

BAB II

LANDASAN TEORI

Pada bab ini membahas penelitian terdahulu yang relevan dan landasan teori yang mendukung topik penelitian. Bagian ini bertujuan memberikan konteks dan memperkuat analisis dengan menguraikan konsep dasar serta kerangka berpikir yang digunakan dalam interpretasi data penelitian.

2.1. Penelitian Terdahulu

Penelitian ini terhubung dengan penelitian sebelumnya, di mana subjeknya memiliki kesamaan dan perbedaan. Karena itu, penting untuk meletakkan penelitian saat ini dalam konteks yang lebih luas dan menunjukkan kontribusi dan nilai tambah yang akan dicapai.

Tabel 2.1 menunjukkan ringkasan penelitian sebelumnya, dan pembahasan dijelaskan secara naratif berdasarkan persamaan dan perbedaan dengan penelitian ini.

Tabel 2.1 Tabel Penelitian Terdahulu

No.	Penulis & Tahun	Judul Penelitian	Metode	Hasil	Analisis GAP
1	Saputra and Ratnasari, (2025)	Optimalisasi Pengelolaan Aset TI Melalui Sistem Manajemen Aset Berbasis Web	1) <i>Waterfall</i> 2) <i>Black Box Testing</i>	Penelitian ini menghasilkan sistem manajemen aset teknologi informasi berbasis web yang dibangun menggunakan framework CodeIgniter 4 dan basis data MySQL. Sistem mampu melakukan pencatatan aset, pemantauan pergerakan, pelaporan secara	Belum mengakomodasi penelusuran pengguna aktif perangkat beserta riwayat perpindahannya, serta belum membedakan pemeliharaan internal dan eksternal.

No.	Penulis & Tahun	Judul Penelitian	Metode	Hasil	Analisis GAP
				real-time, serta perhitungan penyusutan nilai aset secara otomatis.	
2	Effendri and Ma'sum, (2025)	Perancangan Aplikasi Aset Manajemen Menggunakan <i>Framework</i> Laravel di PT Dirgantara Indonesia (IAe)	1) Analisis Kebutuhan 2) Perancangan Aplikasi 3) Implementasi Fitur 4) <i>Black Box Testing</i>	Penelitian ini menghasilkan aplikasi manajemen aset berbasis web menggunakan <i>framework</i> Laravel yang mencakup pencatatan aset, pemantauan status, peminjaman aset, dan pelaporan digital untuk menggantikan proses pencatatan manual di lingkungan perusahaan.	Masih bersifat umum pada pengelolaan dan peminjaman aset, belum berfokus pada perangkat teknologi informasi, dan belum menyediakan penelusuran perpindahan pengguna maupun riwayat pemeliharaan.
3	Yodhapratama, (2024)	Pembangunan Sistem Inventaris Berbasis Web pada Gereja Kristen Jawa Samironobaru Menggunakan <i>Framework</i> Laravel	1) <i>Waterfall</i> 2) <i>Black Box Testing</i> 3) Pengujian UAT	Penelitian ini berhasil mengembangkan sistem inventaris aset berbasis web menggunakan <i>framework</i> Laravel untuk menggantikan pencatatan manual dan meningkatkan efisiensi	Terbatas pada inventarisasi aset fisik umum, belum diarahkan pada perangkat teknologi informasi, serta belum mengintegrasikan penelusuran pengguna dan riwayat

No.	Penulis & Tahun	Judul Penelitian	Metode	Hasil	Analisis GAP
				pengelolaan aset pada lingkungan gereja.	pemeliharaan sebagai bagian dari siklus hidup aset.
4	Setiyawati et al., (2024)	Requirement Engineering Sistem Manajemen Aset Gereja Berbasis Software as a Service dan Progressive Web Application	1) <i>Iterative Requirements Engineering (Loucopoulos & Karakostas)</i> 2) <i>Prototype</i>	Penelitian ini berhasil memodelkan kebutuhan sistem manajemen aset yang mencakup pendataan, operasional, pemeliharaan, dan penghapusan aset mengikuti konsep siklus hidup aset. Model kebutuhan divalidasi melalui prototype dan dinilai telah sesuai dengan kebutuhan pengguna.	Baru mencapai tahap rekayasa kebutuhan dan prototype, belum sampai implementasi sistem yang dapat diuji fungsionalitas dan usabilitasnya secara menyeluruh.
5	Hector and Panjanathan, (2024)	Predictive <i>Maintenance</i> in Industry 4.0: A Survey of Planning Models and Machine Learning Techniques	1) Systematic Literature Review 2) Analisis komparatif 90+ publikasi terkait <i>predictive Maintenance</i> dan <i>Industry 4.0</i>	Penelitian ini menyimpulkan bahwa pendokumentasian riwayat pemeliharaan berperan penting dalam mengidentifikasi pola kerusakan, mendukung perawatan perangkat, serta menjadi dasar	Bersifat kajian literatur tanpa implementasi sistem, dan berfokus pada pemeliharaan prediktif di sektor manufaktur, berbeda konteks dengan pengelolaan perangkat

No.	Penulis & Tahun	Judul Penelitian	Metode	Hasil	Analisis GAP
				pengambilan keputusan penggantian aset secara tepat.	teknologi informasi.

Berdasarkan kajian dan perbandingan terhadap penelitian-penelitian terdahulu pada Tabel 2.1 di atas, terlihat jelas bahwa penelitian yang secara spesifik membahas sistem manajemen aset TI dengan modul penugasan aset (*Asset Assignment*) beserta riwayat perpindahan perangkat secara kronologis, sekaligus dilengkapi modul pemeliharaan (*Maintenance*) yang membedakan antara penanganan *in-house* dan vendor eksternal, masih sangat terbatas. Penelitian ini hadir untuk mengisi celah tersebut dengan membangun sistem yang komprehensif menggunakan *Framework* Laravel.

2.1.1. Hasil Penelitian Saputra and Ratnasari, (2025)

Penelitian yang dilakukan oleh Saputra and Ratnasari, (2025) dengan judul “*Optimalisasi Pengelolaan Aset TI Melalui Sistem Manajemen Aset Berbasis Web*” telah membangun sistem manajemen aset TI berbasis web menggunakan *framework* CodeIgniter 4 dan MySQL untuk menggantikan pengelolaan aset yang sebelumnya dilakukan secara manual. Sistem yang dikembangkan dilengkapi modul pencatatan aset, *monitoring* pergerakan aset, pelaporan *real-time*, dan kalkulasi penyusutan metode garis lurus. Pengujian *black box* menunjukkan seluruh fungsi utama berjalan valid sesuai kebutuhan pengguna.

