BAB II
KAJIAN PUSTAKA

A. Kajian Teoritis

1. Sistem Pakar

Sistem pakar atau *expert system* merupakan salah satu elemen dari teknolgi komputer yang dirancang untuk memberikan kemampuan kepada komputer agar dapat melakukan proses berpikir dan mengambil tindakan menyerupai cara kerja manusia. Melalui kemampuan tersebut, sistem pakar (*expert system*) dapat memecahkan berbagai masalah yang umumnya ditangani oleh ahli serta menghasilkan keputusan yang memiliki tingkat kesesuaian tinggi dengan pendapat yang diyakini oleh masyarakat (Diana et al., 2023). Sistem pakar adalah program komputer yang dibuat untuk menyimpan pengetahuan serta berbagai fakta yang dimiliki oleh seorang pakar. Pengetahuan tersebut kemudian dimanfaatkan oleh sistem sebagai dasar dalam memberikan solusi maupun mengambil keputusan untuk menyelesaikan masalah tertentu (Wijaya et al., 2022). Seiring dengan perkembangannya, sistem pakar mengintegrasikan pengetahuan yang berlandaskan fakta dan metode penelusuran untuk menyelesaikan berbagai permasalahan yang umumnya memerlukan keahlian khusus dalam suatu bidang. Pengembangan sistem pakar dilakukan sesuai dengan kebutuhan sehingga mampu mendukung pekerjaan manusia layaknya seorang pakar, tanpa mengambil alih atau menggantikan peran pakar (Mayatopani et al., 2022). Sistem pakar dirancang untuk membantu menyelesaikan permasalahan yang umumnya tidak dapat ditangani oleh orang

awam dan memerlukan keahlian khusus dari seorang pakar pada bidang terkait. Keberhasilan sistem pakar ditentukan oleh kemampuannya dalam menghasilkan proses pengambilan keputusan maupun keputusan akhir yang memiliki kesesuaian dengan hasil yang diberikan oleh pakar yang sebenarnya (Bangun et al., 2022).

Dari keempat kutipan tersebut, bisa disimpulkan bahwa sistem pakar merupakan jenis *software* berbasis kecerdasan buatan (*artificial intelligence*) yang dibuat untuk meniru cara berpikir, penalaran, dan mengambil keputusan dari seorang ahli di bidang tertentu. Sistem ini bekerja dengan menyimpan serta mengelola pengetahuan, fakta, dan pengalaman yang dimiliki oleh para ahli ke dalam sebuah basis atau *repository* pengetahuan yang nantinya dimanfatkan untuk memberikan solusi terhadap berbagai permasalahan. Dalam perkembangannya, sistem pakar mengintegrasikan pengetahuan berbasis fakta dengan metode penelusuran atau inferensi sehingga mampu membantu menyelesaikan masalah yang memerlukan keahlian khusus. Keberhasilan sistem ini ditentukan oleh kemampuannya dalam menghasilkan keputusan yang akurat, tepat dan sesuai dengan keputusan yang diberikan oleh pakar sebenarnya, sehingga dapat berfungsi sebagai alat bantu yang mendukung pekerjaan atau aktivitas manusia tanpa menggantikan peran pakar tersebut.

2. Pneumonia

Pneumonia merupakan kondisi peradangan yang terjadi pada parenkim paru akibat berbagai penyebab, seperti infeksi oleh organisme kecil, masuknya cairan lambung ke paru, masuknya zat dari luar tubuh, paparan hidrokarbon, bahan berlemak, maupun reaksi alergi yang berlebihan. Penyakit ini cenderung

menyerang anak-anak dan bayi karena sistem imun atau pertahanan mereka belum berkembang secara optimal (Cantika et al., 2023). Manajemen Terpadu Balita Sakit (MTBS) secara resmi mengklasifikasikan penyakit pneumonia pada balita ke dalam 3 kategori, terdiri dari pneumonia berat, pneumonia, dan batuk bukan pneumonia (Kemenkes, 2022:3). Pneumonia merupakan infeksi pada saluran pernapasan yang terjadi akibat peradangan pada *alveoli* di salah satu maupun kedua paru-paru. Kondisi tersebut dapat menimbulkan berbagai gejala, seperti batuk berdahak atau disertai nanah, demam, serta kesulitan bernapas. Penularan pneumonia dapat terjadi melalui tetesan air liur atau cairan dari orang yang terinfeksi keluar saat mereka bersin, kontak erat dengan penderita, maupun dipengaruhi oleh faktor lingkungan (Muniifah et al., 2023).

Berdasarkan ketiga kutipan tersebut, dapat ditarik kesimpulan bahwa pneumonia merupakan penyakit infeksi dan peradangan pada paru-paru, khususnya pada parenkim paru dan *alveoli*, yang dapat terjadi akibat berbagai faktor seperti organisme kecil, masuknya cairan lambung ke paru, masuknya zat dari luar tubuh, paparan hidrokarbon, bahan berlemak, maupun reaksi alergi yang berlebihan. Penyakit ini lebih mudah menyerang bayi dan balita karena sistem kekebalan tubuh mereka belum berkembang secara optimal. Pneumonia bisa menimbulkan berbagai gejala, seperti batuk, demam, dan kesulitan bernapas, serta dapat menular melalui droplet, kontak langsung dengan penderita, maupun faktor lingkungan. Dalam penanganannya, Manajemen Terpadu Balita Sakit (MTBS) mengklasifikasikan pneumonia pada balita ke dalam tiga kategori, terdiri dari pneumonia berat, pneumonia, dan batuk bukan

pneumonia, sehingga memudahkan proses diagnosis dan penentuan tindakan yang tepat dalam penanganan penyakit tersebut.

3. *Forward Chaining*

Metode *Forward Chaining* tergolong sebagai salah satu teknik mekanisme penalaran yang paling sering diterapkan di lingkungan sistem pakar. Cara kerja teknik ini diawali dengan memanfaatkan sekumpulan fakta yang telah diketahui, kemudian dilakukan proses penalaran berdasarkan sekumpulan aturan yang telah ditetapkan hingga diperoleh suatu kesimpulan (Janna & Lizar, 2026). Pada metode *Forward Chaining*, aturan umumnya dinyatakan dalam bentuk pernyataan *IF-THEN*. Apabila kondisi (*IF*) pada suatu aturan sesuai dengan fakta awal yang tersedia, maka sistem akan mengeksekusi tindakan atau menghasilkan konsekuensi (*THEN*) sebagaimana telah ditentukan dalam aturan tersebut (Pangestu et al., 2025). Agar proses inferensi dapat berjalan, sistem memerlukan basis pengetahuan yang memuat kumpulan informasi serta kaidah yang bersumber dari ahli di bidang terkait. Kumpulan informasi atau pengetahuan tersebut dimanfaatkan oleh mesin inferensi untuk menganalisis fakta yang diberikan oleh pengguna sehingga dapat menghasilkan suatu kesimpulan. Sementara itu, pengetahuan yang digunakan dalam proses diagnosis, pemberian saran pengobatan, perawatan, serta upaya pencegahan disampaikan ke dalam bentuk aturan-aturan atau kaidah produksi (Kuswanto & Dapiokta, 2022). Dalam metode *Forward Chaining* saat proses inferensi, sistem akan mencocokkan fakta yang ada di memori kerja dengan aturan yang tersimpan di basis pengetahuan. Apabila ditemukan *rule* yang sesuai, maka kesimpulan dari *rule* tersebut akan dimasukkan ke dalam memori kerja. Jika tidak ditemukan *rule*

yang sesuai, sistem akan mengeluarkan hasil *default output* atau menunjukkan bahwa tidak terdapat kesimpulan yang dapat diperoleh (Arisandy & Amalia, 2025).

