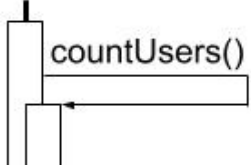
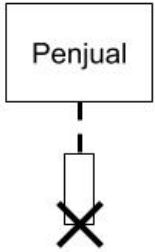
2.5.3. *Sequence Diagram*

Melalui buku manual UML yang diterbitkan oleh lembaga (OMG, 2011) dijelaskan bahwa *sequence diagram* digunakan untuk menampilkan sekumpulan pesan-pesan, atau dalam konteks pengembangan aplikasi pesan-pesan yang dimaksud adalah data, yang disusun dalam urutan waktu. *Sequence diagram* dapat menunjukkan sebuah skenario, atau yang lebih tepat lagi, menunjukkan jejak-jejak data pada suatu kegiatan transaksi. Dalam pengertian yang lain, sebuah *sequence diagram* dapat digunakan untuk menunjukkan urutan perilaku-perilaku pada suatu *use case*. Tabel 2.4 dibawah ini adalah notasi-notasi untuk *sequence diagram*.

Tabel 2. 4 Notasi-notasi *Sequence Diagram*
(Sumber: Letaw L., 2024)

No.	Notasi	Nama	Deskripsi
1		<i>Participants</i> atau <i>Objects</i>	Setiap kolom pada sebuah <i>sequence diagram</i> adalah partisipan yang terlibat. Setiap partisipan umumnya adalah objek. Nama dari partisipan yang terlibat ada di dalam kotak.
2		<i>Lifeline</i>	Garis vertikal putus-putus mewakili lama peran dari sebuah partisipan. Dimulai dari atas dimana awal peran dan dibawah untuk akhir peran. Peran partisipan berakhir apabila partisipan

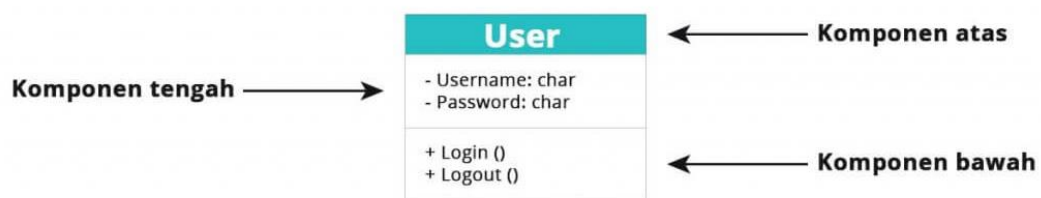
Tabel 2. 4 Notasi-notasi *Sequence Diagram*
(Sumber: Letaw L., 2024)

No.	Notasi	Nama	Deskripsi
			dihapus.
3		<i>Messages</i>	Interaksi antar partisipan digambarkan dengan arah panah. Umumnya digunakan untuk pemanggilan <i>method()</i> .
4		<i>Activation Bar</i>	Kotak yang menempuk <i>lifeline</i> menunjukkan bahwa partisipan sedang aktif.
5		<i>Return</i>	Arah panah dengan garis putus-putus mengindikasikan pengembalian nilai dari <i>method()</i> yang dipanggil.
6		<i>Self-Call</i>	Sebuah <i>method()</i> yang memanggil dirinya sendiri diwakilkan dengan panah yang mengarah kembali ke partisipan.
7		<i>Deletion</i>	Akhir dari peran partisipan. Digambarkan dengan (X) pada bagian bawah <i>lifeline</i>

2.5.4. *Class Diagram*

(Oktriwina, 2021) menjelaskan *class diagram* sebagai salah satu diagram struktur statis yang dimiliki UML dimana *class diagram* menggambarkan struktur sistem dengan menunjukkan sistem *class*, atribut-atribut yang dimiliki, metode-metode yang dapat dieksekusi, dan hubungan antar objek. (Jacobson et al., 2021)

menjelaskan *class* itu sendiri adalah penjelasan dari konsep wewenang aplikasi atau solusi dari aplikasi. *Class* dapat dideskripsikan dari berbagai tingkatan presisi dan seberapa konkrit kita ingin mendeskripsikannya. Pada awal tahapan pengembangan perangkat lunak, *class diagram* seringkali berisi aspek-aspek logika dari masalah yang ingin diatasi. Sedangkan pada akhir tahapan pengembangan, *class diagram* dapat diisi keputusan-keputusan desain dan detail-detail implementasi di dalamnya.