Penelitian ini merupakan referensi yang paling dekat secara topik, sama-sama membahas sistem manajemen aset TI berbasis web sebagai pengganti proses manual. Perbedaan utamanya, penelitian ini menggunakan Laravel bukan CodeIgniter, menambahkan modul penugasan aset (*Asset Assignment*), dan modul pemeliharaan (*Maintenance*) yang membedakan antara servis *in-house* dan vendor eksternal, modul itu tidak ada dalam penelitian tersebut.

2.1.2. Hasil Penelitian Effendri and Ma'sum, (2025)

Penelitian yang dilakukan oleh Effendri and Ma'sum, (2025) dengan judul *“Perancangan Aplikasi Aset Manajemen Menggunakan Framework Laravel di PT Dirgantara Indonesia”* telah merancang dan mengembangkan aplikasi manajemen aset berbasis web menggunakan *framework* Laravel di lingkungan industri manufaktur PT Dirgantara Indonesia. Latar belakang penelitian ini sangat relevan, perusahaan masih menerapkan sistem pencatatan aset secara manual yang mengakibatkan keterlambatan pencatatan, kesalahan data akibat faktor manusia, dan kesulitan pemantauan aset secara *real-time*. Sistem yang dihasilkan memiliki modul pencatatan aset, pemantauan status, dan pembuatan laporan digital. Seluruh pengujian *black box* dinyatakan valid.

Kesamaan dengan penelitian ini terletak pada *framework* (Laravel), permasalahan awal yaitu manual dan tidak *real-time*, dan konteks lingkungan kerja nyata sebagai objek penelitian. Perbedaannya, penelitian Effendri and Ma'sum, (2025) berfokus pada aset umum perusahaan manufaktur berskala besar, sementara penelitian ini berfokus pada perangkat TI secara spesifik dengan penugasan aset (*Asset Assignment*), dan modul pemeliharaan (*Maintenance*) yang membedakan antara servis *in-house* dan vendor eksternal.

2.1.3. Hasil Penelitian Yodhapratama, (2024)

Penelitian yang dilakukan oleh Yodhapratama, (2024) dengan judul *“Pembangunan Sistem Inventaris Berbasis Web pada Gereja Kristen Jawa Samironobaru Menggunakan Framework Laravel”* membangun sistem inventaris aset berbasis web menggunakan *framework* Laravel untuk Gereja Kristen Jawa Samironobaru yang sebelumnya masih mencatat inventaris secara manual. Kondisi awal yang ditemukan: pengelolaan aset yang masih dilakukan secara manual sering menimbulkan kesalahan pencatatan dan menyulitkan proses pemantauan aset. Sistem yang dikembangkan mampu membantu pengelolaan aset organisasi menjadi lebih terstruktur dan menunjukkan bahwa Laravel dapat digunakan untuk membangun sistem inventaris berbasis web.

Penelitian ini relevan karena memiliki kondisi awal yang serupa, yaitu pengelolaan aset yang masih dilakukan secara manual. Namun, terdapat perbedaan karena sistem Yodhapratama, (2024) berfokus pada manajemen aset fisik umum, tidak spesifik pada perangkat TI. Penelitian ini mengembangkan dengan menambahkan modul penugasan aset (*Asset Assignment*), dan modul pemeliharaan (*Maintenance*) yang terdapat riwayat pemeliharaan yang tidak ada dalam penelitian tersebut.

2.1.4. Hasil Penelitian Setiyawati et al., (2024)

Penelitian yang dilakukan oleh Setiyawati et al., (2024) dengan judul “*Requirement Engineering Sistem Manajemen Aset Gereja Berbasis Software as a Service dan Progressive Web Application*” melakukan *requirement engineering* sistem manajemen aset menggunakan model iteratif Loucopoulos dan Karakostas. Mereka mengidentifikasi bahwa permasalahan utama dalam pengelolaan aset tanpa sistem yang memadai meliputi: tidak adanya pendataan aset yang terperinci, tidak adanya *monitoring* atau jadwal perawatan berkala, dan tidak adanya pemantauan pergerakan aset. Hasil penelitian menghasilkan *framework* sistem yang mampu mengakomodasi siklus hidup aset secara menyeluruh yang berisi perencanaan, akuisisi, operasional, pemeliharaan, hingga penghapusan.

Temuan *requirement engineering* ini menjadi landasan konseptual yang kuat bagi penelitian ini. Dimensi permasalahan yang mereka identifikasi, yaitu ketiadaan pendataan detail, pemeliharaan (*Maintenance*), dan penugasan aset (*Asset Assignment*). Kondisi ini mirip dengan yang ditemukan di Departemen ICT. Perbedaannya, penelitian Setiyawati et al., (2024) hanya sampai pada tahap *prototype* dan belum mengimplementasikan sistem secara penuh. Penelitian ini mengisi gap tersebut dengan membangun implementasi sistem secara penuh menggunakan Laravel.

2.1.5. Hasil Penelitian Hector and Panjanathan, (2024)

Penelitian yang dilakukan oleh Hector and Panjanathan, (2024) dengan judul “*Predictive Maintenance in Industry 4.0: A Survey of Planning Models and Machine Learning Techniques*” melakukan survei komprehensif terhadap lebih dari

90 publikasi mengenai model perencanaan dan teknik *machine learning* untuk *predictive Maintenance* dalam konteks Industri 4.0. Temuan penelitian mereka menegaskan bahwa pendokumentasian riwayat pemeliharaan (*Maintenance*) adalah komponen fundamental dalam manajemen aset. Data riwayat pemeliharaan (*Maintenance*) yang terstruktur memungkinkan organisasi mengidentifikasi pola kerusakan yang berulang, merencanakan jadwal perawatan secara lebih efisien, dan membuat keputusan berbasis data mengenai kapan sebuah perangkat perlu diganti sebelum terjadi kegagalan total yang mengganggu operasional.

Penelitian ini memberikan landasan teoritis yang kuat untuk modul pemeliharaan (*Maintenance*) yang dikembangkan dalam penelitian ini. Tanpa rekam jejak pemeliharaan yang terstruktur, Departemen ICT hanya bisa bereaksi terhadap kerusakan, bukan mencegahnya. Modul riwayat servis yang membedakan antara penanganan in-house dan vendor eksternal akan menjawab kebutuhan dokumentasi riwayat pemeliharaan (*Maintenance*) sebagaimana direkomendasikan dari penelitian Hector and Panjanathan, (2024).

2.2. Landasan Teori

Bagian ini menyajikan teori-teori yang menjadi dasar dalam merancang sistem *Asset Lifecycle Management*. Penyusunan teori dilakukan secara berurutan, dimulai dari konsep umum hingga topik khusus sesuai fokus penelitian.