Dari keempat kutipan tersebut, dapat ditarik kesimpulan bahwa metode *Forward Chaining* adalah salah satu teknik inferensi yang digunakan dalam sistem pakar dan melakukan penalaran dengan memanfaatkan fakta yang telah tersedia hingga menghasilkan kesimpulan. Metode ini menggunakan sekumpulan aturan di mana umumnya berbentuk *IF-THEN*, di mana sistem akan mencocokkan fakta yang tersedia dengan ketentuan yang terdapat dalam basis pengetahuan. Proses inferensi dilaksanakan dengan memanfaatkan kumpulan fakta dan aturan yang diperoleh dari pakar, sehingga mesin inferensi dapat menganalisis informasi yang diberikan pengguna untuk menghasilkan kesimpulan, diagnosis, maupun rekomendasi yang sesuai. Selama proses tersebut, setiap aturan (*rule*) yang memenuhi kondisi akan menghasilkan kesimpulan baru yang disimpan dalam memori kerja hingga diperoleh hasil akhir. Apabila tidak ditemukan aturan (*rule*) yang sesuai, sistem akan menampilkan hasil *default ouput* atau menunjukkan bahwa tidak terdapat kesimpulan yang dapat dihasilkan berdasarkan fakta yang tersedia.

4. *Certainty Factor* (CF)

Metode *Certainty Factor* dikategorikan sebagai salah satu cara atau teknik yang digunakan di lingkungan sistem pakar guna menentukan hasil diagnosis. Metode ini bekerja dengan memberikan nilai tingkat keyakinan pakar terhadap setiap gejala, kemudian menghitung tingkat kepastian dari hasil diagnosis yang diperoleh. Pendekatan *Certainty Factor* (CF) pertama kali dikemukakan oleh

Shortliffe dan Buchanan tahun 1975 sebagai pendekatan dalam upaya mengatasi kondisi yang tidak pasti dalam proses penalaran seorang pakar yang tidak sepenuhnya pasti (*inexact reasoning*). Dalam praktiknya, seorang ahli, seperti dokter, kerap menyampaikan hasil analisis menggunakan ungkapan yang menunjukkan tingkat keyakinan tertentu, misalnya “mungkin”, “kemungkinan besar”, atau “hampir pasti” (Santoso et al., 2022). Metode ini memiliki efektivitas tinggi dan relevan untuk diterapkan dalam pembangunan sistem pakar untuk mendeteksi penyakit di bidang kesehatan karena mampu menangani proses pengambilan keputusan yang mengandung unsur ketidakpastian (Zalukhu et al., 2023).

Menurut (Tarimakase et al., 2023) *Certainty Factor* (CF) dapat dinyatakan dengan rumusan berikut:

$$CF(H,E) = MB(H,E) - MD(H,E) \quad (1)$$

Keterangan :

CF(H,E) : Menunjukkan nilai *Certainty Factor* pada hipotesis H yang dipengaruhi oleh fakta atau *evidence* E.

MB (H,E) : *Measure of Belief* merupakan ukuran peningkatan tingkat kepercayaan terhadap hipotesis H akibat adanya fakta E, dengan rentang nilai antara 0 hingga 1.

MD (H,E) : *Measure of Disbelief* menyatakan ukuran peningkatan tingkat ketidakpercayaan terhadap hipotesis H yang dipengaruhi oleh fakta E.

H : Merepresentasikan hipotesis atau kesimpulan yang dihasilkan dengan nilai antara 0 sampai 1.

E : Menyatakan *evidence*, yaitu peristiwa atau fakta yang digunakan dalam proses penalaran.

Pada implementasi sistem pakar deteksi dini pneumonia pada balita ini akan selanjutnya dihitung dengan persamaan berikut (Pamungkas et al., 2023):

$$CF(H,E)1 = CF(H) * CF(E) \quad (2)$$

Keterangan:

CF(E) : Merupakan nilai *Certainty Factor* pada *evidence* atau bukti E yang dipengaruhi oleh *evidence* atau bukti yang ada.

CF(H) : Menunjukkan nilai *Certainty Factor* pada hipotesis H sebagai akibat dari pengaruh *evidence* e.

CF(H,E) : Menyatakan nilai *Certainty Factor* suatu hipotesis dengan asumsi bahwa *evidence* telah diketahui secara pasti, yaitu pada kondisi ketika CF(E,e) = 1.

Setelah nilai CF[H,E] dihitung, jika ada lebih dari satu gejala yang terdeteksi, nilai-nilai CF[H,E] ini akan digabungkan dengan hasil nilai CF yang sudah dihitung sebelumnya dengan menerapkan rumus, mengacu pada persamaan berikut (Andres & Purnomo, 2024):

$$CFcombine = CF(H,E)1 + CF(H,E)2 * [1-CF(H,E)1] \quad (3)$$

$$CFcombine = CFold + CF(H,E)3 * [1-CFold] \quad (4)$$

Keterangan:

CFcombine : Hasil dari penggabungan perhitungan korelasi.

CF(H,E) : Hasil dari perhitungan dari H dan E (CFgejala).

CFold : Hasil dari perhitungan CFcombine yang sebelumnya.

Sebagai langkah akhir dalam menentukan tingkat kepercayaan sistem terhadap diagnosis yang dihasilkan, nilai CFcombine yang sudah dihitung akan dikonversikan ke dalam bentuk persentase menggunakan rumus berikut:

$$CFpersentase = CFcombine * 100 \quad (5)$$

Keterangan:

CFpersentase : Hasil pembulatan dari CFcombine.

CFcombine : Hasil perhitungan penggabungan korelasi.

Dari kelima kutipan di atas, dapat disimpulkan bahwa metode *Certainty Factor* (CF) merupakan metode yang digunakan dalam sistem pakar untuk mengatasi ketidakpastian dan menentukan tingkat keyakinan terhadap suatu hasil diagnosis tertentu. Proses perhitungan diawali dengan mencari nilai CF(H,E) yang diperoleh dari hasil perkalian antara nilai *Certainty Factor* hipotesis yang diberikan oleh pakar dengan nilai *Certainty Factor evidence* atau gejala yang dipilih oleh pengguna. Apabila terdapat lebih dari satu gejala yang terdeteksi, maka nilai-nilai CF(H,E) tersebut akan digabungkan menggunakan rumus CFcombine untuk memperoleh tingkat keyakinan secara keseluruhan. Selanjutnya, hasil dari CFcombine dikonversikan ke dalam bentuk persentase sehingga menghasilkan tingkat keyakinan akhir yang dapat digunakan sebagai dasar dalam menentukan hasil diagnosis pada sistem pakar.

5. Website

Website secara umum merupakan fasilitas yang tersedia di internet dan dapat diakses baik di area lokal maupun melalui jaringan yang lebih luas. Setiap dokumen yang terdapat di dalam website disebut sebagai *web page*, sedangkan tautan (*hypertext*) memberikan kesempatan bagi pengguna untuk beralih dari

satu halaman ke halaman lain, baik dalam server yang identik ataupun pada server yang tersebar pada berbagai negara. Selain menyajikan informasi berupa tulisan, website mampu memuat ilustrasi, elemen audio, serta animasi yang bertujuan meningkatkan daya tarik bagi para pengunjung (Akbar et al., 2024). Website merupakan media yang menyajikan berbagai informasi dengan memanfaatkan internet sehingga dapat digunakan dari berbagai lokasi apabila perangkat pengguna terkoneksi ke jaringan internet (Rahmi et al., 2023). Website berupa kumpulan laman web yang berada pada satu alamat web dan menyediakan beragam konten. Informasi tersebut bisa dijangkau, ditampilkan, dan dibaca melalui layanan pencarian di internet (Iswara et al., 2021). Berdasarkan karakteristiknya, website dibagi menjadi dua jenis, yaitu website statis dan website dinamis. Website statis (*static website*) dikenal sebagai jenis situs web dengan isi yang bersifat tetap sehingga perubahan isi hanya dapat dilakukan melalui pengeditan *source code* secara manual oleh pengembang. Sementara itu, website dinamis merupakan stius web yang mampu menyajikan konten serta berubah secara otomatis dan *real-time* sesuai dengan interaksi pengguna, waktu, maupun faktor lainnya (Rianto, 2025:4-5).

