BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Sistem Pendukung Keputusan (SPK)

Menurut Rahmansyah & Lusinia, (2021:2) pengambilan keputusan pada hakikatnya merupakan sebuah proses sistematis dalam memilih satu alternatif terbaik dari sekian banyak opsi yang tersedia untuk kemudian ditindaklanjuti sebagai sebuah metode pemecahan masalah. Proses penentuan pilihan yang terstruktur ini menjadi fondasi penting dalam penyelesaian konflik atau dilema sebelum sebuah kebijakan dieksekusi. Guna mendukung efisiensi dari aktivitas tersebut di era digital, Siregar et al., (2025:2), menyatakan bahwa Sistem Pendukung Keputusan dirancang sebagai alat bantu dalam mengurai dan menyelesaikan berbagai persoalan yang tidak terstruktur.

Secara fungsional, teknologi ini diimplementasikan sebagai instrumen penunjang efektivitas, bukan untuk mengeliminasi peran manusia selaku pembuat kebijakan. Mahendra et al., (2023:8) mengemukakan bahwa sistem tersebut mampu memformulasikan rekomendasi objektif yang bersumber dari pemrosesan data dan analisis matematis. Kendati demikian, kendali penuh serta otoritas dalam menetapkan hasil akhir tetap berada secara mutlak di tangan pihak yang berwenang.

Berdasarkan sintesis terhadap teori-teori di atas, dapat disimpulkan bahwa Sistem Pendukung Keputusan (*Decision Support System*) merupakan sebuah platform digital interaktif yang dirancang untuk memfasilitasi formulasi kebijakan melalui pemrosesan data dan pemanfaatan model analitis. Kehadiran sistem ini berperan sebagai instrumen bantu dalam menguji berbagai alternatif opsi, sehingga pembuat kebijakan mampu menetapkan solusi paling ideal secara objektif, sistematis, dan terukur. Pada akhirnya, implementasi SPK berkontribusi positif dalam mendongkrak efektivitas dan kualitas hasil akhir, tanpa mendegradasi otoritas serta tanggung jawab penuh pengambil keputusan atas kebijakan yang ditetapkan.

2. Cushion

Penggunaan produk kosmetik dekoratif seperti *foundation cream* dapat menyumbat pori-pori dan memberikan pengaruh buruk terhadap kondisi fisik kulit wajah, khususnya apabila diaplikasikan selama melakukan aktivitas fisik (Yoon et al., 2024:1885). Seiring dengan perkembangan industri kecantikan, formulasi *liquid foundation* terus mengalami inovasi hingga dikembangkan menjadi produk yang lebih praktis, ringkas, dan mudah dibawa, yaitu *cushion*. Spence & Zhang, (2024:835), memaparkan bahwa interaksi antara kulit dengan produk kosmetika tidak hanya ditentukan oleh fungsi dasarnya, melainkan juga dipengaruhi secara signifikan oleh persepsi multisensori pengguna. Pengalaman sensori tersebut mencakup penilaian visual terhadap tampilan

hasil akhir (*finish*) pada wajah, serta kenyamanan taktil dari tekstur produk saat diaplikasikan ke permukaan kulit.

Berdasarkan uraian tersebut, dapat disimpulkan bahwa cushion merupakan inovasi dari produk *foundation* yang dirancang untuk memberikan kemudahan dalam penggunaan tanpa mengurangi fungsinya sebagai kosmetik dasar. Dalam memilih produk cushion, pengguna perlu mempertimbangkan berbagai aspek, seperti kenyamanan pada kulit, tampilan *finish*, efektivitas *coverage*, serta karakteristik produk lainnya yang memengaruhi hasil penggunaan. Oleh karena itu, diperlukan parameter penilaian yang terukur agar konsumen dapat menentukan produk *cushion* yang paling sesuai dengan kebutuhan dan preferensinya.

3. Metode TOPSIS

Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) ialah sebuah instrumen pengambilan keputusan multikriteria yang menitikberatkan pada konsep interpretasi geometris. Taherdoost & Madanchian, (2024:2) menjabarkan bahwa sistem ini bekerja dengan mengukur jarak spasial, di mana opsi paling potensial diidentifikasi dari kedekatan terdekatnya dengan acuan ideal positif serta kerenggangan terjauhnya dari nilai ideal negatif. Selaras dengan hal itu, Setiawansyah (2022:55), menegaskan bahwa fungsi metode ini adalah untuk menyaring berbagai kandidat opsi guna menemukan solusi paling unggul berdasarkan sekumpulan parameter yang sudah ditetapkan terlebih dahulu. Proses

komparasi dalam sistem dijalankan dengan mengalkulasi nilai kemiripan setiap objek terhadap titik acuan terbaik maupun titik acuan terburuk.

Menurut Setiawansyah, (2022:55-56), tahapan perhitungan dalam metode TOPSIS dapat diuraikan sebagai berikut:

1. Pembentukan Matriks Keputusan

   Struktur matriks keputusan ini disusun dengan mengacu pada kombinasi nilai antara seluruh alternatif dan kriteria yang dievaluasi.

$$X = \left[x_{ij}\right]_{mxn}$$

   Dimana:

   a) x_{ij} = merepresentasikan skor evaluasi dari alternatif ke-i pada parameter kriteria j, dengan komponen i dan j bertindak sebagai penunjuk baris dan kolom matriks.
   b) m = menyatakan total seluruh alternatif yang dinilai.
   c) n = menunjukkan jumlah keseluruhan kriteria

2. Melakukan Normalisasi Matriks

   Tahapan normalisasi dijalankan guna menyegeramkan rentang nilai antar-parameter kriteria agar dapat diperbandingkan secara adil.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}$$

   Dimana:

a) r_{ij} = merepresentasikan elemen matriks yang telah dinormalisasi.

b) x_{ij} = menunjukkan performa atau skor aktual alternatif ke-i pada kriteria ke-j.

c) m = menyatakan total kuantitas alternatif yang dievaluasi.