2.2.1. Departemen ICT di GMS Surabaya Barat

GMS Surabaya Barat merupakan gereja sel yang apostolik dan profetik. Gereja yang dipenuhi oleh rupa-rupa karunia Roh Kudus dan bergerak dalam amanat agung untuk menjadikan semua bangsa murid Kristus. GMS kini telah berkembang menjadi gereja yang bertumbuh cepat ke arah 300.000 jemaat. Dari kota-kota di Indonesia, sampai ke benua Asia, Australia, Eropa, dan Amerika, GMS terus berlari dalam misi yang telah Tuhan percayakan, yaitu mendirikan 1.000 gereja lokal yang kuat dengan 1.000.000 murid Kristus. Gereja ini bertempat di Pakuwon Mall Surabaya, Lantai Rooftop 2 Perumahan, Jl. Pakuwon Indah Jl. Puncak Indah Lontar II No.2, Babatan, Wiyung, Surabaya.

Sebagai gereja dengan skala menengah, tentunya ada operasional yang mendukung setiap kegiatan dan ibadah di gereja. Operasional ini bergerak sehari-hari untuk mengelola data jemaat, mengevaluasi kegiatan dan ibadah di gereja, serta membangun suasana di gereja yang nyaman untuk setiap jemaat yang datang. Dalam operasional, ada beberapa staf *fulltimer* yang menggunakan fasilitas kantor seperti laptop, komputer, printer, serta perangkat TI lainnya. Dengan fasilitas yang ada, tentunya Departemen ICT perlu sebuah jawaban untuk bisa mengatur pengelolaan perangkat TI yang dimiliki.

Dalam konteks penelitian ini, Departemen ICT di GMS Surabaya Barat berperan sebagai mitra yang menjadi objek implementasi sistem manajemen aset dengan *Asset Lifecycle Management* berbasis web yang menggunakan Laravel. Sistem yang dikembangkan bertujuan untuk mendukung manajemen aset, pengadaan (*Procurement*), penugasan aset (*Asset Assignment*), pemeliharaan perangkat (*Maintenance*), peremajaan perangkat (*Asset Refresh*), audit trail, serta pengendalian hak akses menggunakan *Role-Based Access Control* (RBAC) Dengan implementasi ini diharapkan dapat meningkatkan efektivitas pengelolaan aset TI, mempercepat pengambilan keputusan untuk peremajaan perangkat, serta mendorong transformasi digital di lingkungan gereja.

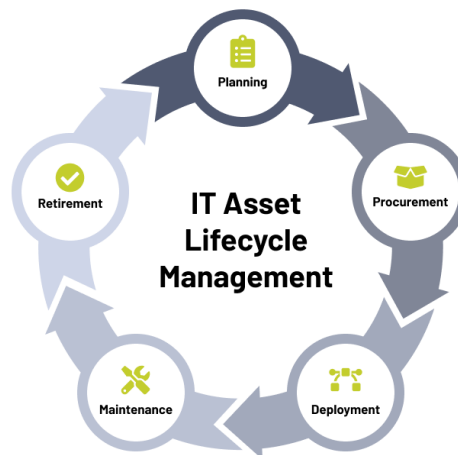
2.2.2. TI Asset Management (TIAM)

TI Asset Management (TIAM) adalah proses pengelolaan atas seluruh aset teknologi informasi yang dimiliki sebuah organisasi, mulai dari pengadaan (*Procurement*), pencatatan, penggunaan aktif, pemeliharaan (*Maintenance*), peremajaan, hingga penghapusan (*Retirement*). Pada (Team, 2022) mendefinisikan TIAM sebagai praktik *end-to-end* yang memastikan setiap aset TI digunakan dengan benar, dipelihara, ditingkatkan, dan dihapus pada waktu yang tepat, dengan memanfaatkan data finansial, kontraktual, dan inventaris untuk mendukung pengambilan keputusan strategis. Tujuan utama TIAM bukan sekadar mencatat daftar perangkat, melainkan memastikan organisasi memiliki visibilitas penuh atas semua aset yang dimiliki, sehingga tidak ada aset yang tidak terlacak, tidak digunakan secara optimal, atau berpotensi menjadi celah keamanan.

(Syaifudin et al., 2020) dalam penelitian mereka tentang efektivitas manajemen aset menegaskan bahwa efektivitas pengelolaan aset sangat dipengaruhi oleh kemampuan organisasi untuk mendokumentasikan aset secara akurat dan menelusuri perubahannya dari waktu ke waktu. Tanpa sistem yang mampu mencatat kepemilikan dan pergerakan aset secara kronologis, organisasi kehilangan kendali atas asetnya sendiri. Kondisi ini akan secara langsung berdampak pada akuntabilitas pengelolaan dan pelaporan. Saputra and Ratnasari, (2025) memperkuat temuan ini dari sisi praktis: sistem manajemen aset TI yang dibangun untuk menggantikan pencatatan manual terbukti mampu memberikan visibilitas *real-time* atas status aset dan pergerakannya, yang sebelumnya tidak mungkin dilakukan dengan Google Sheets.

Dalam penelitian ini, konsep TIAM diimplementasikan secara langsung melalui tiga modul yang saling terhubung. Modul manajemen aset bertugas mencatat seluruh data perangkat TI beserta spesifikasi, nomor perangkat, departemen, dan status terkini. Dari hasil itu akan membangun basis data tunggal yang menjadi sumber kebenaran bagi Departemen ICT. Modul penugasan aset (*Asset Assignment*) mengimplementasikan konsep *ownership* dan *tracking*, jadi setiap perangkat hanya boleh memiliki satu pengguna aktif pada satu waktu, dan setiap perpindahan perangkat dari satu pengguna ke pengguna lain dicatat secara otomatis dengan *timestamp*, nama pengguna lama dan baru, departemen, serta alasan perpindahan akan menghasilkan riwayat kronologis yang dapat ditelusuri sepenuhnya. Modul peremajaan mencatat relasi antara perangkat lama dan perangkat penggantinya, sehingga visibilitas aset tetap terjaga bahkan ketika sebuah perangkat digantikan dengan unit baru.