Dari keempat kutipan tersebut, dapat ditarik kesimpulan bahwa website dapat diartikan sebagai media berbasis internet dan terdiri dari sekumpulan halaman web dalam satu domain yang digunakan untuk menyajikan dan menyebarkan berbagai jenis informasi kepada pengguna. Website dapat diakses dari berbagai lokasi melalui jaringan internet serta memungkinkan pengguna untuk berpindah antar halaman menggunakan tautan (*hypertext*). Selain menyajikan informasi berupa tulisan, website mampu menampilkan ilustrasi,

audio, animasi, serta berbagai elemen multimedia lainnya untuk meningkatkan daya tarik serta kemudahan dalam penyampaian konten. Ditinjau dari karakteristiknya, website terbagi menjadi website statis serta website dinamis, di mana website statis memiliki konten yang cenderung tidak berubah, sedangkan website dinamis mampu menampilkan konten yang dapat berubah secara otomatis sesuai dengan interaksi pengguna maupun kondisi tertentu secara *real-time.*

6. Laragon

Laragon merupakan aplikasi yang digunakan untuk mengembangkan program atau aplikasi pada server lokal (*localhost*) di komputer. Salah satu keunggulan Laragon adalah kemudahan pengoperasiannya melalui antarmuka grafis (*Graphical User Interface* atau GUI) serta fitur yang lebih lengkap dibandingkan dengan aplikasi sejenis. Selain mendukung pengembangan aplikasi menggunakan Git dan pengelolaan basis data melalui phpMyAdmin, Laragon juga kompatibel dengan berbagai teknologi, seperti Node.js/MongoDB, Python/Django/Flask/PostgreSQL, Ruby, Java, dan Go. Aplikasi ini menawarkan kompatibilitas yang luas dengan berbagai sistem operasi, antara lain Windows, macOS, dan Linux (Sitorus et al., 2024:1). Laragon merupakan lingkungan pengembangan lokal yang dirancang agar ringan sekaligus memiliki kemampuan yang andal dalam mendukung proses pengembangan aplikasi. Perangkat lunak ini memungkinkan pengembang menjalankan server web dengan memanfaatkan teknologi seperti PHP, MySQL, dan Apache. Selain itu, keberadaan fitur "*auto virtual host*" dan "*portable environment*' memberikan kemudahan dalam mengelola serta mengonfigurasi proyek pengembangan.

Laragon juga memiliki keunggulan berupa proses instalasi dan konfigurasi yang sederhana, disertai dukungan terhadap berbagai versi PHP yang dapat disesuaikan dengan kebutuhan masing-masing proyek (Prastyia et al., 2025). Laragon sangat tepat digunakan untuk mengembangkan dan menguji sistem secara lokal sebelum sistem tersebut diunggah ke server hosting atau produksi (Andriansyah et al., 2025).

Berdasarkan ketiga kutipan tersebut, dapat ditarik kesimpulan bahwa Laragon merupakan platform pengembangan lokal yang dimanfaatkan untuk membangun, mengoperasikan, serta mengevaluasi aplikasi pada server lokal (*localhost*). Laragon dirancang memiliki tampilan grafis yang mudah dipahami dan dioperasikan dan mendukung berbagai teknologi serta bahasa pemrograman, sehingga dapat memudahkan pengembang dalam mengelola proses pengembangan aplikasi. Selain memiliki proses instalasi dan konfigurasi yang sederhana, Laragon juga menyediakan berbagai fitur yang mendukung pengelolaan proyek secara efisien, seperti auto *virtual host*, *portable environment*, serta dukungan terhadap berbagai versi PHP dan basis data (*database*). Dengan kemampuan tersebut, Laragon menjadi pilihan yang baik untuk mengembangkan dan menguji sistem secara lokal sebelum diimplementasikan pada server *hosting* atau lingkungan produksi.

7. MySQL

MySQL merupakan *Database Management System* (DBMS) di mana memanfaatkan SQL (*Structural Query Language*) sebagai bahasa guna mengatur basis data serta sering diterapkan pada pembuatan sistem berbasis web. Berdasarkan jenis lisensinya, MySQL terbagi menjadi dua kategori,

Kategori awal merupakan *Free Software*, yang merupakan perangkat lunak yang bisa digunakan dan dipakai secara bebas bagi semua orang. Sementara itu, kategori kedua adalah *Shareware*, yakni perangkat lunak berpemilik yang penggunaannya dibatasi sesuai dengan ketentuan lisensi yang berlaku (Remawati & Wijayanto, 2021:21). MySQL merupakan program *database* server yang memiliki kemampuan untuk memproses penerimaan dan pengiriman data dengan kecepatan tinggi serta mendukung penggunaan oleh banyak pengguna (*multi-user*). Pengembangan MySQL mula-mula dipelopori oleh Michael Widenius yang merupakan pengembang basis data. Disamping itu, MySQL bersifat ringan dan mudah dioperasikan (Primadewi & Sunarni, 2022:4). MySQL dikenal sebagai salah satu sistem basis data yang sering dimanfaatkan pada proses pembangunan aplikasi berbasis website dan juga dikenal di antara basis data yang paling populer di dunia. Penggunaannya sangat umum pada sistem yang dibangun dengan *platform* web, misalnya PHP serta Apache. Tingginya popularitas MySQL di kalangan pengembang didukung sebab keandalannya dalam mengelola basis data serta ketersediaannya yang dapat dimanfaatkan secara cuma-cuma (Aziz et al., 2025).

Berdasarkan ketiga kutipan tesebut, dapat ditarik kesimpulan bahwa MySQL merupakan *Database Management System* (DBMS) yang memanfaatkan SQL (*Structural Query Language*) sebagai bahasa utama dalam pengolahan dan pengaturan basis data. MySQL sering dimanfaatkan dalam pengembangan sistem berbasis website karena memiliki kemampuan dalam memproses, melakukan penerimaan dan pengiriman data secara cepat serta mendukung penggunaan oleh sejumlah pengguna sekaligus (*multi-user*). Selain

bersifat ringan dan mudah dioperasikan, MySQL juga dikenal sebagai salah satu sistem basis data yang populer dan bagus dalam mengelola data. Ketersediaannya dalam versi yang dapat digunakan secara gratis serta kompatibilitasnya dengan berbagai teknologi pengembangan web menjadikan MySQL sebagai salah satu opsi utama bagi para pengembang saat menciptakan aplikasi yang berbasis web.

8. PHP

PHP adalah kependekan dari *Hypertext Preprocessor*. PHP awalnya diciptakan berupa proyek *open source* yang berskala kecil, lalu terus berkembang menjadi bahasa pemrogramman yang sangat terkenal dan sering dimanfaatkan untuk mengembangkan aplikasi berbasis web. Popularitasnya sangat meningkat karena terbukti sangat mudah digunakan. PHP adalah salah satu bahasa pemrograman *scripting* paling awal yang mendukung paradigma Berorentasi Objek, memiliki sintaks yang mudah dipahami, relatif sederhana dalam pengodean, serta membantu dalam pembuatan halaman web dinamis (Singh et al., 2024:1). PHP diciptakan untuk pertama kalinya oleh Rasmus Lerdorf pada tahun 1994. Bahasa pemrogramman ini merupakan perangkat lunak sumber terbuka yang memungkinkan siapa saja untuk memanfaatkannya tanpa biaya. PHP dikeluarkan dengan lisensi PHP *License*, yang memiliki perbedaan dibandingkan dengan lisensi GNU *General Public License* (GPL) yang secara umum diterapkan pada berbagai proyek sumber terbuka (Noviana, 2022). PHP merupakan bahasa pemrograman yang bersifat interpretatif, yaitu bahasa pemrograman yang menerjemahkan setiap baris *pseudocode* menjadi kode yang dapat dipahami oleh komputer pada saat program dijalankan. Selain

itu, PHP dikategorikan sebagai bahasa pemrograman *server-side programming* karena seluruh proses eksekusi program dilakukan di sisi server (Darmawan & Geni, 2023).