Gambar 2. 2 Notasi *Class Diagram*
(Sumber : Setiawan R., 2021)

Dari gambar 2.3 diatas, dapat kita ketahui bahwa komponen atas berisikan nama *class*. Setiap *class* pasti memiliki nama yang berbeda-beda, sebutan lain untuk nama ini adalah *simple name* (nama sederhana). Komponen tengah berisikan atribut dari *class*, komponen ini digunakan untuk menjelaskan kualitas dari suatu kelas. Atribut ini dapat menjelaskan dapat ditulis lebih detail, dengan cara memasukan tipe nilai. Sedangkan komponen bawah menyertakan operasi yang ditampilkan dalam bentuk daftar. Operasi ini dapat menggambarkan bagaimana suatu *class* dapat berinteraksi dengan data.

2.6. Pengujian Perangkat Lunak

Pengujian perangkat lunak pada era modern saat ini mendapat berbagai macam interpretasi, beberapa perbedaan yang mempengaruhi diantaranya adalah kebutuhan, tujuan, dan metodologi yang digunakan. Beberapa ahli dan praktisi ada yang menganggap pengujian perangkat lunak sebagai tahapan penting dalam menjamin kualitas dan keamanan perangkat lunak sebelum sampai di tangan pengguna. Sedangkan di pihak lain beranggapan bahwa, pengujian dibutuhkan untuk menjamin bahwa aplikasi yang sudah dikembangkan mampu beradaptasi

dengan perubahan. Banyak pendekatan-pendekatan yang saat ini mulai bermunculan, seperti pengujian otomatis, pengujian berbasis AI, atau pengujian konvensional seperti pengujian langsung bersama dengan pengguna.

(Dhaifullah, I. R., et al, 2022) menjelaskan bahwa proses pengujian perangkat lunak dimaksudkan untuk menemukan kesalahan aplikasi pada saat penggunaan. Salah satu contoh dari proses pengujian perangkat lunak yang dimaksud ialah, memberikan kesempatan kepada kerabat atau rekan tim pengembang untuk mencoba aplikasi. Atau proses pengujian yang dilakukan oleh sebuah tim penguji eksternal juga termasuk kedalam salah satu contoh proses pengujian perangkat lunak. Dijelaskan selanjutnya bahwa masing-masing teknik atau metode pengujian memiliki perbedaan dan persamaan, sehingga, memilih teknik pengujian yang tepat akan memberikan hasil terbaik untuk perangkat lunak yang dikembangkan.

2.7. *Black Box Testing*

Black Box Testing menguji elemen-elemen penting dalam sistem tanpa menghiraukan logika atau *source code* dari sistem yang diuji. (Adriko, S.A., 2024) *Black box testing* adalah teknik pengujian dengan merancang data pengujian yang didasarkan pada kinerja sistem yang tujuannya untuk menguji apakah sistem yang dibangun sudah beroperasi dengan benar sesuai yang diharapkan. Selanjutnya, (Setiawan, R., 2021) menjelaskan *black box testing* sebagai berikut. *Black box testing* atau dapat disebut juga *Behavioral Testing* adalah pengujian yang dilakukan untuk mengamati hasil input dan output dari perangkat lunak tanpa mengetahui struktur kode dari perangkat lunak. Pengujian ini dilakukan di akhir pembuatan perangkat lunak untuk mengetahui apakah perangkat lunak dapat berfungsi dengan baik. Untuk melakukan pengujian, penguji tidak harus memiliki kemampuan menulis kode program. Pengujian ini dapat dilakukan oleh siapa saja. Pengujian *black box* dapat dicontohkan sebagai berikut.

Tingkat pengujian dalam *black box testing* menurut (Singh, 2023) dibagi menjadi 6 kategori, namun, demi kesederhanaan pengujian sistem, kami akan

memilih 3 kategori saja dimulai dari tertinggi ke yang terendah, *Major*, fungsi dengan tingkat pengujian ini sering kali merupakan masalah signifikan yang memengaruhi fungsi inti perangkat lunak, namun tidak menonaktifkannya sepenuhnya. *Minor*, biasanya dianggap gangguan yang tidak memengaruhi fungsionalitas, namun merupakan gangguan kecil yang muncul di dalam sistem. Sedangkan *Trivial*, tingkat pengujian yang lebih kecil *Minor*, biasanya hanya masalah tampilan saja dengan sedikit atau tanpa dampak pada fungsionalitas. Mereka berprioritas rendah dan umumnya hanya diperbaiki untuk meningkatkan pengalaman pengguna atau untuk mempertahankan penampilan yang baik.