2.2.3. Asset Lifecycle Management



Gambar 2.1 *TI Asset Lifecycle Management*

(Sumber: <https://seibert.group/blog/en/what-is-it-asset-management/>)

Pada gambar 2.1 menjelaskan bahwa setiap perangkat TI memiliki siklus hidup yang dapat dipetakan dengan jelas dari pertama kali diadakan hingga akhirnya tidak lagi digunakan. Konsep *Asset Lifecycle Management* (ALM) adalah pendekatan terstruktur untuk mengelola setiap fase siklus hidup aset tersebut secara sistematis. (Setiyawati et al., 2024) dalam penelitian *requirement engineering* sistem manajemen aset mengidentifikasi lima fase siklus hidup aset yang harus diakomodasi oleh sebuah sistem: (1) perencanaan dan pengadaan (*procurement*), yaitu identifikasi kebutuhan dan perolehan aset baru melalui mekanisme yang terstruktur dan dapat dipertanggungjawabkan; (2) penerapan (*deployment*), yaitu pencatatan aset ke dalam inventaris dan penugasan kepada pengguna yang membutuhkan; (3) operasional dan pemeliharaan (*maintenance*), yaitu pemakaian aktif yang disertai perawatan rutin dan perbaikan ketika perangkat mengalami kerusakan; (4) penggantian (*replacement*), yaitu pergantian perangkat lama dengan unit baru atau dengan perangkat yang kondisinya lebih baik untuk memenuhi kebutuhan pengguna yang terus berkembang; dan (5) penghapusan (*retirement*), yaitu penarikan aset dari layanan aktif ketika sudah tidak dapat digunakan kembali. Tanpa pengelolaan yang terstruktur pada setiap fase ini, organisasi kehilangan kendali atas asetnya dan menghadapi risiko ketidakefisienan yang berulang.

Dari perspektif internasional, IBM (2022) menegaskan bahwa siklus hidup aset TI adalah inti dari seluruh praktik *TI Asset Management* (TIAM) yang efektif. Pemahaman menyeluruh atas setiap fase siklus hidup memungkinkan organisasi

mengoptimalkan penggunaan aset, merencanakan anggaran dengan lebih akurat, dan menghindari pembelian perangkat yang sebenarnya tidak diperlukan karena ada aset yang tidak terpantau. (Syaifudin et al., 2020) juga menemukan bahwa kemampuan organisasi menelusuri perubahan status aset sepanjang siklus hidupnya secara langsung berkontribusi pada kewajaran penyajian aset dalam laporan, yang merupakan aspek krusial bagi akuntabilitas organisasi. Lebih lanjut, fase pengadaan dalam siklus hidup aset tidak dapat berdiri sendiri sebagai proses administratif biasa, ia harus terintegrasi dengan mekanisme persetujuan yang melibatkan pihak-pihak yang berwenang secara finansial dan teknis agar setiap keputusan pengadaan dapat dipertanggungjawabkan secara institusional.

Dalam penelitian ini, konsep *Asset Lifecycle Management* diimplementasikan secara menyeluruh ke dalam tujuh modul sistem dengan pemetaan yang jelas pada setiap fasenya. Fase perencanaan dan pengadaan diwakili oleh modul pengadaan (*Procurement*) yang memungkinkan Admin ICT mengajukan kebutuhan perangkat baru dengan nomor pengajuan yang dihasilkan otomatis berformat REQ-YYYY-MM-XXXX, kemudian melewati alur persetujuan berlapis yang melibatkan Purchasing sebagai peninjau teknis dan pelaksana pembelian, serta Finance sebagai pemegang otoritas persetujuan anggaran. Ketika perangkat berhasil diadakan, sistem secara otomatis membuat entri aset baru di inventaris tanpa perlu pencatatan ulang secara manual, sehingga fase pengadaan dan pencatatan terhubung dalam satu alur yang tidak terputus. Fase *deployment* diwakili oleh modul *Asset Assignment* yang mencatat penugasan perangkat kepada pengguna aktif beserta departemennya, tanggal mulai penggunaan, dan histori seluruh perpindahan perangkat antar pengguna dari waktu ke waktu. Fase operasional dan pemeliharaan diwakili oleh modul *Maintenance* yang mencatat servis preventif maupun korektif, membedakan penanganan secara *in-house* oleh teknisi ICT dan penanganan oleh vendor eksternal, serta merekam detail keluhan, tindakan perbaikan, biaya, dan status penyelesaian. Fase penggantian diwakili oleh modul peremajaan (*Asset Refresh*) yang mencatat pergantian perangkat lama dengan perangkat pengganti beserta relasi otomatis antara keduanya, sehingga dapat diketahui bahwa suatu perangkat merupakan hasil peremajaan dari perangkat sebelumnya maupun sebaliknya. Fase penghapusan diwakili oleh perubahan status perangkat menjadi "Dihapus" yang

dicatat beserta tanggalnya sebagai penanda resmi bahwa perangkat telah ditarik dari layanan aktif. Seluruh fase tersebut terhubung melalui satu entitas tunggal, yaitu data aset perangkat TI, sehingga sistem memungkinkan Departemen ICT GMS Surabaya Barat menelusuri riwayat lengkap setiap perangkat dari hari pertama diajukan untuk diadakan hingga hari terakhir digunakan dalam satu platform yang terintegrasi.

2.2.4. *Audit Trail*

Audit trail adalah catatan kronologis terstruktur yang mendokumentasikan setiap aktivitas, perubahan, atau transaksi yang terjadi dalam sebuah sistem informasi, mencakup informasi tentang siapa yang melakukan tindakan, tindakan apa yang dilakukan, kapan dilakukan, dan data apa yang berubah. Paranona et al., (2024) menemukan bahwa tanpa sistem yang mampu merekam jejak aktivitas pengguna secara otomatis, peluang terjadinya penyimpangan dalam pengelolaan data tetap terbuka karena tidak ada mekanisme untuk mendeteksi siapa yang mengubah data apa dan kapan data itu diubah. Ini bukan sekadar masalah teknis, namun ini adalah masalah akuntabilitas organisasi. *Audit trail* mengubah sistem informasi dari sekadar alat pencatatan menjadi sebuah instrumen tata kelola yang dapat dipertanggungjawabkan.

Pada penelitian Syaifudin et al., (2020) menegaskan bahwa *traceability* adalah kemampuan untuk menelusuri kembali perubahan yang terjadi pada data aset sepanjang waktu. Ini adalah salah satu faktor utama efektivitas manajemen aset. Tanpa *traceability* yang baik, organisasi tidak dapat membuktikan bahwa laporan asetnya akurat, tidak dapat mendeteksi perubahan data yang tidak sah, dan tidak dapat mempertanggungjawabkan keberadaan atau kondisi aset kepada pihak yang berkepentingan. Effendri and Ma'sum, (2025) menambahkan bahwa sistem manajemen aset yang efektif harus mampu mencatat setiap perubahan status aset secara otomatis, sehingga riwayat aset dapat diverifikasi kapan saja oleh pihak yang berwenang.