Dari ketiga kutipan di atas, dapat disimpulkan bahwa PHP (*Hypertext Preprocessor*) merupakan sebuah bahasa pemrograman skrip yang tersedia secara gratis dan banyak diterapkan pada pembuatan aplikasi web. PHP dirancang untuk mempermudah pembuatan halaman web dinamis karena menawarkan struktur penulisan kode yang sederhana, mudah dipelajari, dan mendukung konsep pemrograman berorientasi objek. Sebagai bahasa skrip *server-side*, seluruh proses eksekusi program PHP dilakukan di sisi server sebelum hasilnya dikirimkan kepada pengguna. Selain itu, sifatnya yang gratis (*open source*), fleksibel, dan mudah digunakan menjadikan PHP sebagai salah satu bahasa pemrograman yang banyak dikenal serta banyak digunakan untuk membangun beragam aplikasi.

9. Laravel

Laravel merupakan salah satu *framework* PHP terpopuler yang dikembangkan oleh Taylor Otwell. *Framework* web ini menggunakan PHP, yang merupakan bahasa pemrograman *open-source* dan gratis, serta dirancang guna mengembangkan aplikasi web dengan arsitektur MVC (*Model-View-Controller*) (Sinlae et al., 2024). Laravel merupakan kerangka kerja yang dirilis di bawah lisensi MIT dan memakai GitHub sebagai tempat untuk berbagi kode (Amanullah & Santoso, 2023). Laravel adalah satu-satunya *framework* yang bisa membantu meningkatkan penggunaan PHP dalam mengembangkan situs web. *Framework* ini rancang untuk mendukung pembuatan aplikasi website dan

mengikuti pola MVC, yaitu *model, view*, dan *controller*. MVC merupakan suatu cara pembuatan pendekatan dalam pengembangan *software* yang memisahkan bagian alur proses aplikasi dengan bagian tampilannya. MVC membagi sistem menjadi bagian-bagian seperti pengelolaan data, *controller*, dan antarmuka pengguna (Sweetania & Herawati, 2022).

Dari ketiga kutipan tersebut, dapat ditarik kesimpulan bahwa Laravel merupakan suatu *framework* berbasis PHP yang gratis serta bisa digunakan secara bebas, serta dibuat untuk mempermudah pembuatan aplikasi situs web. Kerangka kerja ini memakai arsitektur MVC yang membagi tugas aplikasi menjadi tiga kategori yaitu logika program, pengelolaan data, dan tampilan antarmuka pengguna. Dengan cara ini, pembuatan aplikasi menjadi lebih rapi, lebih cepat, dan lebih mudah dikelola. Selain itu, Laravel dibuat dengan lisensi MIT dan didukung oleh GitHub sebagai media kolaborasi serta pengelolaan kode, sehingga menjadikannya salah satu kerangka kerja PHP yang terkenal dan umum diterapkan dalam mengembangkan aplikasi web.

10. *Black Box Testing*

Black box testing adalah metode pengujian perangkat lunak yang berfokus pada keluaran sistem tanpa memperhatikan struktur internal maupun cara sistem beroperasi. Pendekatan tesebut menitikberatkan di pengujian fungsi serta hasil akhir sistem secara keseluruhan, sehingga penguji tidak perlu memahami secara mendalam proses atau cara kerja sistem di bagian internalnya (Saputra & Mardiati, 2025). Metode ini memiliki peran penting dalam memverifikasi bahwa perangkat lunak berjalan sesuai dengan spesifikasi yang telah ditetapkan serta mampu memberikan hasil keluaran yang tepat sesuai dengan *input* yang

dimasukkan. Proses pengujian dilakukan tanpa meninjau atau mempertimbangkan mekanisme pemrosesan kode program pada bagian internal sistem (Fauzi & Yunial, 2025:104). *Black Box Testing* adalah teknik pengujian yang diterapkan oleh pengembang maupun penguji (*tester*) untuk memverifikasi bahwa sistem sudah beroperasi selaras dengan spesifikasi serta persyaratan yang sudah ditentukan. Proses pengujian ini dijalankan tanpa memerlukan pengetahuan terkait cara kerja maupun proses internal yang terjadi di dalam sistem (Wicaksono, 2023:38).

Berdasarkan ketiga kutipan tersebut, dapat ditarik kesimpulan bahwa *Black Box Testing* adalahh teknik pengujian sistem yang menitikberatkan pada pengujian fungsi, *input*, dan *output* sistem tanpa meninjau struktur maupun proses dalam sistem yang terjadi pada aplikasi. Metode tesebut digunakan agar memverifikasi bahwa sistem telah berjalan dan sesuai berdasarkan spesifikasi dan ketentuan yang telah ditetapkan. Pada pelaksanaannya, penguji hanya mengevaluasi kesesuaian hasil yang dihasilkan sistem berdasarkan masukan yang diberikan, tanpa perlu memahami mekanisme atau kode program yang digunakan. Dengan demikian, *Black Box Testing* dapat membantu memverifikasi dan memastikan bahwa setiap fitur sistem bekerja secara optimal serta menghasilkan keluaran yang selaras dengan kebutuhan pengguna.

11. *System Usability Scale* (SUS)

System Usability Scale (SUS) merupakan instrumen yang berfungsi untuk mengevaluasi aspek (*usability*) sebuah sistem maupun aplikasi. Metode ini diperkenalkan John Brooke pada tahun 1986 serta sudah diterapkan secara luas di beragam penelitian maupun penilaian produk. SUS memuat 10 pernyataan

yang digunakan untuk menilai berbagai aspek kegunaan, dengan penilaian yang diberikan oleh responden menggunakan skala Likert lima poin. Hasil penilaian kemudian dikonversi menjadi skor keseluruhan dalam rentang 0 hingga 100 sehingga memudahkan interpretasi terhadap tingkat kegunaan sistem (Sa'adah et al., 2024). Dalam metode *System Usability Scale* (SUS), soal yang bernomor ganjil disusun dalam bentuk pernyataan positif, sedangkan soal yang bernomor genap menggunakan pernyataan negatif (Fauzi et al., 2022). Metode *System Usability Scale* (SUS) memiliki sejumlah keunggulan, antara lain proses evaluasi yang sederhana sehingga mudah dipahami oleh responden, kemampuan menghasilkan penilaian yang optimal meskipun menggunakan jumlah sampel yang terbatas, serta kemampuannya dalam membedakan apakah suatu aplikasi memiliki tingkat kegunaan yang baik atau sebaliknya (Fatmawati, 2021).

Dari ketiga kutipan di atas, dapat disimpulkan bahwa *System Usability Scale* (SUS) merupakan metode evaluasi yang digunakan untuk mengukur tingkat kegunaan (*usability*) suatu sistem atau aplikasi berdasarkan penilaian langsung dari pengguna. Metode SUS ini menggunakan 10 pernyataan yang terdiri atas pernyataan positif pada nomor ganjil dan pernyataan negatif pada nomor genap, dengan penilaian menggunakan skala Likert. Hasil dari penilaian kemudian diolah dan dikonversikan menjadi skor dalam rentang 0 hingga 100 untuk menunjukkan tingkat kegunaan sistem yang diuji. Selain memiliki proses evaluasi yang sederhana dan mudah dipahami oleh responden, metode SUS juga mampu menghasilkan penilaian yang efektif meskipun menggunakan jumlah sampel yang terbatas, sehingga dapat digunakan untuk menilai apakah suatu sistem memiliki tingkat kegunaan yang baik atau tidak.

