BAB II

TINJAUAN LITERATUR

2.1 Penelitian Terkait

Tinjauan pustaka pada penelitian ini dimulai dengan mengkaji berbagai pendekatan yang telah dikembangkan dalam sistem deteksi plagiarisme dokumen, terutama yang memanfaatkan kemiripan teks berbasis vektor. Metode *cosine similarity* banyak digunakan karena kemampuannya mengukur kesamaan sudut antar vektor representasi dokumen secara efisien, seperti yang dijelaskan dalam studi terkait *text mining* dan pemrosesan dokumen digital (Dealova, Pradana and Ramadhan, 2026). Selanjutnya, penelitian ini meninjau implementasi teknik praproses teks seperti *tokenization*, *stop word removal*, dan *stemming* yang krusial untuk meningkatkan akurasi deteksi. Selain itu, pengembangan antarmuka pengguna (*Graphical User Interface*) dengan Python dan Tkinter juga ditelusuri dari berbagai literatur untuk memastikan sistem yang dibangun tidak hanya akurat secara komputasi, tetapi juga mudah dioperasikan oleh pengguna umum.

Penelitian oleh (Fadhullah, Fauziah and Winarsih, 2022) Jurnal ini membahas tentang pembuatan aplikasi berbasis web untuk mendeteksi dini plagiarisme pada penelitian ilmiah menggunakan algoritma *cosine similarity*. Tujuannya adalah membantu mahasiswa tingkat akhir agar bisa mengecek kemiripan judul tugas akhir mereka terhadap judul-judul yang sudah ada, sehingga terhindar dari revisi di awal karena plagiarisme yang tidak disengaja. Metode yang dipakai adalah *cosine similarity*, yang menghitung tingkat kemiripan antar

dokumen berdasarkan vektor kata kunci setelah melalui proses *noise removal*, *case folding*, *stemming*, dan *stopword removal*. Dari hasil pengujian terhadap 100 data latih, aplikasi ini mampu menampilkan persentase kemiripan tertinggi sebesar 31,18% dengan status “sedang”, dan dianggap berjalan cukup baik meskipun masih perlu pengembangan lebih lanjut, seperti penambahan metode klasifikasi lain atau efisiensi perhitungan. Penelitian ini juga mereview berbagai studi terkait yang menggunakan *cosine similarity* untuk berbagai kasus, mulai dari klasifikasi buku, pencarian terjemahan Al-Qur'an, hingga penilaian otomatis jawaban pendek.

Penelitian oleh (Supiyanto and Sriyono, 2023) jurnal ini membahas tentang penerapan metode *cosine similarity* untuk mendeteksi kemiripan pada dokumen teks menggunakan bahasa pemrograman Python. Tujuan penelitiannya adalah membangun aplikasi yang bisa menghitung tingkat kemiripan antar dokumen berformat .txt melalui proses *pre-processing* seperti *case folding* (mengubah huruf menjadi kecil), *tokenizing* (pemotongan kata), *stopword removal* (menghilangkan kata umum seperti "dan", "di"), *stemming* (menghilangkan imbuhan), serta distribusi frekuensi kata. Setelah data teks diubah menjadi vektor, kemiripan dihitung menggunakan rumus *cosine similarity* yang menghasilkan nilai antara 0 (tidak mirip) hingga 1 (sangat mirip lalu dikonversi ke persentase). Hasil pengujian menunjukkan bahwa aplikasi mampu mendeteksi kemiripan 100% untuk dokumen yang sama persis, di atas 80% untuk kalimat yang hanya diubah dari aktif ke pasif setelah melalui *pre-processing*, serta kemiripan antar dokumen berbeda tema seperti sekitar 26,79% hingga 35,61%. Penulis juga menyarankan pengembangan

lebih lanjut, seperti mendukung format file lain (PDF, Word) dan penambahan antarmuka GUI agar lebih menarik dan ramah pengguna.

Penelitian oleh (Arkan *et al.*, 2026) jurnal ini membahas perancangan aplikasi deteksi plagiarisme offline menggunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) dan *cosine similarity* berbasis *Natural Language Processing* (NLP). Latar belakangnya didasari oleh keterbatasan akses terhadap alat deteksi komersial seperti Turnitin yang mahal, sementara alat gratis memiliki batasan jumlah kata dan masalah privasi data karena harus mengunggah dokumen ke server eksternal. Aplikasi yang dikembangkan menggunakan Python 3.13, framework PySide6 untuk antarmuka pengguna, dan SQLite sebagai database. Proses deteksi dimulai dengan *text preprocessing* yang meliputi *case folding*, *punctuation removal*, *tokenization*, *stopword removal*, dan *stemming* menggunakan Sastrawi untuk bahasa Indonesia. Selanjutnya, dilakukan pembobotan TF-IDF untuk mengekstraksi fitur kata penting, lalu kemiripan antar dokumen dihitung dengan rumus *cosine similarity*. Hasil pengujian pada satu dokumen uji terhadap sembilan dokumen pembanding menghasilkan tingkat kemiripan tertinggi sebesar 9,38% (menggunakan metode *scikit-learn*) dengan waktu pemrosesan 8,55 detik, yang tidak termasuk kategori indikasi plagiarisme. Aplikasi ini diharapkan menjadi solusi gratis, mandiri, dan aman bagi mahasiswa serta institusi pendidikan, meskipun masih terbatas pada deteksi tingkat leksikal (berdasarkan kata) dan belum menyentuh aspek semantik.