Contoh 1: Masukkan *username* dan *password* dengan benar. Respon yang diharapkan adalah pengguna berhasil masuk ke aplikasi Instagram dan sistem mengarahkan pengguna ke halaman profil.

Contoh 2: Pengguna mengunggah foto berukuran besar ke aplikasi Instagram. Respon yang diharapkan adalah sistem menampilkan pesan kesalahan untuk memberi tahu pengguna bahwa ukuran foto terlalu besar.

2.8. *Software Quality Assurance*

Jaminan Kualitas Perangkat Lunak dapat diartikan sebagai serangkaian proses, layanan, atau kegiatan yang bertujuan untuk membangun keyakinan akan produk atau dalam hal ini perangkat lunak yang dikembangkan. Proses penjaminan kualitas perangkat lunak meliputi evaluasi kontrol internal yang memastikan bahwa transaksi baik data maupun komersial antar pihak dapat diandalkan. Dimana tujuan utama dari proses penjaminan kualitas perangkat lunak selain mengurangi risiko, adalah meningkatkan kontrol internal dan meningkatkan kualitas produk atau jasa yang ditawarkan melalui perangkat lunak yang dihasilkan.

Jaminan kualitas perangkat lunak adalah aktivitas yang meliputi keseluruhan siklus pengembangan perangkat lunak, bahkan dalam kasus manajemen proyek. (Bhanushali, A. 2023) Proses penjaminan perangkat lunak bertujuan untuk memastikan bahwa kualitas kerja pada masing-masing tahapan pengembangan perangkat lunak, mulai dari proses perencanaan hingga implementasi memberikan

kualitas yang diinginkan. Pentingnya dilakukan penjaminan perangkat lunak didasari akan kerugian dalam hal material, seperti yang dikemukakan oleh penelitian yang dilakukan di Amerika Serikat yang memperkirakan kegagalan perangkat lunak menyebabkan kerugian sekitar \$59,5 miliar setiap tahunnya.

Studi menunjukkan bahwa sebagian besar *error* atau kesalahan yang muncul pada sistem atau perangkat lunak yang dirancang baru dapat dikenali ketika sudah terlambat. Maksudnya, ketika perangkat lunak sudah digunakan secara luas, kesalahan ini dapat dikurangi apabila metode, perangkat, infrastruktur dan praktisi penjaminan perangkat lunak melakukan pengujian yang lebih baik. Namun, semakin tahun, kompleksitas perangkat lunak dan tuntutan akan waktu pengembangan yang semakin pendek semakin membuka kesempatan bagi *bug* untuk tidak terdeteksi. Maka, solusi dari masalah tersebut adalah melakukan pengujian perangkat lunak secara menyeluruh.

2.9. ISO 25010:2023

Lembaga (ISO, 2011) menjelaskan ISO 25010:2023 sebagai dokumen ini mendefinisikan model kualitas produk, yang berlaku untuk produk ICT (*Information and Communication Technology*) dan produk perangkat lunak. Model kualitas produk terdiri dari sembilan karakteristik (yang dibagi lagi menjadi sub-sub karakteristik) yang berhubungan dengan sifat kualitas produk. Karakteristik dan subkarakteristik memberikan model acuan kualitas produk yang akan ditentukan, diukur dan dievaluasi. Tabel 2.5 dibawah ini menjelaskan tentang Karakteristik dan sub-sub karakteristik dari ISO 25010:2023.

Tabel 2. 5 Karakteristik ISO 25010:2023
(Sumber : ISO, 2023)

No	Karakteristik	Sub Karakteristik
1	<i>Functional Suitability</i>	<i>Functional Completeness</i>
		<i>Functional Correctness</i>
		<i>Functional Appropriateness</i>

Tabel 2. 5 Karakteristik ISO 25010:2023 (Lanjutan-1)
(Sumber : ISO, 2023)