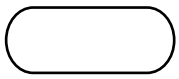
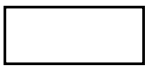
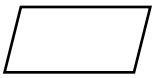
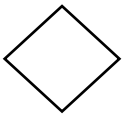
12. *Flowchart*

Diagram alir atau *flowchart* adalah gambaran visual yang dipakai untuk menjelaskan suatu proses, sistem, dan langkah kerja dengan menggunakan simbol-simbol yang terhubung satu sama lain melalui garis atau panah. Penggunaan diagram alir (*flowchart*) bertujuan untuk mempermudah pemahaman terhadap setiap tahapan dalam suatu proses serta membantu menyajikan informasi yang kompleks agar lebih mudah dipahami oleh berbagai pihak (Suwanda et al., 2025:82). Bagan alir adalah metode analisis yang berbentuk visual dan digunakan untuk menampilkan berbagai elemen dalam sistem informasi dengan cara yang jelas, singkat, dan rasional, sehingga dapat mempermudah pemahaman terhadap alur yang ada (Tuasamu et al., 2023). Tujuan utama penyusunan *flowchart* adalah menggambarkan urutan langkah secara sistematis sebagai solusi terhadap suatu permasalahan melalui proses manipulasi aritmetika atau logika yang dapat diterjemahkan menjadi instruksi bagi komputer (Khairunnisa et al., 2023:89).

Dari ketiga kutipan di atas, dapat disimpulkan bahwa *flowchart* atau diagram alir adalah gambaran visual yang dipakai untuk menunjukkan suatu proses, sistem, maupun urutan kerja dengan cara yang jelas dan terstruktur melalui simbol-simbol yang saling terhubung. *Flowchart* berfungsi sebagai alat bantu untuk menyajikan informasi yang kompleks agar lebih mudah dipahami, serta mempermudah proses analisis dan pemahaman terhadap alur suatu sistem. Selain itu, *flowchart* berfungsi untuk menampilkan rangkaian langkah-langkah secara teratur dalam menyelesaikan masalah berdasarkan proses logika maupun aritmatika yang dapat diterjemahkan menjadi instruksi

bagi komputer, sehingga membantu dalam perancangan dan pengembangan sistem secara lebih efektif.

Tabel 2. 1 Simbol *Flowchart*

No	Gambar	Simbol	Keterangan
1.		Terminal	Simbol yang berfungsi untuk menunjukkan awal dan akhir dari serangkaian proses yang berkaitan dengan komputer.
2.		Garis Alir	Simbol yang berfungsi untuk menghubungkan antar simbol
3.		Pemrosesan Komputer	Simbol yang berfungsi untuk menggambarkan suatu proses yang dilakukan oleh sistem komputer.
4.		*Input-Output*	Simbol yang berfungsu untuk menggambarkan operasi *input/output*
5.		Keputusan	Simbol yang berfungsi untuk menggambarkan posisi di mana keputusan harus diambil untuk menentukan tindakan selanjutnya.
6		*Input/Output* Dokumen	Simbol yang berfungsi ketika masukan berasal dari dokumen dan keluaran menuju dokumen.

Sumber: (Khairunnisa et al., 2023:90)

13. *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah alat pemodelan visual yang digunakan untuk merancang dan memodelkan sistem perangkat lunak, termasuk dalam konteks proses bisnis. UML mengintegrasikan berbagai teknik pemodelan yang telah dikembangkan sebelumnya ke dalam suatu kerangka kerja yang konsisten dan terstandarisasi (Wicaksono, 2023:21). UML menyediakan standar dalam penyusunan *blueprint* sebuah sistem yang meliputi pemodelan alur bisnis, desain kelas-kelas di bahasa pemrograman tertentu, penyusunan skema basis

data, serta elemen-elemen yang diperlukan dalam proses pengembangan perangkat lunak. Di samping itu, UML juga adalah salah satu alat pemodelan yang dimanfaatkan untuk mendukung desain dan pengembangan perangkat lunak yang berfokus pada objek (Saputra et al., 2023:140). UML juga menawarkan standar untuk menyusun dokumentasi desain sistem secara teratur dan konsisten, yang dapat memperbaiki komunikasi serta keselarasan pemahaman di antara beragam pihak yang terlibat dalam proyek pengembangan perangkat lunak. UML juga memiliki berbagai tipe diagram, seperti *class diagram*, *use case diagram*, *activity diagram*, dan *sequence diagram*, di mana masing-masing menggambarkan sudut pandang yang berbeda dalam menampilkan sistem yang sedang dirancang (Purnama, 2024).

Berdasarkan ketiga kutipan diatas, bisa disimpulkan bahwa *Unified Modeling Language* (UML) merupakan bahasa visual yang dipakai untuk merancang, memodelkan, dan mendokumentasikan sistem perangkat lunak dengan cara yang terstruktur dan terstandarisasi. UML berfungsi sebagai alat bantu dalam pengembangan perangkat lunak dengan menyediakan *blueprint* yang mencakup pemodelan proses bisnis, perancangan kelas, basis data, serta berbagai komponen sistem lainnya. Selain mendukung pengembangan perangkat lunak berbasis objek, UML juga berperan dalam meningkatkan komunikasi dan kesamaan pemahaman di antara pihak-pihak yang terlibat dalam proses pengembangan sistem. Untuk menggambarkan berbagai aspek sistem, UML menawarkan berbagai jenis diagram, seperti *use case diagram*, *class diagram*, *activity diagram*, dan *sequence diagram*, yang dapat digunakan sesuai kebutuhan dalam perancangan dan dokumentasi sistem.

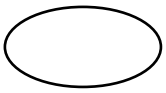
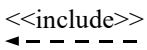
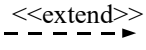
a. *Use Case Diagram*

Use Case Diagram adalah suatu cara pemodelan yang berfungsi untuk menunjukkan interaksi antara sistem yang sedang dibangun dengan pihak-pihak yang terlibat. Dalam penyusunannya, sistem direpresentasikan di dalam sebuah batas sistem (*system boundary*), sedangkan sedikitnya satu aktor ditempatkan di luar batas tersebut untuk menunjukkan hubungan dengan sistem (Hendri et al., 2022). *Use Case Diagram* berfungsi sebagai alat untuk menggambarkan, memvisualisasikan, dan mendokumentasikan perilaku suatu sistem. Diagram ini dipakai untuk menampilkan berbagai tindakan yang dapat dilakukan oleh aktor terhadap sistem, serta menjelaskan kebutuhan atau persyaratan sistem dari perspektif aktor dan juga pengguna sistem (Rura & Ardiansyah, 2023). *Diagram Use Case* berfungsi untuk menggambarkan hubungan yang berlangsung antara pengguna (aktor) dan sistem yang sedang diteliti. Dengan menggunakan diagram ini, kemampuan inti dari sistem dapat dikenali dan sekaligus memberikan gambaran mengenai cara pengguna berinteraksi dengan sistem tersebut (Wicaksono, 2023:50). Tujuan penggunaan *Use Case Diagram* adalah untuk menggambarkan kemampuan atau fungsi yang dapat dijalankan oleh sistem serta aktivitas yang dapat dilakukan oleh pengguna dalam berinteraksi dengan sistem tersebut (Syam & Erdisna, 2022).

Dari empat kutipan di atas, dapat disimpulkan bahwa *Diagram Use Case* adalah salah satu jenis diagram dalam UML yang berfungsi untuk memodelkan serta menggambarkan hubungan antara aktor (pengguna) dengan sistem yang sedang dibuat. Diagram ini berfungsi sebagai alat bantu

untuk mendeskripsikan, memvisualisasikan, dan mendokumentasikan kebutuhan serta fungsionalitas sistem berdasarkan sudut pandang pengguna. Dalam penyusunannya, sistem digambarkan di dalam batas sistem (*system boundary*), sedangkan aktor ditempatkan di luar batas tersebut untuk menunjukkan hubungan dan interaksi yang terjadi. Dengan demikian, *Diagram Use Case* dapat berfungsi untuk mengidentifikasi fungsi-fungsi utama sistem dan memberikan gambaran yang jelas mengenai aktivitas yang dapat dilakukan pengguna saat berinteraksi dengan sistem.