Dalam penelitian ini, konsep *audit trail* akan diimplementasikan pada tiga titik kritis sistem. Pertama, pada modul *asset assignment* akan membuat setiap tindakan penugasan atau perpindahan perangkat secara otomatis membuat *record*

baru di tabel *asset_assignment*. Tidak ada data lama yang ditimpa, sehingga seluruh riwayat perpindahan perangkat dari pengguna pertama hingga pengguna saat ini dapat ditelusuri secara kronologis kapan saja. Kedua, pada modul *Maintenance* akan membuat setiap pencatatan servis, pembaruan status, dan penyelesaian perbaikan tersimpan sebagai *record* permanen di tabel *maintenance_records* beserta *timestamp* dan identitas pengguna ICT yang mencatat. Tidak ada data servis yang dapat diubah atau dihapus tanpa meninggalkan jejak. Ketiga, pada modul manajemen pengguna akan membuat seluruh aktivitas *login*, perubahan data, dan akses ke modul sensitif dicatat dalam log aktivitas sistem, sehingga Admin ICT dapat mengetahui siapa yang mengakses atau mengubah data apa dan kapan mengubahnya.

2.2.5. Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) adalah mekanisme keamanan sistem informasi yang membatasi akses pengguna terhadap modul dan data berdasarkan peran (*role*) yang ditetapkan kepada mereka. Prinsip dasarnya sederhana namun sangat efektif, yaitu dimulai memberikan izin akses secara individual ke setiap pengguna, kemudian izin dikumpulkan dalam sebuah '*role*' dan pengguna diberi hak akses yang sesuai dengan tanggung jawab pekerjaannya. Pada penelitian Sahyudi and Susanto, (2025) dalam analisis implementasi RBAC pada sistem informasi berbasis web menegaskan bahwa RBAC telah menjadi pendekatan utama dalam meningkatkan keamanan data di berbagai sistem informasi modern, karena memungkinkan pengelolaan hak akses yang terpusat, konsisten, dan mudah dipelihara. Prasetya and Manongga, (2024) menambahkan bahwa RBAC sangat efektif untuk sistem dengan banyak pengguna yang memiliki tanggung jawab berbeda-beda, karena perubahan hak akses dapat dilakukan cukup dengan mengubah definisi '*role*' tanpa harus memperbarui pengaturan setiap pengguna secara individual.

Dalam konteks keamanan sistem informasi berbasis web, RBAC memberikan lapisan perlindungan yang penting terhadap ancaman dari dalam (*insider threat*) yaitu situasi di mana pengguna yang memiliki akses sah menyalahgunakan akses tersebut untuk mengubah atau menghapus data yang bukan tanggung jawabnya.

Paranona et al., (2024) menemukan bahwa celah akses tanpa pembatasan berbasis peran (*role*) menjadi salah satu faktor yang membuat pengawasan data aset bergantung sepenuhnya pada individu, bukan pada mekanisme sistem. Ini merupakan kondisi yang rentan terhadap penyimpangan. Sistem yang menerapkan RBAC secara konsisten mampu memastikan bahwa setiap pengguna hanya dapat melihat dan memodifikasi data yang memang menjadi domain tanggung jawabnya.

2.2.6. Pemeliharaan Perangkat TI (*Maintenance*)

Pemeliharaan perangkat TI adalah serangkaian kegiatan yang dilakukan untuk memastikan perangkat tetap berfungsi optimal dan meminimalkan risiko kegagalan yang dapat mengganggu operasional organisasi. Dalam praktiknya, *maintenance* dibedakan menjadi dua jenis utama. *Preventive Maintenance* adalah perawatan rutin terjadwal yang dilakukan secara proaktif sebelum kerusakan terjadi, seperti pembersihan komponen internal, pembaruan sistem operasi dan *driver*, pengecekan performa *hardware*, serta penggantian komponen yang mendekati akhir masa pakainya. *Corrective Maintenance* adalah perbaikan reaktif yang dilakukan setelah perangkat mengalami gangguan atau kerusakan nyata, seperti penggantian komponen yang rusak, instalasi ulang sistem operasi, atau perbaikan akibat kerusakan fisik. Hector and Panjanathan, (2024) dalam survei komprehensif mereka menegaskan bahwa kedua jenis *Maintenance* ini bukan pilihan yang harus dipilih salah satu, melainkan pelengkap satu sama lain. *Preventive Maintenance* mengurangi frekuensi *corrective Maintenance*, sementara data dari *corrective Maintenance* memberikan informasi untuk menjadwalkan *preventive Maintenance* yang lebih tepat sasaran.

Saputra and Ratnasari, (2025) menemukan bahwa tanpa sistem yang terdokumentasi, *maintenance* perangkat TI di sebuah perusahaan sepenuhnya bergantung pada ingatan teknisi, tidak ada data riwayat yang dapat digunakan untuk menganalisis pola kerusakan atau merencanakan jadwal perawatan yang efisien. Di penelitian Setiyawati et al., (2024) mengidentifikasi hal yang sama, yaitu ketiadaan *monitoring* dan jadwal perawatan berkala adalah salah satu permasalahan utama dalam organisasi yang mengelola aset tanpa sistem yang memadai. Hector and Panjanathan, (2024) menegaskan dari sudut pandang akademis internasional bahwa

data riwayat *maintenance* yang terstruktur adalah komponen fundamental dalam manajemen kesehatan aset jangka panjang. Tanpa data riwayat *maintenance*, organisasi hanya bisa bereaksi terhadap kerusakan, bukan mencegahnya.

Dalam penelitian ini, kedua jenis *Maintenance* diimplementasikan dalam satu modul terpadu dengan modul pembeda yang spesifik. Untuk setiap catatan servis, Staf *Maintenance* memilih jenis *Maintenance* (*preventive* atau *corrective*), memilih jenis penanganan (*in-house* oleh teknisi ICT internal atau vendor eksternal melalui service center), mengisi tanggal masuk dan perkiraan selesai, mendeskripsikan gejala kerusakan, mencatat tindakan yang dilakukan, dan mencatat biaya servis. Perbedaan antara *in-house* dan vendor eksternal ini memberikan nilai tambah yang signifikan, seperti Admin ICT dapat mengetahui berapa banyak servis yang diselesaikan secara internal dengan berapa banyak yang harus dialihkan ke vendor, serta berapa total biaya yang dikeluarkan per periode. Ini adalah informasi yang sangat berguna untuk perencanaan anggaran dan evaluasi kapasitas tim ICT.