Penelitian oleh (Lumbansiantar, Dwiasnati and Fatonah, 2023) jurnal ini membahas penerapan metode *cosine similarity* untuk mendeteksi plagiarisme pada

abstrak jurnal online. Latar belakangnya adalah maraknya kasus plagiarisme di kalangan akademik, terutama karena kemudahan akses jurnal digital yang justru memicu perilaku salin-tempel. Penelitian ini menggunakan 100 data abstrak jurnal yang dikumpulkan dari situs penerbit ilmiah, kemudian diproses melalui tahap *pre-processing* seperti *case folding*, *tokenizing*, *stopword removal*, dan *stemming* untuk membersihkan teks dari inkonsistensi. Selanjutnya, dilakukan pembobotan kata menggunakan metode TF-IDF untuk menghitung pentingnya setiap kata dalam dokumen, lalu kemiripan antar dokumen dihitung dengan rumus *cosine similarity*. Hasil penelitian menunjukkan bahwa metode *cosine similarity* mampu mendeteksi tingkat kemiripan antar dokumen dengan menghasilkan nilai dalam rentang 0 hingga 1, yang kemudian diurutkan dari kemiripan terkecil hingga terbesar. Penelitian ini tidak sampai pada tahap pembuatan aplikasi, melainkan lebih fokus pada pengujian algoritma. Kesimpulannya, *cosine similarity* efektif digunakan untuk mendeteksi kemiripan dokumen jurnal, dengan ambang batas plagiarisme yang dapat disesuaikan berdasarkan kebijakan masing-masing institusi (misalnya ringan <30%, sedang 30-70%, berat >70%).

Penelitian oleh (Darmanto, Indah Pradasari and Wahyudi, 2024) jurnal ini membahas pengembangan sistem deteksi plagiarisme dan kesalahan penulisan pada tugas akhir mahasiswa di Politeknik Negeri Ketapang menggunakan pendekatan Natural Language Processing (NLP). Latar belakangnya adalah masih konvensionalnya pengajuan judul tugas akhir yang rawan plagiarisme serta sulitnya pengelolaan data administrasi. Penelitian ini menggabungkan dua algoritma,

yaitu Jaro-Winkler untuk mendeteksi kemiripan string dan kesalahan penulisan (seperti typo atau transposisi huruf), serta TF-IDF untuk menghitung bobot pentingnya kata dalam dokumen. Data yang digunakan sebanyak 20 dokumen tugas akhir mahasiswa yang diambil secara acak. Tahapan pre-processing meliputi case folding, punctuation removal, stemming, dan stopword removal. Hasil pengujian menunjukkan bahwa algoritma Jaro-Winkler mampu mengukur kemiripan teks dengan sangat tinggi (contohnya 0,98091 untuk dua kalimat yang hampir sama), sementara TF-IDF berhasil memberikan bobot pada kata-kata penting seperti "speedboat", "transportasi", "aplikasi", dan "penumpang". Sistem yang dibangun memiliki fitur koreksi ejaan dengan menyoroti kata salah berwarna merah serta menampilkan persentase kemiripan antar dokumen. Penelitian ini menyimpulkan bahwa sistem berjalan dengan baik berdasarkan pengujian black-box dan UAT, serta direkomendasikan untuk menguji efektivitas algoritma ini dengan algoritma lain di masa mendatang.

Berdasarkan referensi tersebut penelitian ini terletak pada beberapa hal. Pertama, dari kelima jurnal di atas, tidak ada yang secara spesifik mengintegrasikan antarmuka GUI berbasis Tkinter dengan algoritma cosine similarity untuk deteksi plagiarisme dokumen; kebanyakan masih berbasis web (Fadhullah), CLI (Supiyanto & Sriyono), atau PySide6 (Arkan). Tkinter sebagai library bawaan Python menawarkan kemudahan deployment tanpa instalasi tambahan. Kedua, penelitian Anda akan menggabungkan kemudahan penggunaan offline desktop application dengan tetap mempertahankan akurasi cosine similarity, menjawab celah dari jurnal keempat yang hanya menguji

algoritma tanpa membuat aplikasi, serta melengkapi jurnal kelima yang menggunakan algoritma berbeda. Ketiga, Anda berpotensi menambahkan fitur batch processing atau visualisasi hasil kemiripan (misalnya dalam bentuk grafik batang) yang belum banyak dibahas pada jurnal-jurnal sebelumnya. Keempat, penelitian Anda dapat mengimplementasikan preprocessing yang lebih lengkap (case folding, tokenizing, stopword removal, stemming dengan Sastrawi) dalam satu paket aplikasi yang user-friendly.