3. Penyusunan Matriks Normalisasi Terbobot

Tahapan ini dilakukan dengan mengalikan setiap elemen nilai yang telah dinormalisasi dengan bobot preferensi dari masing-masing kriteria yang bersesuaian.

$$y_{ij} = w_j \cdot r_{ij}$$

Dimana:

a) y_{ij} = nilai terbobot

b) w_j = bobot kriteria ke-j berdasarkan tingkat kepentingan

c) r_{ij} = nilai hasil normalisasi

4. Penetapan Solusi Ideal Positif dan Negatif

Solusi ideal ditentukan dengan mempertimbangkan karakteristik dari masing-masing indikator, apakah kriteria tersebut dikategorikan sebagai keuntungan (*benefit*) atau pengeluaran (*cost*).

Solusi ideal positif : $A^+ = (y_1^+, y_2^+, \dots, y_n^+)$

Solusi ideal negatif : $A^- = (y_1^-, y_2^-, \dots, y_n^-)$

Dengan:

a) y_j^+ = min (y_{ij}) untuk kriteria biaya (cost)

b) y_j^+ = max (y_{ij}) untuk kriteria keuntungan (benefit)

c) y_j^- = min (y_{ij}) untuk kriteria keuntungan (benefit)

d) y_j^- = max (y_{ij}) untuk kriteria biaya (cost)

5. Menghitung Jarak Antara Matriks Alternatif dengan Solusi Ideal

   Tahapan ini mengukur tingkat deviasi atau seberapa jauh jarak posisi setiap alternatif dari solusi ideal dengan menerapkan formula *Euclidean Distance*:

   Jarak terhadap solusi ideal positif : $D_i^+ = \sqrt{\sum_{j=1}^{n}\left(y_{ij} - y_j^+\right)^2}$

   Jarak terhadap solusi ideal negatif : $D_i^- = \sqrt{\sum_{j=1}^{n}\left(y_{ij} - y_j^-\right)^2}$

6. Menghitung Nilai Preferensi

   Nilai preferensi berada pada rentang 0 - 1. Semakin besar nilai, semakin baik dan direkomendasikan alternatif tersebut.

$$V_i = \frac{D_i^-}{D_i^- + D_i^+}$$

7. Perangkingan

   Urutan prioritas alternatif disusun berdasarkan akumulasi nilai preferensi, mulai dari skor tertinggi hingga terendah. Dengan demikian, objek alternatif yang memperoleh nilai preferensi paling maksimal ditetapkan sebagai rekomendasi atau solusi terbaik.

Berdasarkan prosedur di atas, kesimpulan utama yang dapat ditarik adalah bahwa algoritma TOPSIS mengidentifikasi pilihan terbaik dengan mengukur kedekatan relatifnya terhadap solusi ideal positif serta keterpautannya dari solusi ideal negatif. Proses ini menghasilkan nilai preferensi (V_i) yang dijadikan indikator utama dalam mengurutkan peringkat seluruh alternatif yang dievaluasi. Apabila perolehan skor preferensi tersebut semakin besar atau mendekati nilai mutlak satu, maka tingkat kelayakan alternatif tersebut untuk direkomendasikan sebagai varian *cushion* terbaik bagi pengguna juga akan semakin tinggi.

4. Sistem Berbasis Web

Menurut Rahmi et al., (2023:822), sistem berbasis web merupakan sebuah platform digital berbentuk kumpulan halaman yang mampu menyajikan berbagai muatan informasi seperti teks, gambar, maupun video. Karakteristik utama dari teknologi ini terletak pada aspek aksesibilitasnya, di mana sistem dapat diakses secara fleksibel dari mana saja dan kapan saja selama terhubung dengan jaringan internet guna memberikan kemudahan dalam menunjang aktivitas manusia. Selaras dengan hal tersebut, Tanaman et al., (2023:44) menyatakan bahwa pemanfaatan sistem berbasis web merupakan solusi ideal untuk mentransformasi mekanisme kerja manual yang tidak efisien menjadi sebuah prosedur digital yang praktis dan berdaya guna tinggi. Melalui sistem yang dijalankan secara daring ini, pengguna dapat memperoleh keunggulan berupa visibilitas data secara seketika (*real-time*) yang

terbukti mampu menyederhanakan kerumitan proses operasional sekaligus memberikan pengalaman penggunaan yang lebih nyaman.

Berdasarkan uraian teori tersebut, dapat disimpulkan bahwa sistem berbasis web merupakan platform digital terintegrasi yang mampu mengelola pengolahan data secara terstruktur serta menyajikan informasi multimedia kepada pengguna secara *real-time*. Fleksibilitas akses daring (*online*) melalui koneksi internet serta kemampuan sistem ini dalam mentransformasi mekanisme manual yang tidak efisien menjadikannya sebuah solusi komputasi yang praktis dan berdaya guna tinggi. Atas dasar tersebut, sistem berbasis web dikategorikan sebagai infrastruktur yang sangat relevan dalam menyederhanakan kerumitan proses operasional, sehingga layanan fungsionalnya dapat dijangkau oleh pengguna secara luas tanpa batasan tempat dan waktu.

5. Hypertext Preprocessor (PHP)

Pengembangan proyek web modern saat ini banyak mengadopsi *Hypertext Preprocessor* (PHP) sebagai salah satu bahasa pemrograman sisi server (*server-side*) yang paling dominan (Sotnik et al., 2023:11). Tingginya preferensi dan distribusi luas terhadap teknologi ini didasari oleh karakteristik unggulannya yang meliputi efisiensi ekonomi, skalabilitas pengembangan (*scalability*), kesederhanaan struktur, serta kompatibilitas yang tinggi terhadap mayoritas penyedia layanan *hosting* untuk membangun sistem berbasis web yang dinamis.