2.2.7. *Framework Laravel*



Gambar 2.2 *Laravel Logo*

(Sumber: <https://id.wikipedia.org/wiki/Laravel>)

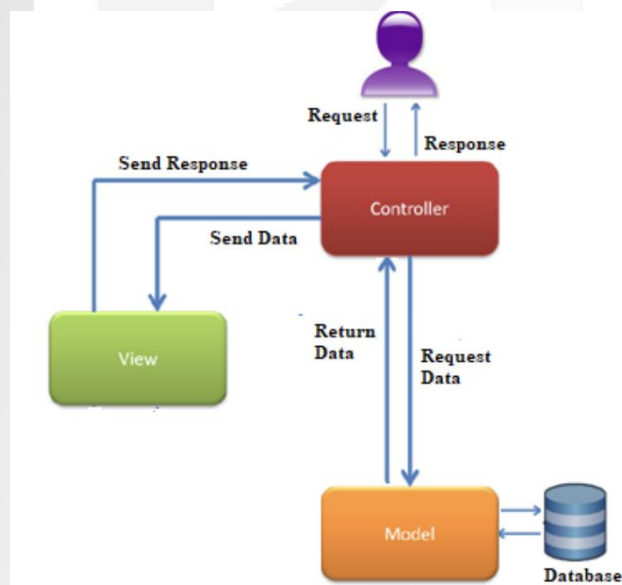
Laravel adalah kerangka kerja (*framework*) untuk membangun aplikasi web berbasis bahasa pemrograman PHP yang mengikuti arsitektur *Model-View-Controller (MVC)*. *Framework* ini membantu dalam pembangunan aplikasi web menggunakan struktur yang terorganisir, dilengkapi dengan sistem pengelolaan ketergantungan antarpaket (*package*) melalui Composer, serta menyediakan berbagai utilitas bawaan yang mempercepat pengembangan Honggara, (2020).

Beberapa modul unggulan Laravel yang digunakan dalam penelitian ini antara lain *Eloquent ORM* untuk interaksi dengan basis data secara elegan tanpa

harus menulis *query SQL* secara manual, Blade sebagai *template engine* yang memisahkan logika PHP dari tampilan HTML, serta sistem *routing* dan *middleware* yang memudahkan pengelolaan alur permintaan dan keamanan aplikasi. Dalam penelitian Nugroho and Prasetyo, (2024), menuliskan bahwa Laravel mampu meningkatkan efisiensi penyampaian data sekaligus memberikan keamanan yang andal dalam melindungi data dari ancaman eksternal.

Pada penelitian ini, Laravel dipilih sebagai *framework* utama karena ekosistemnya yang matang, dokumentasi yang lengkap di laravel.com, serta dukungan komunitas yang luas membuat pengembangan dapat dilakukan jangka panjang. Semua modul sistem, mulai dari pencatatan aset hingga laporan dan manajemen pengguna, akan dibangun dengan fondasi Laravel.

2.2.8. Model-View-Controller (MVC)



Gambar 2.3 Desain Arsitektur MVC

(Sumber: <https://eu-opensci.org/index.php/ejece/article/view/19448>)

Pada gambar 2.3 menjelaskan *Model-View-Controller(MVC)* adalah pola arsitektur dalam pengembangan perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama yang berbeda tanggung jawabnya. *Model* bertugas mengelola data dan logika bisnis termasuk membaca, menyimpan, mengubah, dan menghapus data dari basis data. *View* bertugas menampilkan informasi kepada pengguna dalam bentuk antarmuka yang bisa dilihat dan diinteraksikan. *Controller*

bertugas menjadi penghubung antara *Model* dan *View* yang menerima input dari pengguna, memprosesnya, dan menentukan *View* mana yang harus ditampilkan (Nguyen et al., 2022).

Kenapa arsitektur ini penting? Karena dengan pemisahan yang jelas antara tiga komponen tersebut, pengembang dapat bekerja pada satu bagian tanpa perlu mengubah bagian lain. Ini membuat kode lebih bersih, lebih mudah diuji, dan lebih mudah dikembangkan, terutama ketika modul bertambah. Dalam konteks Laravel, pola arsitektur *MVC* diimplementasikan secara *native*: setiap tabel basis data memiliki *file Model*, setiap halaman memiliki *file Blade (View)*, dan setiap alur sistem diatur oleh *Controller* yang spesifik.

Dalam penelitian ini, pola arsitektur *MVC* diterapkan pada seluruh tujuh modul sistem. Misalnya, modul *asset assignment* memiliki *AssetModel* untuk mengelola data perangkat, *AssignmentController* untuk memproses logika perpindahan pengguna, dan *assignment.blade.php* sebagai tampilan antarmuka yang dilihat oleh Admin ICT.

2.2.9. XAMPP



Gambar 2.4 XAMPP Logo

(Sumber: <https://www.apachefriends.org/>)

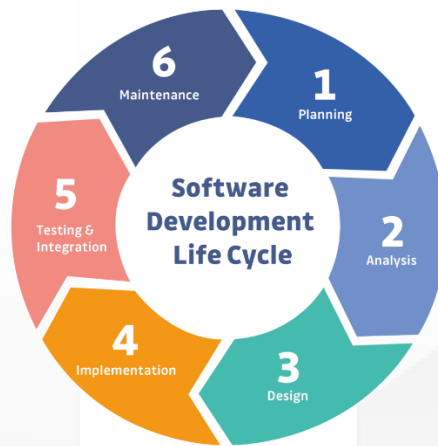
Dalam proses pengembangan sebuah sistem informasi berbasis web, dibutuhkan sebuah lingkungan lokal (localhost) yang memungkinkan pengembang membangun, menjalankan, dan menguji aplikasinya di komputer sendiri sebelum sistem tersebut dipindahkan ke server produksi yang sesungguhnya. XAMPP adalah paket perangkat lunak *open-source* yang menjawab kebutuhan tersebut. Nama XAMPP sendiri merupakan akronim dari komponen-komponen utama yang terkandung di dalamnya: X (*cross-platform*, artinya dapat berjalan di Windows, Linux, maupun macOS), A untuk Apache sebagai *web server*, M untuk MySQL atau MariaDB sebagai sistem manajemen basis data, P untuk PHP sebagai bahasa pemrograman *server-side*, dan P terakhir untuk Perl sebagai bahasa skrip

pendukung. Dengan menginstal satu paket XAMPP, pengembang langsung memiliki seluruh ekosistem yang dibutuhkan untuk menjalankan aplikasi web berbasis PHP dan MySQL tanpa harus mengonfigurasi masing-masing komponen secara terpisah.

Syafril and Wahyudin, (2025) dalam penelitian mereka membangun SIMASPAM (Sistem Informasi Manajemen Aset PAM JAYA) sebuah sistem yang konteksnya sangat relevan dengan penelitian ini karena sama-sama merupakan sistem informasi manajemen aset TI pada divisi teknologi informasi sebuah perusahaan yang sebelumnya mengandalkan Google Sheets manual secara eksplisit menyatakan bahwa sistem tersebut diimplementasikan secara lokal menggunakan XAMPP selama proses pengembangan berlangsung.