No	Karakteristik	Sub Karakteristik
2	<i>Performance Efficiency</i>	<i>Time Behavior</i>
		<i>Resource Utilization</i>
		<i>Capacity</i>
3	<i>Compatibility</i>	<i>Co-Existence</i>
		<i>Interoperability</i>
4	<i>Interaction Capability</i>	<i>Appropriate Recognizability</i>
		<i>Learnability</i>
		<i>Operability</i>
		<i>User Error Protection</i>
		<i>User Engagement</i>
		<i>Inclusivity</i>
		<i>User Assistance</i>
		<i>Self-Descriptiveness</i>
5	<i>Reliability</i>	<i>Faultlessness</i>
		<i>Availability</i>
		<i>Fault Tolerance</i>
		<i>Recoverability</i>
6	<i>Security</i>	<i>Confidentiality</i>
		<i>Integrity</i>
		<i>Non-Repudiation</i>
		<i>Accountability</i>
		<i>Authenticity</i>
		<i>Resistance</i>
7	<i>Maintainability</i>	<i>Modularity</i>
		<i>Reusability</i>
		<i>Analysability</i>
		<i>Modifiability</i>
		<i>Testability</i>
8	<i>Flexibility</i>	<i>Adaptability</i>
		<i>Scalability</i>
		<i>Installability</i>
		<i>Replaceability</i>

Tabel 2. 5 Karakteristik ISO 25010:2023 (Lanjutan-2)
(Sumber : ISO, 2023)

No	Karakteristik	Sub Karakteristik
9	<i>Safety</i>	<i>Operational Constraint</i>
		<i>Risk Identification</i>
		<i>Fail Safe</i>
		<i>Hazard Warning</i>
		<i>Safe Integration</i>

2.10. Skala Likert

(Adriko, S. A., 2024) menjelaskan bahwa skala Likert, sering diaplikasikan untuk mengukur penilaian, pandangan, sikap, persepsi seseorang atau kelompok tentang suatu isu atau fenomena. Dengan demikian, pengujian sistem menggunakan ISO 25010 sebelumnya, dapat diukur untuk setiap pertanyaan atau pernyataan yang diajukan kepada partisipan *testing* dengan lebih objektif. Berikut adalah bobot nilai menggunakan skala Likert yang akan dijelaskan melalui tabel 2.6.

Tabel 2. 6 Bobot Penilaian menggunakan Skala Likert

Jawaban	Bobot
Sangat Setuju	5
Setuju	4
Ragu-Ragu	3
Tidak Setuju	2
Sangat Tidak Setuju	1

2.11. Interpretasi Kelayakan Sistem

Dalam penelitian yang dilakukan oleh (Bahri, N. B. Z., 2020) skala interpretasi kelayakan sistem ditujukan untuk mengetahui sejauh mana sebuah indeks atau hasil penghitungan dari evaluasi perangkat lunak yang dilakukan dicerminkan melalui kategori “Sangat Baik”, “Baik”, “Cukup”, “Kurang”, dan “Sangat Kurang”. Berikut adalah penjelasan dari masing-masing kategori dimulai dari kategori tertinggi.

Kategori pertama, yaitu “Sangat Baik” yang ada di rentang 81 – 100% menunjukkan bahwa perangkat lunak yang dirancang telah memenuhi hampir semua kriteria evaluasi dengan tingkat kesempurnaan yang sangat tinggi. Sistem dalam kategori ini dianggap sangat layak untuk digunakan tanpa membutuhkan perbaikan secara signifikan. Kategori selanjutnya, “Baik” berada di rentang 61 – 80% menunjukkan bahwa perangkat lunak yang dirancang sudah berfungsi dengan baik walaupun di beberapa bagian masih membutuhkan sedikit penyempurnaan untuk dapat dianggap optimal.

Kategori tiga, “Cukup” berada di rentang 41 – 60% mengindikasikan bahwa perangkat lunak memiliki kekurangan yang cukup signifikan, yang memerlukan perbaikan pada beberapa aspek penting agar dapat memenuhi standar. Kategori keempat “Kurang” berada di rentang 21 – 40% menunjukkan bahwa perangkat lunak memiliki banyak kelemahan yang harus diperbaiki yang membuat beberapa fungsi berperilaku tidak sebagaimana mestinya, Kategori terakhir “Sangat Kurang” menunjukkan kondisi perangkat lunak yang sangat buruk meskipun memenuhi beberapa standar apabila skor yang diperoleh berada di rentang 0 – 20%, perangkat lunak dianggap tidak layak dan sangat memerlukan perbaikan. Tabel 2.7 dibawah ini menjelaskan rentang skor dan kategori kelayakan perangkat lunak.

Tabel 2. 7 Skala Interpretasi Kelayakan Perangkat Lunak
(Sumber : Bahri, N.B.Z. et al., 2020)

No.	Skor yang diperoleh (%)	Kategori
1.	81 - 100	Sangat Baik
2.	61 – 80	Baik
3.	41 – 60	Cukup
4.	21 - 40	Kurang
5.	0 - 20	Sangat Kurang