Pemanfaatan PHP ini memegang peranan yang sangat krusial dalam memfasilitasi evolusi halaman website, yaitu mengubah format yang semula bersifat statis dan pasif menjadi platform digital yang dinamis serta interaktif (Sinlae et al., 2024:69). Sebagai bahasa *scripting* yang menyatu langsung dengan dokumen HTML, seluruh baris kode di dalamnya akan dieksekusi secara penuh di lingkungan server, sehingga komponen yang dikirimkan menuju peramban (*browser*) pengguna hanya berupa hasil akhir pemrosesannya saja. Melalui integrasi antara PHP dan MySQL, sebuah sistem dapat dikembangkan untuk menyajikan data secara terstruktur, menyediakan fitur interaksi kompleks seperti formulir masuk (*login*), hingga melakukan personalisasi konten yang adaptif terhadap kebutuhan pengguna.

Sintesis dari berbagai teori di atas menunjukkan bahwa karakteristik PHP sebagai bahasa pemrograman *server-side* menjadikannya pilar fundamental dalam arsitektur pengembangan platform web modern. Kemampuannya dalam menjalankan logika program pada sisi *server* serta mendukung integrasi dengan berbagai sistem manajemen basis data memungkinkan PHP mengolah masukan dari pengguna secara dinamis dan menghasilkan respons yang sesuai. Oleh karena itu, PHP mampu mengubah halaman web yang bersifat statis menjadi aplikasi yang interaktif, dinamis, dan mampu memenuhi kebutuhan pengguna secara lebih efektif.

6. *Framework* Laravel

Sebagai salah satu opsi terpopuler berbasis PHP, framework Laravel dirancang khusus untuk membantu pengembang dalam membangun aplikasi web di tengah meningkatnya kompleksitas kebutuhan bisnis saat ini. Rangkuti et al., (2026), menguraikan bahwa fondasi utama platform ini bertumpu pada pengadopsian pola arsitektur *Model-View-Controller* (MVC) yang menerapkan prinsip *Separation of Concerns* (SoC), yaitu sebuah pendekatan rekayasa perangkat lunak yang mengisolasi komponen manajemen data, visual antarmuka, dan aliran instruksi secara mandiri. Pemisahan struktural ini sengaja diintegrasikan untuk mendukung filosofi *code expressiveness* agar sintaks program yang dihasilkan tidak sekadar fungsional, melainkan juga memiliki keterbacaan yang tinggi serta mudah dirawat dalam jangka panjang.

Selaras dengan karakteristik tersebut, kemampuan framework Laravel dalam memisahkan logika bisnis dari aspek presentasi visual terbukti mampu menciptakan tata kelola kode yang bersih sekaligus meminimalkan redundansi lewat prinsip *Don't Repeat Yourself* (DRY) (Sinlae et al., 2024:120). Di samping kekuatan rancangan arsitekturnya, akselerasi pengerjaan komponen *backend* disokong oleh ekosistem fitur bawaan yang melimpah. Pengembang dapat memanfaatkan *Eloquent Object-Relational Mapping* (ORM) untuk berinteraksi dengan basis data secara lebih intuitif melalui sintaks objek terstruktur, sehingga mereduksi kerumitan penulisan kueri SQL konvensional yang kompleks (Sinlae et al.,

2024:121). Siklus hidup pengelolaan data tersebut kemudian disempurnakan oleh sistem migrasi database berbasis kode PHP yang mempermudah kolaborasi tim serta menyediakan fungsionalitas *rollback* untuk memulihkan kondisi skema data demi menjaga stabilitas operasional aplikasi.

Berdasarkan uraian teori tersebut, dapat disimpulkan bahwa *framework* Laravel merupakan kerangka kerja pengembangan aplikasi web yang mengutamakan efisiensi melalui penerapan arsitektur *Model-View-Controller* (MVC) serta dukungan fitur otomatisasi dalam pengelolaan basis data. Kombinasi kedua aspek tersebut menghasilkan struktur kode yang lebih terorganisir, bersih, dan mudah dikembangkan sesuai dengan kebutuhan aplikasi. Oleh karena itu, Laravel menjadi *framework* yang sesuai untuk membangun aplikasi web dinamis yang memerlukan pengelolaan logika bisnis dan manipulasi data yang kompleks secara sistematis.

7. MySQL

Pemilihan sistem manajemen basis data yang sesuai menjadi salah satu aspek penting dalam pengembangan aplikasi berbasis web karena berpengaruh terhadap kinerja serta stabilitas sistem pada sisi backend. Menurut Putri et al., (2023), MySQL merupakan perangkat lunak *Relational Database Management System* (RDBMS) berlisensi *open-source* yang menjadi pilihan utama bagi para pengembang karena dinilai lebih stabil, cepat, mudah digunakan, serta memiliki jaminan keamanan

akses yang kuat. Platform ini mengadopsi *Structured Query Language* (SQL) sebagai bahasa standar penulisan kueri dan memiliki keunggulan fungsional yang mendukung akses multitumpu, di mana banyak pengguna dapat berinteraksi dengan basis data secara bersamaan (*real-time*).

Kemampuan MySQL dalam menangani beban operasional tersebut didukung oleh struktur penyimpanan datanya yang fleksibel dan terorganisir.. Silalahi (2022:2) menguraikan bahwa sebagai sistem manajemen basis data relasional, server MySQL mampu mengelola banyak basis data secara simultan, di mana informasi di dalamnya disimpan ke dalam satu atau lebih entitas tabel. Struktur tabel ini diatur ke dalam baris yang merepresentasikan entitas data dan kolom yang memuat item informasi spesifik dari entitas tersebut. Lebih lanjut, tabel-tabel di dalam basis data MySQL dapat saling dihubungkan melalui relasi antar-kolom yang cocok, sehingga memungkinkan pengelolaan data yang kompleks dapat berjalan secara sistematis dan efisien.

Berdasarkan uraian teori tersebut, dapat disimpulkan bahwa MySQL merupakan sistem manajemen basis data relasional yang menyediakan infrastruktur penyimpanan data secara terstruktur, cepat, dan stabil. Kemampuannya dalam mengelola hubungan antar-tabel serta mendukung penulisan kueri standar menjadikan MySQL mampu menjaga keteraturan struktur dan keamanan akses informasi yang tersimpan. Oleh karena itu, dalam pengembangan sistem yang dinamis, MySQL menjadi salah satu komponen krusial yang memfasilitasi tata kelola data secara

sistematis serta mampu melayani banyak pengguna secara bersamaan *(real-time)*.