Tabel 2. 2 Simbol *Use Case Diagram*

No	Simbol	Nama	Keterangan
1.		*Actor/Role*	Simbol yang memnunjukkan peran sistem lain, orang, atau alat ketika berhubungan dalam *use case.*
2.		*Use Case*	Simbol yang menunjukkan abstraksi dan cara berinteraksi antara *actor* dengan sistem.
3.		*Generalization*	Simbol yang menggambarkan interaksi aturan-aturan dengan elemen lain.
4.		*Include*	Simbol yang digunakan untuk memasukkan sebuah *use case* ke dalam *use case* lainnya.
5.		*Extend*	Simbol yang digunakan untuk memperluas *use case* agar dapat memasukkan perilaku yang tidak wajib.

Sumber : (Nurelasari, 2021:19)

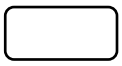
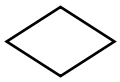
b. *Activity Diagram*

Diagram Aktivitas adalah salah satu tipe diagram dalam UML yang berfungsi untuk menggambarkan proses atau alur aktivitas yang terjadi di dalam sebuah sistem bisnis. Diagram ini mendukung pengembang perangkat lunak dalam merancang dan memodelkan alur kerja serta proses bisnis dengan cara yang lebih terorganisir dan sistematis (Pranoto et al., 2024). Diagram Aktivitas berfungsi untuk menggambarkan rangkaian langkah dalam suatu proses bisnis serta memperlihatkan hubungan antara berbagai aktivitas yang ada di dalamnya. Dengan diagram ini, setiap tahapan dalam proses bisnis dapat diidentifikasi beserta hubungan yang terjadi antara satu aktivitas dengan aktivitas lainnya (Wicaksono, 2023:51). *Activity Diagram* juga dimanfaatkan untuk mendefinisikan serta mengelompokkan alur aktivitas yang terdapat pada suatu sistem. Diagram ini tersusun atas berbagai komponen dengan simbol tertentu yang saling dihubungkan menggunakan tanda panah untuk menunjukkan urutan aktivitas yang berlangsung secara berkesinambungan, mulai dari proses awal hingga proses akhir (Trivaika & Senubekti, 2022).

Dari ketiga kutipan di atas, dapat disimpulkan bahwa *Activity Diagram* adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang berfungsi untuk memodelkan serta memvisualisasikan alur dari aktivitas, proses bisnis, atau bahkan alur kerja yang ada dalam suatu sistem. Diagram ini berperan sebagai alat bantu yang menggambarkan rangkaian aktivitas dengan cara yang teratur dan sistematis, sehingga membuat pemahaman terhadap proses yang berlangsung dalam sistem menjadi lebih

mudah. *Activity Diagram* terdiri atas beragam simbol yang saling terhubung melalui panah untuk menunjukkan relasi dan urutan antaraktivitas, mulai dari langkah awal hingga langkah akhir. Dengan demikian, *Activity Diagram* dapat dimanfaatkan untuk mengidentifikasi setiap tahap dalam proses serta saling keterkaitan antaraktivitas dalam sistem dengan cara yang jelas dan terstruktur.

Tabel 2. 3 Simbol *Activity Diagram*

No	Simbol	Nama	Keterangan
1.		Status Awal / *Initial Node*	Simbol yang menandakan permulaan dari berbagai tindakan atau aktivitas.
2.		*Activity*	Simbol yang menggambarkan sekumpulan tindakan (*action*).
3.		*Decision Node*	Simbol yang memiliki tujuan untuk menandakan situasi pengujian sehingga aliran kontrol atau objek hanya dapat melalui satu jalur saja. Dilambangkan dengan kriteria yang menentukan untuk melanjutkan ke jalur tertentu.
4.		Status Akhir / *Final Node*	Simbol yang berfungsi untuk menghentikan seluruh aliran kontrol dan objek dalam suatu aktivitas.
5.		*Control Flow*	Simbol yang digunakan Simbol yang berfungsi untuk menunjukkan urutan eksekusi.
6.		*Swimlane*	Simbol yang digunakan untuk membagi diagram aktivitas ke dalam baris dan kolom agar kegiatan tertentu bisa diberikan kepada individu atau objek yang

No	Simbol	Nama	Keterangan
			bertanggung jawab untuk melaksnakannya.

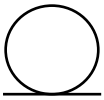
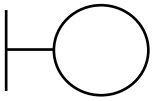
Sumber : (Nurelasari, 2021:21)

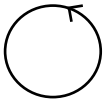
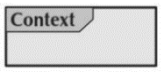
c. *Sequence Diagram*

Diagram Sequence adalah salah satu tipe diagram dalam UML yang digunakan untuk menggambarkan interaksi antara objek dalam suatu sistem berdasarkan waktu. Diagram ini sering digunakan selama proses pengembangan *software* untuk menggambarkan aliran pesan yang terjadi antara objek-objek yang berinteraksi. Dengan demikian, *Sequence Diagram* dapat membantu dalam memahami perilaku sistem beserta interaksi yang berlangsung di dalamnya (Mulyati et al., 2024). *Sequence Diagram* dapat dianalogikan sebagai sebuah skenario yang memperlihatkan urutan interaksi antarelemen dalam sistem ketika menjalankan suatu tugas. Penyusunan *Sequence Diagram* menjadi lebih rinci dan akurat dengan memanfaatkan informasi yang diperoleh dari *Use Case Diagram* dan *Activity Diagram*. Kedua diagram tersebut berfungsi sebagai acuan dalam menyusun *Sequence Diagram* sehingga interaksi antara objek dan sistem dapat digambarkan secara lebih mendetail (Khairunnisa & Voutama, 2024). *Sequence Diagram* digunakan untuk memvisualisasikan proses interaksi antarobjek di dalam sistem beserta mekanisme pengiriman dan penerimaan pesan yang berlangsung berdasarkan urutan waktu. Diagram ini berperan dalam membantu mengidentifikasi alur komunikasi serta bentuk kolaborasi yang terjadi antara objek-objek di dalam sistem (Wicaksono, 2023:52).

Dari ketiga kutipan di atas, dapat disimpulkan bahwa *Diagram Sequence* adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang berfungsi untuk memodelkan serta memvisualisasikan interaksi antara objek di dalam suatu sistem berdasarkan urutan waktu. Diagram ini menunjukkan alur pesan yang dikirimkan dan diterima oleh objek-objek yang berinteraksi dalam suatu proses atau tugas tertentu. Dalam proses pembuatannya, *Diagram Sequence* dapat menggunakan informasi dari *Diagram Use Case* dan Diagram Aktivitas agar dapat memberikan gambaran interaksi yang lebih detail dan tepat. Oleh karena itu, *Sequence Diagram* dapat membantu dalam memahami perilaku sistem, mengidentifikasi alur komunikasi, serta menggambarkan kolaborasi yang terjadi antarobjek dalam sistem secara jelas dan terstruktur. Selain itu, diagram ini juga menjadi acuan dalam proses perancangan sistem agar alur interaksi antarobjek dapat diimplementasikan sesuai dengan kebutuhan fungsional sistem.

Tabel 2.4 Simbol *Sequence Diagram*

No	Gambar	Nama	Keterangan
1.		*Actor*	Simbol yang menggambarkan entitas, baik berupa manusia maupun sistem, yang terlibat dalam suatu alur dengan mengirimkan dan/atau menerima pesan.
2.		*Entity Class*	Simbol yang mewakili *Class.*
3.		*Boundary Class*	Simbol yang menggambarkan tampilan dari sistem

No	Gambar	Nama	Keterangan
4.		*Control Class*	Simbol yang menggambarkan *controller* dari sistem.
5.		*Message*	Simbol yang dipakai untuk menyampaikan informasi dari satu objek ke objek lainnya.
6.		*Life Line*	Simbol yang menunjukkan adanya kehidupan pada suatu objek dalam urutan tertentu.
7.		*Execution Occurrence*	Simbol yang menggambarkan momen ketika sebuah objek melakukan aktivitas pengiriman atau penerimaan pesan.
8.		*Frame*	Simbol yang menunjukkan konteks *sequence diagram*.