Dalam penelitian ini, XAMPP digunakan sebagai lingkungan pengembangan lokal (*local development environment*) selama seluruh proses pembangunan sistem berlangsung. Secara konkret, peran (*role*) masing-masing komponen XAMPP adalah sebagai berikut. Apache bertugas sebagai web server lokal yang memproses setiap permintaan HTTP dari browser dan meneruskannya ke Laravel sehingga pengembang dapat membuka dan menguji sistem melalui browser di alamat localhost persis seperti mengakses website sungguhan. MySQL dalam XAMPP berfungsi sebagai basis data server lokal tempat seluruh tabel sistem termasuk tabel *assets*, *asset_assignments*, *maintenance_records*, dan *users* dibuat, diisi data percobaan, dan diuji integritasnya selama fase pengembangan. PHP dalam XAMPP menjadi runtime yang menjalankan seluruh *framework* Laravel beserta logika tujuh modul sistem. Setelah seluruh modul selesai dibangun dan lolos pengujian black box serta SUS, sistem kemudian dapat dipindahkan ke server produksi dengan konfigurasi yang sudah terverifikasi selama pengembangan di XAMPP.

2.2.10. Software Development Life Cycle (SDLC) Waterfall



Gambar 2.5 *Software Development Life Cycle*

(Sumber: <https://it.telkomuniversity.ac.id/mengenal-apa-itu-software-development-life-cycle-dan-metode-nya/>)

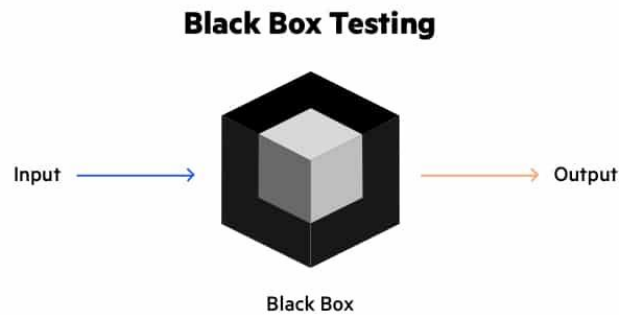
Pada gambar 2.5 menjelaskan bahwa *Software Development Life Cycle* (SDLC) adalah kerangka kerja terstruktur yang mendefinisikan tahapan-tahapan dalam proses pengembangan perangkat lunak dari awal hingga akhir. Di antara berbagai model SDLC yang ada, metode *waterfall* adalah model yang paling klasik dan sekuensial. Setiap fase harus diselesaikan sepenuhnya sebelum fase berikutnya dimulai, seperti air yang mengalir dari satu undakan ke undakan berikutnya. (Rovce, 1987) memperkenalkan model ini dalam makalah seminalnya '*Managing the Development of Large Software Systems*', yang mendeskripsikan pengembangan perangkat lunak sebagai proses berurutan melalui fase: *requirements, design, implementation, verification*, dan *Maintenance*. Petersen et al., (2009) dalam studi kasus empiris mereka di Ericsson AB mengkonfirmasi bahwa *waterfall* masih menjadi model pengembangan yang paling banyak digunakan di perusahaan pengembang perangkat lunak, dan model ini paling efektif ketika kebutuhan sistem dapat didefinisikan dengan baik sejak awal. Kejelasan fase dan ketergantungan antar fase menjadikan *waterfall* sangat cocok untuk proyek seperti penelitian ini, di mana permasalahan di Departemen ICT sudah dapat diobservasi dan didokumentasikan secara langsung sebelum pengembangan dimulai.

Nugroho and Prasetyo, (2024) menggunakan metode *waterfall* dalam penelitian mereka yang membangun sistem informasi berbasis Laravel dan

menemukan bahwa metode ini sangat memudahkan proses perancangan sistem karena setiap tahapan menghasilkan *output* yang jelas dan terukur sebelum masuk ke tahap berikutnya. Yodhapratama, (2024) juga menerapkan *waterfall* dalam pembangunan sistem inventaris berbasis web menggunakan Laravel, dan hasilnya menunjukkan bahwa struktur *waterfall* yang linear membantu peneliti mengelola kompleksitas pengembangan secara lebih terkontrol. Ningsih and Nurfauziah, (2023) menjelaskan bahwa *waterfall* paling tepat diterapkan ketika ruang lingkup proyek sudah dapat dipahami dengan baik dari awal. Ini merupakan kondisi yang mirip dalam penelitian ini, karena permasalahan di Departemen ICT sudah dapat diobservasi, didokumentasikan, dan dianalisis secara langsung sebelum pengembangan dimulai.

Dalam penelitian ini, metode *waterfall* diimplementasikan dalam lima fase yang berurutan dan terdokumentasi. Fase pertama adalah perencanaan(*planning*): peneliti melakukan rencana untuk kunjungan observasi di tempat yang akan diteliti. Fase kedua adalah analisis kebutuhan(*analysis*): peneliti melakukan observasi langsung terhadap kondisi Google Sheets ICT dan wawancara terstruktur dengan Kepala Departemen ICT untuk mengidentifikasi seluruh kebutuhan fungsional dan non-fungsional sistem. Fase ketiga adalah perancangan sistem(*design*): berdasarkan hasil analisis, dibuat *use case diagram*, *activity diagram*, *ERD (Entity Relationship Diagram)*, dan desain antarmuka menggunakan UML. Fase keempat adalah implementasi(*implementation*): sistem dibangun menggunakan Laravel dan MySQL di atas lingkungan XAMPP, mengikuti rancangan yang telah dibuat. Fase kelima adalah pengujian(*testing & integration*): sistem diuji menggunakan *black box testing* untuk memvalidasi fungsionalitas, dilanjutkan evaluasi sistem dengan SUS bersama staf ICT. Fase keenam adalah *maintenance*: evaluasi pasca-implementasi dan dokumentasi untuk pengembangan lebih lanjut di masa mendatang.