8. Flowchart

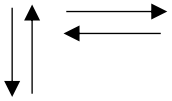
Dalam tahapan perancangan sistem, diagram alir *(flowchart)* memegang peranan krusial sebagai representasi grafis yang memetakan langkah-langkah serta urutan prosedur dari suatu program. Menurut Sutanti et al., (2020) *flowchart* merupakan diagram berbentuk gambar yang mengadopsi aliran sekuensial satu atau dua arah untuk merepresentasikan dan mendesain komponen-komponen di dalam bahasa pemrograman. Melalui visualisasi tersebut, sistem ini membantu *analyst* maupun *programmer* dalam memecahkan masalah kompleks ke dalam segmen-segmen yang lebih kecil, sehingga mempermudah proses analisis terhadap berbagai alternatif lain dalam pengoperasian sistem.

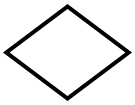
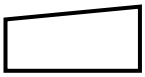
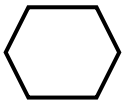
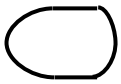
Pecahan segmen-segmen prosedural tersebut kemudian diperjelas melalui penggambaran alur kerja yang mendalam guna mengidentifikasi setiap aktivitas sistem secara terstruktur. Listyoningrum et al., (2023:101), memaparkan bahwa implementasi *flowchart* sangat efektif untuk memvisualisasikan langkah-langkah kompleks, keputusan, serta alur data yang terlibat di dalam sistem secara terperinci. Dengan menyajikan pandangan mendalam terhadap interaksi yang rumit tersebut, alat visual ini mampu meminimalkan potensi kesalahan atau kebingungan selama proses perancangan, sekaligus memfasilitasi komunikasi yang efektif di

dalam tim serta mendukung pemantauan dan evaluasi kinerja sistem secara sistematis.

Berdasarkan uraian teori tersebut, dapat disimpulkan bahwa *flowchart* merupakan salah satu metode perancangan sistem yang efektif untuk menggambarkan alur proses dan logika algoritma ke dalam bentuk simbol-simbol grafis yang terstandarisasi. Penggunaan *flowchart* memudahkan proses analisis terhadap hubungan antarproses di dalam sistem secara terstruktur sehingga mengurangi kesalahan dalam memahami alur kerja sebelum tahap implementasi program dilakukan. Visualisasi beserta fungsionalitas dari komponen baku yang digunakan dalam perancangan *flowchart* dipaparkan melalui strukturnya pada tabel di bawah ini.

Tabel 2. 1 Simbol-Simbol *Flowchart*

Simbol	Keterangan
	**Flow Direction** Penghubung antar-notasi untuk menunjukkan arah alur proses (*connecting line*).
	**Terminator** Penanda titik inisiasi (*start*) atau bagian akhir (*stop*) suatu aktivitas..
	**Connector Symbol** Jalur keluar-masuk atau penyambungan proses pada halaman yang sama.
	**Connector Symbol** Jalur keluar-masuk atau penyambungan proses pada halaman yang berbeda.
	**Processing Symbol** Menunjukkan tahapan pemrosesan atau komputasi data oleh komputer.

	**Simbol Manual Operation** Menunjukkan proses pengolahan fisik tanpa melibatkan sistem komputer.
	**Simbol Decision** Penentu arah alur berdasarkan evaluasi kondisi atau pilihan yang ada.
	**Symbol *Input-Output*** Operasi memasukkan (*input*) atau mengeluarkan (*output*) data secara umum.
	**Simbol Manual *Input*** Proses entri data secara langsung oleh pengguna melalui *keyboard online*.
	**Simbol Preparation** Penyiapan atau alokasi ruang memori (*storage*) untuk pemrosesan data.
	**Simbol Predefine Proses** Eksekusi modul terpisah, *procedure*, atau bagian dari *sub-program*.
	**Simbol Display** Media keluaran visual seperti layar monitor, *printer*, maupun *plotter*.
	**Simbol disk and On-line Storage** Proses penyerapan data dari *disk* atau penyimpanan informasi ke *disk*.
	**Simbol magnetik tape Unit** Operasi membaca data dari pita magnetik atau menyimpannya ke pita magnetik.
	**Simbol Punch Card** Menunjukkan *input* bersumber dari kartu atau *output* yang ditulis ke kartu.
	**Simbol Dokumen** Menunjukkan *input* dari berkas kertas atau *output* yang dicetak fisik.

Sumber: Sutanti et al., (2020)

9. UML

Dalam proses perancangan perangkat lunak, Unified Modeling Language (UML) digunakan sebagai alat pemodelan untuk

menggambarkan rancangan sistem sebelum diimplementasikan. Pada penerapannya, Aryani et al., (2024), menunjukkan bahwa penggunaan UML dalam perancangan aplikasi perpustakaan berbasis web mampu mendukung pengelolaan dan pendataan informasi secara lebih sistematis dan terkomputerisasi. Penerapan tersebut diharapkan dapat meningkatkan efektivitas dan efisiensi pengelolaan sistem sesuai dengan kebutuhan organisasi.

Sebagai instrumen pendukung dalam tahap analisis dan perancangan sistem, UML menyediakan kemampuan visualisasi yang membantu pengembang dalam memodelkan struktur serta alur sistem secara terstruktur. Anardani et al., (2023:524) menjelaskan bahwa UML mempermudah pengembang dalam membangun sistem melalui berbagai jenis diagram, baik yang bersifat statis maupun dinamis, untuk menggambarkan spesifikasi, konstruksi, dan dokumentasi sistem berbasis objek. Melalui pemodelan visual tersebut, komunikasi antara pengguna dan pengembang menjadi lebih efektif, sekaligus memberikan acuan yang jelas dalam proses implementasi sistem ke dalam kode program.