Sumber : (Nurelasari, 2021:30)

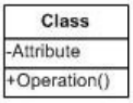
d. *Class Diagram*

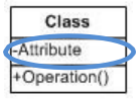
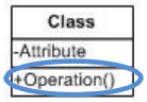
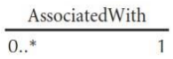
Class Diagram merupakan sebuah representasi visual yang menampilkan struktur sistem berdasarkan pada kelas, atribut, dan relasi yang terhubung antar kelas. Diagram ini berfungsi untuk membantu dalam mengidentifikasi komponen-komponen yang membentuk sistem beserta keterkaitan di antara setiap komponen tersebut (Wicaksono, 2023:51). *Class Diagram* digunakan untuk menampilkan kelas-kelas yang terdapat dalam suatu sistem beserta hubungan logis yang terbentuk di antara kelas tersebut. Diagram ini menyajikan representasi statis dari sistem atau perangkat lunak dengan menggambarkan keterkaitan yang terjadi antar komponen di dalamnya (Pertiwi et al., 2023). *Class Diagram* terdiri atas sekumpulan

interface, class, relasi, serta *collaboration* yang saling berkaitan di dalam sistem. Pada tahap analisis, diagram ini digunakan untuk mengkaji aturan dan tanggung jawab setiap entitas dalam mendukung perilaku sistem (Subariah & Eriana, 2021:158).

Berdasarkan tiga kutipan yang diatas, dapat disimpulkan bahwa *Diagram Class* adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang berfungsi untuk memvisualisasikan struktur tetap suatu sistem dengan mengacu pada kelas, atribut, metode, serta interaksi yang ada di antara kelas-kelas tersebut. Diagram ini berfungsi sebagai alat bantu untuk mengidentifikasi komponen-komponen yang membentuk sistem beserta keterkaitan yang terjadi di antara komponen tersebut. *Class Diagram* juga digunakan untuk menggambarkan *interface, class, relasi*, dan *collaboration* yang terdapat dalam sistem sehingga dapat memberikan representasi yang jelas mengenai struktur perangkat lunak. Selain itu, pada tahap analisis, diagram ini membantu dalam memahami aturan dan tanggung jawab setiap entitas yang berperan dalam mendukung perilaku serta fungsi sistem secara keseluruhan.

Tabel 2.5 Simbol *Class Diagram*

No	Gambar	Nama	Keterangan
1.		*Generalization*	Simbol yang mewakili jenis hubungan antara beberapa kelas.
2.		*Class*	Simbol ini mewakili entitas seperti individu, lokasi, atau objek tertentu yang datanya perlu direkam dan disimpan oleh sistem.

No	Gambar	Nama	Keterangan
3		*Attribute*	Simbol yang mewakili properti yang menjelaskan kondisi sebuah objek.
4		*Operation*	Simbol yang menggambarkan tindakan atau fungsi spesifik yang dapat dijalankan oleh suatu kelas.
5		*Association*	Simbol yang mewakili keterkaitan antar kelas maupun hubungan sebuah kelas dengan dirinya sendiri.

Sumber : (Santoso & Migunani, 2021:187)

B. Kajian Empiris

Penelitian yang dilakukan oleh Andres & Purnomo tahun 2024 berfokus dalam membuat sistem pakar diagnosis ISPA menggunakan metode *Certainty Factor*. Sistem tersebut dirancang untuk membantu dokter dalam melakukan pengambilan keputusan diagnosis secara otomatis. Hasil penelitian menunjukkan bahwa sistem pakar mampu menghasilkan kesimpulan awal berupa diagnosis yang sesuai berdasarkan pengetahuan dokter maupun masukan dari pengguna sistem. Sebagai pengembangan pada penelitian berikutnya, disarankan agar cakupan sistem pakar diperluas melalui penambahan berbagai fitur yang dapat meningkatkan tingkat keakuratan diagnosis.

Penelitian yang dilakukan oleh Fitri et al., pada tahun 2023 berfokus dalam pembuatan sistem pakar pakar untuk mendeteksi dini infeksi saluran pernafasan akut (ISPA) berdasarkan gejala pasien. Dalam penelitian tersebut menerapkan kombinasi metode *Forward Chaining* dan *Certainty Factor*. Penerapan kedua

metode tersebut mampu menghasilkan tingkat akurasi sistem yang berada pada rentang persentase yang tinggi.

Penelitian yang dilakukan oleh Arisandy & Amalia pada tahun 2025 berfokus dalam implementasi metode *Forward Chaining* pada sistem pakar diagnosis penyakit diare pada anak. Metode *Forward Chaining* memungkinkan sistem melakukan proses penalaran dari gejala yang diketahui hingga memperoleh diagnosis yang paling sesuai. Melalui aplikasi berbasis web tersebut mampu membantu para orang tua untuk memberikan solusi awal untuk mencegah terjadinya kembali penyakit diare pada anak tanpa harus bertemu dengan pakarnya yaitu seorang dokter.

Sementara itu, penelitian yang dilakukan oleh Kuswanto & Dapiokta pada tahun 2022 berfokus pada penerapan metode *Forward Chaining* dalam mendiagnosis penyakit pneumonia. Pengembangan sistem pakar di bidang kedokteran tersebut dirancang untuk membantu mengurangi beban kerja dokter ketika melakukan proses penegakan diagnosis penyakit. Hasil penelitian menunjukkan bahwa gejala-gejala pneumonia pada manusia dapat direpresentasikan ke dalam basis pengetahuan berbentuk kaidah *IF-THEN-ELSE*. Selain itu, metode *Forward Chaining* berhasil diimplementasikan pada mesin inferensi untuk diagnosis pneumonia beserta rekomendasi penanganannya. Proses inferensi tetap mampu menghasilkan diagnosis meskipun data gejala yang diberikan pengguna belum lengkap. Hasil yang ditampilkan mencakup jenis penyakit pneumonia beserta rekomendasi penanganan dan langkah pencegahannya.

Penelitian yang dilakukan oleh Azizah & Niska pada tahun 2024 berfokus pada pembuatan suatu sistem pakar untuk mendiagnosis penyakit paru-paru. Sistem

pakar tersebut dikembangkan dengan menggunakan metode *Certainty Factor*, dimana didalamnya terdapat data gejala serta nilai CF yang dialami. Hasil penelitian ini adalah sebuah aplikasi sistem pakar yang mampu mengenali penyakit pada paru-paru. Dengan adanya sistem pakar ini, pasien lebih mudah mengetahui jenis penyakit yang dialaminya, dan diharapkan bisa membantu masyarakat yang ingin berkonsultasi tanpa perlu bertemu dokter spesialis langsung.

Adapun Pamungkas et al., pada tahun 2023 melakukan penelitian yang bertujuan untuk merancang dan membangun sistem pakar untuk mendiagnosis penyakit lambung menggunakan metode *Certainty Factor*. Sistem dibuat dengan menggunakan metode *Rapid Application Development* (RAD). Pada tahap desain, sistem dirancang menggunakan *flowchart*, *Unified Modeling Language*, serta perancangan basis data. Sistem ini menggunkaan metode *Certainty Factor* yang digunakan untuk menentukan diagnosis, desain *database* menggunakan MySQL, serta proses pembuatan website dengan bahasa pemrograman PHP, CSS, dan JavaScript. Hasil penelitian ini menunjukkan bahwa sistem pakar mampu memberikan diagnosis yang sesuai dengan aturan yang ditentukan oleh para ahli serta metode yang digunakan. Sistem bekerja dengan baik dan bisa memberikan hasil berupa arah diagnosis berdasarkan gejala-gejala yang dialami oleh pengguna yang sedang menderita sakit lambung.