2.2.11. Black Box Testing



Gambar 2.6 Gambaran *Black Box Testing*

(Sumber: <https://www.imperva.com/learn/application-security/black-box-testing>)

Pada gambar 2.6 menjelaskan sebelum sebuah sistem informasi diserahkan kepada pengguna dan digunakan dalam operasional nyata, sistem tersebut harus melewati tahap pengujian yang ketat untuk memastikan bahwa setiap modul bekerja sesuai dengan yang diharapkan. Salah satu metode pengujian yang paling umum dan tepat digunakan untuk sistem informasi berbasis web adalah *Black Box Testing*. Dari penelitian Sakinah et al., (2024) mendefinisikan *Black Box Testing* sebagai metode pengujian perangkat lunak yang dilakukan untuk mengevaluasi fungsionalitas sistem dari perspektif pengguna tanpa memerlukan pemahaman teknis tentang kode sumber internal aplikasi. Pengujian ini murni berfokus pada input yang diberikan dan output yang dihasilkan oleh sistem. Nama '*Black Box*' sendiri menggambarkan cara pengujian ini bekerja dengan sistem diperlakukan seperti sebuah kotak hitam yang tidak tembus pandang, di mana penguji hanya melihat apa yang masuk dan apa yang keluar, tanpa peduli proses di dalamnya.

Sakinah et al., (2024) dalam penelitian mereka yang secara spesifik menerapkan *Black Box Testing* pada aplikasi manajemen aset sebuah perusahaan BUMN menegaskan bahwa metode *Black Box Testing* sangat tepat untuk menguji fungsionalitas aplikasi dari perspektif pengguna karena tidak memerlukan pengetahuan teknis tentang kode internal sistem. Penelitian tersebut menggunakan teknik *equivalence partitioning* dalam proses pengujiannya, yaitu teknik yang membagi data input menjadi kelompok-kelompok setara dan menguji perwakilan dari setiap kelompok, baik data valid maupun data tidak valid untuk memastikan sistem merespon dengan benar di semua kondisi. Syafril and Wahyudin, (2025)

yang membangun SIMASPAM (Sistem Informasi Manajemen Aset Divisi TI) juga menggunakan *Black Box Testing* sebagai metode validasi utama, dan hasilnya menunjukkan bahwa seluruh fungsionalitas sistem berjalan sesuai dengan yang diharapkan sebelum diserahkan kepada pengguna.

Dalam penelitian ini, *Black Box Testing* diterapkan pada seluruh tujuh modul sistem untuk memvalidasi bahwa setiap modul menghasilkan output yang benar sesuai skenario yang telah didefinisikan. Skenario pengujian dirancang berdasarkan fungsionalitas yang ditetapkan pada fase analisis kebutuhan, mencakup kasus-kasus berikut: pada modul manajemen aset, pengujian meliputi penambahan perangkat baru dengan nomor perangkat yang sudah ada (harus ditolak sistem), pembaruan status perangkat, dan penghapusan data; pada modul *asset assignment*, pengujian memastikan bahwa sistem secara otomatis menutup *record* pengguna lama dan membuka *record* baru saat terjadi *reassign* perangkat; pada modul *Maintenance*, pengujian memverifikasi bahwa pemilihan jenis servis '*in-house*' dan '*vendor eksternal*' menampilkan field yang sesuai dan menyimpan data dengan benar ke tabel *maintenance_records*; pada modul laporan, pengujian memastikan ekspor PDF menghasilkan file yang sesuai dengan data yang tersimpan di basis data; serta pada modul manajemen pengguna, pengujian memvalidasi bahwa hak akses setiap peran (Admin ICT, Staf *Maintenance*) benar-benar membatasi modul yang dapat diakses. Seluruh hasil pengujian didokumentasikan dalam tabel skenario uji yang memuat kolom fungsi yang diuji, input yang diberikan, output yang diharapkan, output yang dihasilkan, dan kesimpulan (valid/tidak valid).

2.2.12. System Usability Scale (SUS)

System Usability Scale (SUS) adalah instrumen pengukuran usabilitas yang dikembangkan oleh John Brooke pada tahun 1986 di Digital Equipment Corporation, dan dipublikasikan secara luas pada tahun 1996. SUS terdiri dari sepuluh pernyataan yang direspons menggunakan skala *Likert* 1–5 (sangat tidak setuju hingga sangat setuju), dengan lima pernyataan positif dan lima pernyataan negatif yang disusun secara bergantian untuk meminimalkan bias respons. Skor SUS dihitung dengan rumus khusus yang menghasilkan nilai akhir antara 0 hingga 100. Brooke, (1996) menyebut SUS sebagai '*quick and dirty usability scale*', bukan

karena tidak valid, melainkan karena dapat diterapkan dengan sangat cepat dan efisien bahkan dengan jumlah responden yang kecil. Bangor et al., (2008) mengonfirmasi validitas SUS melalui analisis terhadap hampir sepuluh tahun data pengujian pada berbagai produk, dan menyimpulkan bahwa SUS adalah alat pengukuran usabilitas yang andal dan konsisten di berbagai konteks.

Interpretasi skor SUS menggunakan skala *adjective* yang telah distandarisasi: skor di bawah 51 dikategorikan '*Not Acceptable*'; 51–67 dikategorikan '*Poor*'; 68–80,3 dikategorikan '*Good*'; 80,3–90 dikategorikan '*Excellent*'; dan di atas 90 dikategorikan '*Best Imaginable*'. Dalam konteks evaluasi sistem informasi, skor di atas 68 umumnya dianggap sebagai ambang batas minimum penerimaan, sementara skor di atas 80 menunjukkan sistem yang benar-benar memuaskan pengguna. Suria, (2024) menggunakan SUS untuk mengevaluasi sistem informasi berbasis web dan menemukan bahwa SUS efektif mengidentifikasi area antarmuka yang perlu diperbaiki, meskipun jumlah respondennya terbatas. Penelitian di *Journal of Information Systems and Informatics* (2024) juga mengonfirmasi bahwa SUS memiliki validitas statistik yang baik bahkan dengan sampel kecil, menjadikannya sangat cocok untuk evaluasi sistem informasi dengan jumlah pengguna terbatas seperti pada penelitian ini.

Dalam penelitian ini, SUS digunakan sebagai instrumen evaluasi usabilitas pada tahap akhir pengujian, melengkapi hasil *black box testing*. Kuesioner SUS disebarikan kepada seluruh pengguna sistem di Departemen ICT yang meliputi Admin ICT dan Staf *Maintenance*, setelah mereka mencoba menggunakan sistem secara langsung dalam sesi pengujian terstruktur. Sepuluh pernyataan SUS diadaptasi agar relevan dengan konteks sistem manajemen aset TI, misalnya 'Saya merasa dapat menggunakan sistem ini dengan mudah tanpa bantuan orang lain' dan 'Saya merasa sistem ini terlalu rumit untuk digunakan sehari-hari'. Skor SUS yang dihasilkan kemudian diinterpretasikan menggunakan skala *adjective* standar Bangor et al., (2008) untuk menentukan apakah sistem yang dibangun telah mencapai tingkat usabilitas yang dapat diterima dan layak diimplementasikan secara penuh di Departemen ICT.