Uraian teori tersebut, menegaskan kedudukan UML sebagai bahasa grafis standar dalam memetakan spesifikasi perangkat lunak berbasis objek. Kehadiran UML berfungsi menjembatani kebutuhan fungsional dengan desain arsitektur sistem secara terstruktur. Akibatnya, pemanfaatan model ini mampu menciptakan alur pengembangan sistem informasi yang

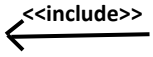
lebih tertata, sekaligus mengoptimalkan fase implementasi serta perawatan perangkat lunak.

a. *Use Case Diagram*

Dalam fase analisis dan pengembangan perangkat lunak, *Use Case Diagram* digunakan sebagai instrumen visual untuk memetakan ruang lingkup fungsionalitas sistem serta menetapkan batas sistem *(system boundary)*. Wayahdi & Ruziq, (2023), menambahkan bahwa diagram ini berfungsi untuk mengomunikasikan fungsi-fungsi utama *(high-level function)* sistem serta menggambarkan interaksi antara aktor dengan sistem dari sudut pandang pengguna. Lebih lanjut, identifikasi aktor dan *use case* membentuk *system boundary* yang membedakan secara jelas aktivitas yang dilakukan oleh sistem dengan aktivitas yang dilakukan oleh entitas di luar sistem.

Melalui rangkuman teoritis di atas, dapat disimpulkan bahwa *Use Case Diagram* bertindak sebagai instrumen grafis untuk memetakan spesifikasi fungsional sistem berdasarkan jalinan interaksi antara aktor dan kapabilitas yang disediakan. Diagram ini membantu memperjelas batasan sistem, mengurangi ambiguitas dalam pendefinisian fungsi bisnis, serta menjadi acuan dalam proses perancangan sebelum memasuki tahap pengembangan perangkat lunak. Adapun ragam notasi dalam penyusunan *Use Case Diagram* dapat dicermati pada tabel di bawah ini.

Tabel 2. 2 Simbol *Use case Diagram*

Simbol	Keterangan
	**Aktor** Representasi manusia, sistem luar, atau perangkat yang berinteraksi dengan *use case*.
	***Use case*** Abstraksi fungsi atau bentuk interaksi yang terjadi antara sistem dan aktor..
	**Assocation** Garis relasi yang menghubungkan jalinan komunikasi antara aktor dan *use case*.
	Generalisasi Menunjukkan hubungan spesialisasi atau penurunan sifat antar-aktor atau antar-*use case*.
	**Include** Menandakan bahwa suatu *use case* wajib menyertakan fungsionalitas dari *use case* lain.
	**Extend** Perluasan fungsi opsional dari suatu *use case* apabila kondisi tertentu terpenuhi.

Sumber: Gunawan et al., (2023)

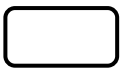
b. Activity Diagram

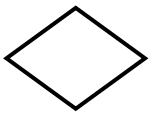
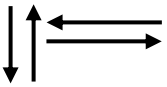
Dalam pemodelan sistem menggunakan UML, *Activity Diagram* diterapkan untuk memvisualisasikan alur aktivitas atau rangkaian proses berjalan di dalam perangkat lunak yang akan dibangun. Narulita et al., (2024) memaparkan bahwa diagram ini menggambarkan tahapan operasional sistem secara berurutan, mulai dari proses inisiasi awal, titik percabangan keputusan, hingga

bagaimana sebuah proses berakhir, termasuk memetakan proses paralel yang terjadi saat sistem dieksekusi. Lebih lanjut, perancangan diagram ini memiliki hubungan erat dengan pemodelan fungsional, di mana satu atau beberapa *use case* minimal direpresentasikan oleh satu *activity diagram* guna merinci jalannya proses di dalam sistem, sementara *use case* itu sendiri berfokus pada interaksi aktor dengan sistem.

Sintesis dari landasan teoritis di atas mengarah pada konklusi bahwa *Activity Diagram* bertindak sebagai instrumen pemodelan visual yang digunakan untuk menjabarkan tata alur kerja serta karakteristik dinamis dari sebuah sistem. Kehadiran diagram ini memberikan kontribusi besar dalam menyederhanakan tahapan evaluasi proses operasional, sekaligus memandu transformasi logika internal sistem ke dalam baris kode program secara presisi. Adapun berbagai notasi baku yang diterapkan dalam penyusunan *Activity Diagram* dipaparkan secara terperinci pada tabel di bawah.

Tabel 2. 3 Simbol *Activity Diagram*

Simbol	Keterangan
	**Activity** Visualisasi pola interaksi antarkelas antarmuka (*interface*) di dalam sistem.
	**Action** Kondisi sistem yang mencerminkan eksekusi dari suatu tindakan spesifik.

	**Initial Node** Penanda titik awal dimulainya seluruh rangkaian alur aktivitas.
	**Activity Final Node** Penanda titik akhir selesainya seluruh rangkaian alur aktivitas.
	**Decision** Pilihan tindakan atau percabangan alur berdasarkan parameter kondisi tertentu.
	**Line Connector** Garis penghubung arah aliran (*flow*) dari satu notasi ke notasi lainnya.

Sumber: Gunawan et al., (2023)

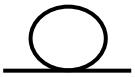
c. Sequence Diagram

Dalam perancangan arsitektur perangkat lunak, *Sequence Diagram* digunakan sebagai instrumen visual untuk memetakan alur interaksi dan koordinasi antarobjek di dalam sistem secara terstruktur. Menurut Sunarsa, (2024), diagram ini sangat efektif untuk memvisualisasikan bagaimana komponen utama seperti *actor*, *view*, *controller*, dan *model* saling bekerja sama untuk memproses data hingga menghasilkan *output* yang tepat. Melalui pemodelan ini, pengembang dapat menggambarkan alur kontrol secara kronologis, mulai dari pengguna *(actor)* yang memicu aksi melalui antarmuka *(view)*, pengaturan alur kontrol oleh *controller*, hingga pengelolaan struktur data pada *model* yang berkomunikasi langsung dengan *database*.