Penelitian yang dilakukan oleh Samudro et al., pada tahun 2024 bertujuan untuk membuat sistem yang bisa membantu petugas kesehatan dalam mengenali penyakit yang disebabkan oleh gigitan nyamuk secara lebih cepat dan lebih tepat menggunakan *framework* Laravel. Metode yang diterapkan adalah *Forward Chaining* dan *Certainty Factor* untuk meningkatkan ketepatan serta akurasi dalam

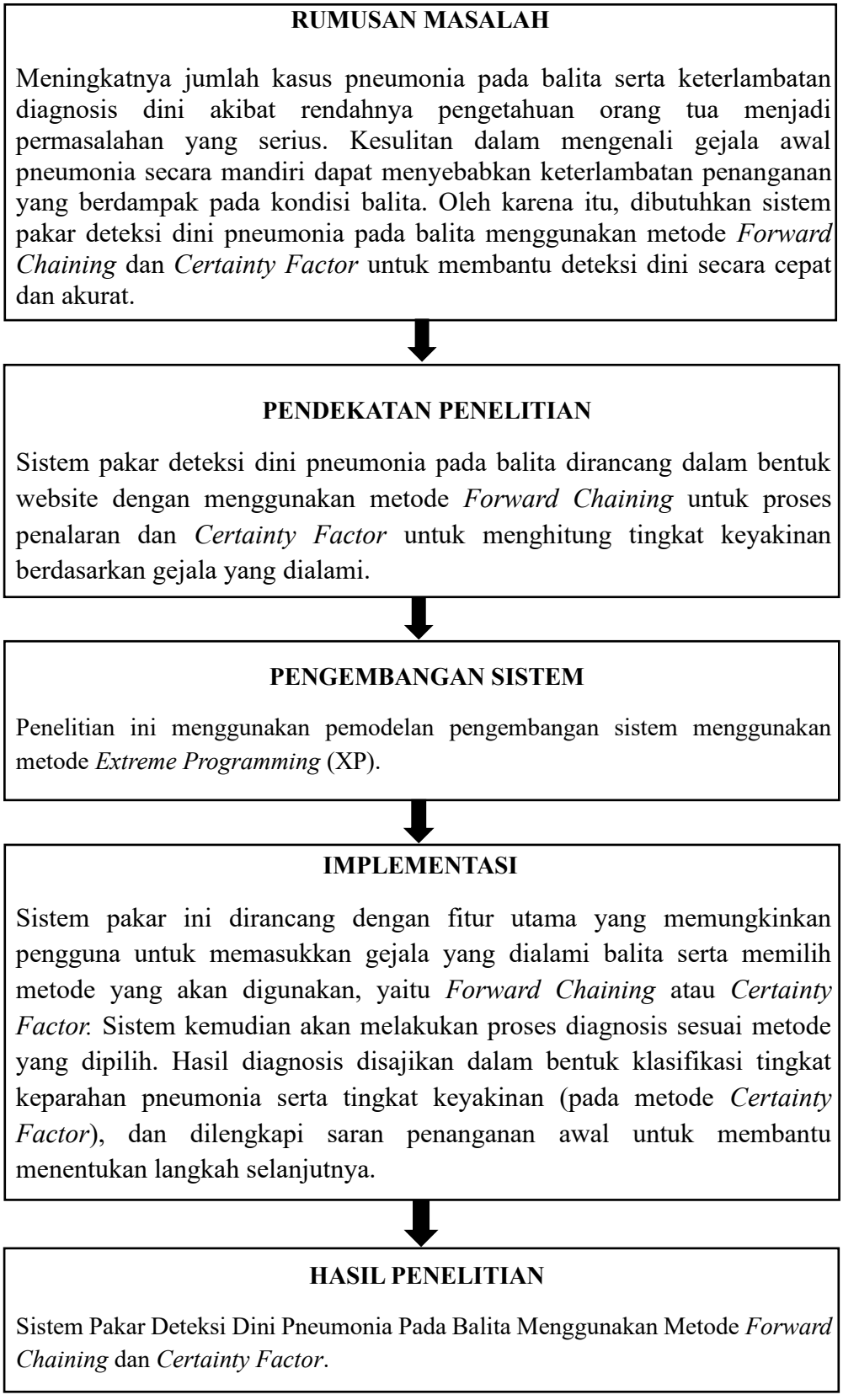
mengidentifikasi jenis penyakit berdasarkan gejala yang muncul. Penelitian ini menerapkan pendekatan pengujian *Black Box Testing* yang menunjukkan bahwa sistem ini memenuhi ekspektasi pengguna, sehingga dapat digunakan untuk identifikasi penyakit. Saran untuk penelitian selanjutnya dengan mengembangkan aplikasi ini dengan menambahkan fitur-fitur baru, meningkatkan antarmuka pengguna, dan memperluas *platform* ke perangkat seperti Android dan IOS.

Penelitian yang akan dilakukan bertujuan untuk merancang dan membangun sistem pakar deteksi dini pneumonia pada balita menggunakan metode *Forward Chaining* dan *Certainty Factor*. Sistem ini dibuat untuk mengatasi keterbatasan pada penelitian sebelumnya, dimana penelitian (Andres & Purnomo, 2024) yang hanya dapat digunakan untuk mengidentifikasi penyakit ISPA tanpa spesifik membahas pneumonia, dan penelitian (Arisandy & Amalia, 2025) yang hanya dapat mendiagnosis penyakit diare pada balita menggunakan pendekatan metode *Certainty Factor*. Selain itu untuk mengatasi keterbatasan pada penelitian yang dilakukan oleh (Kuswanto & Dapiokta, 2022) yang hanya menerapkan metode tunggal *Forward Chaining* untuk diagnosis penyakit pneumonia.

Penelitian ini mengacu pada metode yang digunakan oleh (Azizah & Niska, 2024) yaitu metode *Certainty Factor*, dan (Pamungkas et al., 2023) akan tetapi menggunakan metode pengembangan sistem *Extreme Programming* dalam pembuatan sistem pakar. Selain itu, penelitian ini juga mengacu pada penelitian (Samudro et al., 2024) yang mengkombinasikan metode *Forward Chaining* dan *Certainty Factor* dalam pembuatan sistem pakar. Dengan demikian, penelitian ini diharapkan dapat menjadi solusi sistem pakar yang dapat mendiagnosa pneumonia pada balita dengan efisien dan akurat.

C. Kerangka Berfikir

Kerangka berpikir ini dibuat untuk menunjukkan langkah-langkah dalam merancang dan membangun sistem pakar deteksi dini pneumonia pada balita berbasis website menggunakan metode *Forward Chaining* dan *Certainty Factor.* Penelitian ini dilatarbelakangi oleh tingginya kasus pneumonia pada balita serta rendahnya pengetahuan orang tua dalam mengenali gejala awal secara mandiri, sehingga diperlukan solusi teknologi yang mampu memberikan hasil diagnosis dengan mudah, cepat, dan akurat. Menggunakan pendekatan metode *Extreme Programming* yang meliputi tahap perencanaan, perancangan, pengkodean, hingga pengujian, sistem ini bekerja dengan memproses gejala yang di*inputkan* pengguna melalui penalaran maju *Forward Chaining* maupun perhitungan tingkat keyakinan *Certainty Factor*. Hasil akhir dari penelitian ini berupa klasifikasi tingkat keparahan pneumonia pada balita dalam bentuk persentase serta rekomendasi penanganan medis untuk meningkatkan kewaspadaan orang tua terhadap kondisi kesehatan balita. Berikut adalah kerangka berpikir pada penelitian ini dapat dilihat pada Gambar 2.1:



Gambar 2. 1 Gambar Kerangka Berfikir