Berdasarkan teori-teori tersebut, dapat diambil kesimpulan bahwa *Sequence Diagram* bertindak sebagai instrumen pemodelan

untuk mengilustrasikan dinamika hubungan antarelemen objek sekaligus kronologi komunikasi yang berlangsung sepanjang eksekusi suatu proses. Keberadaan diagram ini memberikan kemudahan bagi tim pengembang untuk memetakan masa hidup (*lifeline*) tiap-tiap objek, lalu lintas pengiriman pesan, hingga korelasi antarkomponen sistem, guna memvalidasi ketepatan logika komunikasi sebelum akhirnya ditransformasikan ke dalam fase pengodean program. Adapun ragam notasi standar yang diterapkan dalam penyusunan *Sequence Diagram* dirangkum secara terstruktur melalui tabel di bawah ini.

Tabel 2. 4 Simbol *Squence Diagram*

Simbol	Keterangan
	**Actor** Representasi pengguna manusia atau entitas luar yang berinteraksi dengan sistem.
	**Entinity Class** Representasi elemen data atau informasi persisten yang dikelola oleh sistem.
	**Boundary Class** Komponen antarmuka yang menjembatani komunikasi pengguna dengan sistem internal.
	**Control Class** Pengatur alur logika bisnis yang menghubungkan boundary dengan tabel data.
	**A Focus of Control & A Life Line** Penanda linimasa aktif objek sekaligus titik awal dan akhir pertukaran pesan.

⟶	**A Message** Garis penunjuk arah pengiriman pesan atau komunikasi antar-objek.

Sumber: Gunawan et al., (2023)

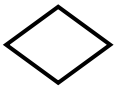
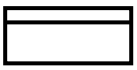
d. Class Diagram

Dalam perancangan sistem berorientasi objek, *Class Diagram* digunakan untuk menangkap struktur serta memvisualisasikan hubungan antar kelas yang membentuk arsitektur perangkat lunak. Menurut Ramdany et al., (2024), diagram ini mampu memberikan pandangan global atas sebuah sistem melalui representasi kumpulan kelas dan relasinya seperti *associations*, *generalization*, dan *aggregation* yang dilengkapi dengan spesifikasi atribut serta operasi *(method)*. Selain berfungsi menentukan perilaku sistem melalui aturan dan tanggung jawab entitas pada tahap analisis, implementasi diagram ini juga berperan penting untuk menggambarkan alur jalannya sebuah basis data *(database)* guna menyelaraskan hubungan antara perancangan dengan proses pembuatan programnya.

Rangkuman teoritis di atas mengarah pada konklusi bahwa Class Diagram merupakan instrumen pemodelan arsitektural yang digunakan untuk menegaskan struktur data beserta interelasi objek pada sistem berorientasi objek. Diagram ini membantu memastikan konsistensi struktur data, keterkaitan antar kelas, serta mempermudah proses implementasi sistem ke dalam basis data dan

kode program. Dengan demikian, *Class Diagram* berperan penting dalam mendukung pengembangan perangkat lunak yang terstruktur, mudah dikembangkan, dan memiliki integritas data yang baik. Adapun simbol yang diterapkan dalam rancangan *Class Diagram* diuraikan secara terstruktur melalui tabel di bawah ini.

Tabel 2. 5 Simbol *Class Diagram*

Simbol	Keterangan
	Garis Lurus Notasi pewarisan sifat, struktur data, dan perilaku dari kelas induk (ancestor) ke kelas anak (descendant).
	**Nary Association** Penghubung relasi struktural kompleks yang melibatkan lebih dari dua objek atau kelas sekaligus.
	**Class** Cetak biru (blueprint) dari kumpulan objek yang memiliki kesamaan atribut, operasi, dan perilaku.
	**Collaboration** Spesifikasi pola interaksi antar elemen untuk mewujudkan fungsi atau hasil sistem yang terukur.
	**Realization** Hubungan yang menandakan perwujudan atau implementasi nyata dari operasi yang didefinisikan oleh elemen lain.
	**Dependency** Relasi ketergantungan di mana perubahan pada elemen mandiri (independent) akan memengaruhi elemen di bawahnya.
	Association Garis relasi statis yang menghubungkan jalur komunikasi antar-objek di dalam sistem.

Sumber: Gunawan et al., (2023)

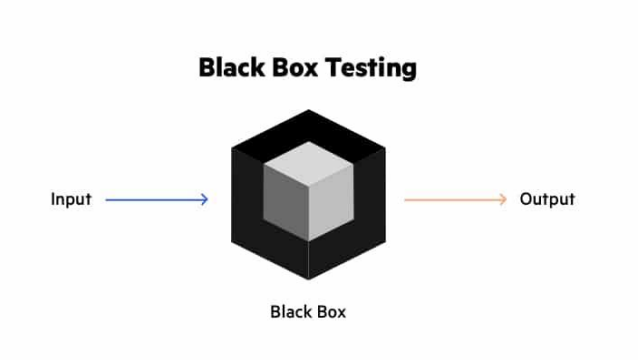
10. Metode Pengujian

a. Black Box Testing

Pengujian perangkat lunak merupakan tahapan krusial sebelum sistem dipublikasikan guna memastikan seluruh fungsi berjalan dengan baik dan sesuai kebutuhan pengguna. Abdillah et al., (2023) menjelaskan bahwa salah satu metode pengujian yang umum digunakan adalah *black box testing*, yang berfokus pada evaluasi spesifikasi fungsional perangkat lunak tanpa menguji desain internal maupun kode programnya. Melalui pendekatan ini, pengujian dilakukan dengan memasukkan data untuk memastikan kesesuaian antara masukan *(input)* dan keluaran *(output)* sistem, sehingga dapat meminimalkan risiko kesalahan fungsionalitas saat perangkat lunak diimplementasikan.

Dalam proses rekayasa perangkat lunak, *black-box testing* diterapkan sebagai metode pengujian fungsional yang dilakukan tanpa memperhatikan struktur internal maupun logika kode program di balik layar. Menurut Saputra & Mardiati, (2025) pengujian ini menitikberatkan pada analisis terhadap hubungan antara masukan *(input)* yang diberikan dengan keluaran *(output)* serta respons yang dihasilkan oleh sistem. Tujuan utama dari pendekatan ini adalah untuk memastikan bahwa fungsionalitas sistem berjalan sepenuhnya sesuai dengan spesifikasi yang telah ditetapkan, sehingga pengembang dapat mengevaluasi pemenuhan syarat fungsional guna membangun sistem

baru yang mampu meningkatkan pelayanan informasi secara lebih baik.



Gambar 2. 1 Metode *Black Box Testing*

Rangkuman teori di atas menegaskan kesimpulan bahwa black-box testing bertindak sebagai metodologi validasi dan verifikasi sistem untuk menakar kapabilitas fungsi perangkat lunak berdasarkan hasil hubungan dinamis antara input yang dimasukkan dan output yang diproduksi. Karena mekanisme pengujian ini mengabaikan kompleksitas arsitektur kode di sisi backend, orientasi penilaiannya murni bertumpu pada ketepatan fungsi sistem dalam menjawab kebutuhan yang telah ditetapkan. Dengan karakteristik tersebut, pendekatan black-box menjadi sebuah solusi efektif untuk memastikan bahwa sistem informasi yang dikembangkan memiliki tingkat ketahanan, stabilitas, dan kualitas fungsional yang prima guna menopang seluruh aktivitas operasional pengguna.

b. System Usability Scale (SUS)

Dalam tahap akhir pengembangan perangkat lunak, pengujian kelayakan dilakukan dengan mengimplementasikan kuesioner *System*

Usability Scale (SUS) guna menakar tingkat kepuasan pengguna secara langsung di lapangan. Berdasarkan pendekatan praktis dari Tuloli et al., (2022), evaluasi subjektif melalui kuesioner ini sangat efektif untuk menjaring umpan balik dari pengguna riil serta mendeteksi kendala operasional yang masih sering ditemui saat sistem dijalankan. Data kepuasan yang diperoleh dari hasil pengujian ini nantinya menjadi dasar bagi pengembang untuk memastikan bahwa aplikasi yang dirancang telah memenuhi standar kegunaan *(usability)* secara utuh sebelum diserahkan kepada pengguna akhir.

Dalam pelaksanaannya, proses evaluasi ini dilakukan secara sistematis dengan mendistribusikan instrumen penilaian berupa sepuluh butir pernyataan menggunakan model skala Likert lima poin, mulai dari sangat tidak setuju hingga sangat setuju. Menurut Hakim et al., (2025), pengumpulan data penilaian subjektif tersebut akan menghasilkan skor kuantitatif konkret yang merepresentasikan pengalaman pengguna *(user experience)* secara nyata saat mencoba sistem. Skor akhir dari pengujian kuantitatif inilah yang menjadi tolok ukur reliabel bagi pengembang untuk menilai tingkat penerimaan pengguna serta kelayakan sistem sebelum aplikasi siap dirilis secara resmi.

Berdasarkan jabaran landasan teoritis di atas, dapat ditarik sebuah konklusi bahwa *System Usability Scale* (SUS) merupakan instrumen evaluasi terstandar yang memiliki kapabilitas untuk

mengukur parameter keberdayaan atau kegunaan sistem dengan mengacu pada persepsi langsung serta pengalaman empiris dari pihak pengguna. Hasil pengujian SUS memberikan nilai kuantitatif yang dapat digunakan sebagai dasar dalam menilai kualitas antarmuka, kemudahan penggunaan, serta tingkat penerimaan pengguna terhadap sistem yang dikembangkan. Oleh karena itu, metode SUS dapat dimanfaatkan sebagai acuan dalam melakukan evaluasi dan penyempurnaan sistem agar lebih mudah digunakan serta sesuai dengan kebutuhan pengguna.

B. Kajian Empiris

Penerapan Sistem Pendukung Keputusan (SPK) dalam menentukan prioritas pemilihan produk terus mengalami perkembangan, khususnya pada bidang kosmetik dan *skincare.* Sejumlah studi terdahulu telah membuktikan bahwa implementasi pendekatan *Multi-Criteria Decision Making (MCDM)* sanggup memproduksi luaran rekomendasi yang lebih objektif melalui kalkulasi berbagai parameter secara simultan. Salah satu studi terkait penentuan produk cushion yang dilakukan oleh Wahyuni et al., (2024), menggunakan metode *Additive Ratio Assessment* (ARAS). Temuan dari riset tersebut mengindikasikan bahwa variabel-variabel seperti harga, daya tahan produk, serta ketersediaan variasi bertindak sebagai indikator utama dalam siklus pengambilan keputusan. Meskipun mengkaji domain yang serupa, penelitian yang dilakukan saat ini memiliki distingsi dan kebaruan jika dibandingkan dengan studi terdahulu. Perbedaan tersebut terletak pada

penerapan metode *Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)* yang diintegrasikan ke dalam ekosistem *framework* Laravel. Selain itu, sistem ini juga diperkuat dengan tahapan pengujian usability, sehingga sistem yang dikembangkan tidak hanya menghasilkan rekomendasi yang akurat secara komputasional, tetapi juga memperhatikan kemudahan penggunaan sesuai dengan kebutuhan pengguna.

Di sisi lain, efektivitas metode TOPSIS pada bidang kecantikan juga telah dibuktikan oleh Azzahra & Sulaiman, (2025) melalui penelitian mengenai pemilihan produk *lip cream* terbaik menggunakan pendekatan gabungan AHP-TOPSIS. Hasil penelitian tersebut menunjukkan bahwa metode TOPSIS mampu menghasilkan peringkat produk yang akurat berdasarkan berbagai preferensi konsumen. Sejalan dengan penelitian tersebut, Nadya et al., (2022) menerapkan metode TOPSIS untuk memberikan rekomendasi produk *sunscreen* yang disesuaikan dengan karakteristik kulit berjerawat. Temuan dari studi tersebut mengonfirmasi bahwa metodologi TOPSIS memiliki kapabilitas yang tinggi dalam memproduksi urutan prioritas atau pemeringkatan produk secara presisi dengan mengakomodasi beragam preferensi konsumen.

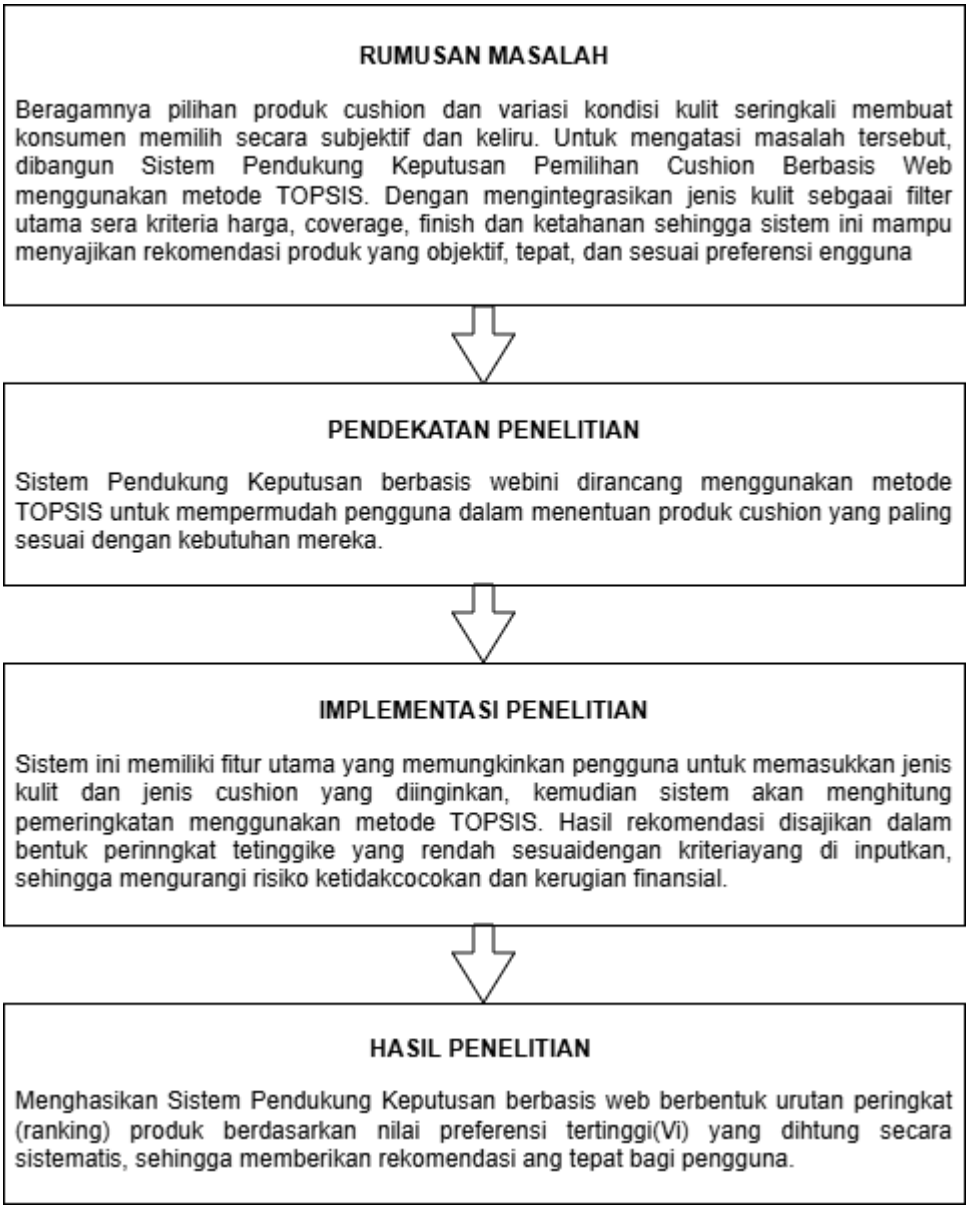
Keandalan metode TOPSIS juga didukung oleh berbagai penelitian di luar bidang kecantikan. Aisyah et al., (2024) menerapkan metode TOPSIS pada kasus pemilihan pemasok (*supplier*), sedangkan Sari et al., (2023) menggunakannya untuk pemilihan aplikasi *chat*. Hasil kedua penelitian tersebut menunjukkan bahwa metode TOPSIS memiliki stabilitas perhitungan yang baik serta mampu menghasilkan keputusan yang konsisten. Prinsip

TOPSIS yang mencari jarak terdekat dengan solusi ideal positif dan jarak terjauh dari solusi ideal negatif membuatnya sangat mudah diterapkan pada sistem web untuk pengambilan keputusan secara objektif.

Berbagai studi terdahulu menjadi pijakan bagi penelitian ini untuk menerapkan metode TOPSIS dalam sistem pendukung keputusan pemilihan *cushion*. Kebaruan yang ditawarkan terletak pada pengembangan sistem berbasis *framework* Laravel yang dipadukan dengan pengujian *usability* (SUS). Alhasil, platform ini tidak hanya menghasilkan rekomendasi TOPSIS yang akurat, tetapi juga menawarkan kemudahan operasional yang optimal bagi pengguna.

C. Kerangka Berpikir

Konstruksi berpikir dalam studi ini dirancang sebagai landasan penyelesaian masalah yang dihadapi, yaitu kompleksitas yang dihadapi pengguna dalam memilih produk *cushion*. Kerangka berpikir ini menggambarkan keterkaitan antara teori-teori yang telah dikaji dengan proses pengembangan sistem yang dilakukan secara terstruktur. Secara umum, kerangka berpikir menghubungkan kebutuhan pengguna, penerapan metode TOPSIS sebagai metode perhitungan, hingga menghasilkan Sistem Pendukung Keputusan yang mampu memberikan rekomendasi produk secara objektif. Alur logis penelitian secara keseluruhan dapat divisualisasikan melalui diagram kerangka berpikir yang disajikan pada gambar berikut.



Gambar 2. 2 Kerangka Berpikir