2.2 Plagiarisme

Plagiarisme didefinisikan sebagai tindakan mengambil, menyalin, atau menggunakan gagasan, kata-kata, maupun karya orang lain dan menyajikannya sebagai karya sendiri tanpa memberikan pengakuan atau atribusi yang sesuai terhadap sumber aslinya. Dalam konteks akademik, plagiarisme merupakan pelanggaran serius terhadap etika dan integritas keilmuan karena merusak nilai kejujuran intelektual yang menjadi fondasi utama dunia pendidikan tinggi (Humaidi *et al.*, 2026).

Plagiarisme dapat terjadi dalam berbagai bentuk, mulai dari penyalinan kata demi kata (copy-paste) tanpa kutipan, parafrase tanpa mencantumkan sumber, hingga penggunaan gagasan orang lain tanpa izin maupun pengakuan. Maraknya plagiarisme di lingkungan akademik turut dipengaruhi oleh kemudahan akses terhadap sumber digital seperti jurnal daring, repositori skripsi, dan artikel ilmiah yang dapat diakses secara bebas, sehingga memicu perilaku salin-tempel yang tidak bertanggung jawab (Pratiwi and Niki, 2021).

Dampak plagiarisme tidak hanya merugikan institusi pendidikan dari sisi reputasi dan kredibilitas lulusan, tetapi juga merugikan individu pelaku secara langsung. Mahasiswa yang terbukti melakukan plagiarisme berisiko kehilangan kepercayaan dari pembimbing maupun komite penguji, serta dapat menghambat peluang kolaborasi penelitian dan jenjang karier akademik di masa mendatang. Oleh karena itu, dibutuhkan suatu mekanisme deteksi yang mampu mengidentifikasi indikasi plagiarisme secara cepat, akurat, dan dapat diakses oleh seluruh civitas akademika tanpa terkendala faktor biaya maupun infrastruktur.

2.3 Text Mining

Text mining merupakan proses penggalian informasi dan pola tersembunyi dari data tekstual yang tidak terstruktur menjadi bentuk yang lebih terstruktur sehingga dapat dianalisis secara komputasional. Salah satu penerapan utama text mining adalah dalam pengukuran kemiripan antar dokumen, yang menjadi dasar bagi berbagai aplikasi seperti pendeteksi plagiarisme, mesin pencari, dan sistem rekomendasi (Haris, Baihaqi and Jananto, 2026).

2.4 Text Preprocessing

Text preprocessing adalah rangkaian tahapan yang dilakukan untuk mengubah teks mentah menjadi bentuk yang lebih bersih dan terstandarisasi sebelum diproses lebih lanjut oleh algoritma (Asriyani Arsad, Mustamin Hamid and M Santosa, 2024). Tahapan preprocessing yang umum digunakan dalam sistem deteksi plagiarisme antara lain:

1. Case Folding proses penyeragaman seluruh karakter dalam teks menjadi huruf kecil (lowercase) untuk menghindari perbedaan representasi kata akibat penggunaan huruf besar dan kecil yang tidak konsisten.
2. Tokenization proses pemecahan teks menjadi satuan-satuan kata (token) sebagai unit dasar dalam analisis lebih lanjut.
3. Stopword Removal proses penghapusan kata-kata umum yang tidak memiliki makna signifikan dalam analisis kemiripan, seperti "dan", "yang", "di", "ke", dan "dari" untuk Bahasa Indonesia.
4. Stemming proses pengubahan kata berimbuhan menjadi bentuk kata dasarnya. Untuk Bahasa Indonesia, proses stemming umumnya dilakukan menggunakan algoritma Sastrawi, sehingga variasi kata seperti "mendeteksi", "pendeteksian", dan "terdeteksi" akan diseragamkan menjadi bentuk dasar "deteksi".

Keempat tahapan tersebut dilaksanakan secara berurutan agar data teks yang dihasilkan benar-benar bersih dari elemen-elemen yang dapat mengganggu akurasi perhitungan kemiripan dokumen pada tahap selanjutnya.

2.5 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF (Term Frequency-Inverse Document Frequency) merupakan metode pembobotan kata yang digunakan untuk mengukur seberapa penting suatu kata terhadap sebuah dokumen dalam suatu kumpulan atau korpus dokumen. Metode ini menggabungkan dua komponen utama, yaitu Term Frequency (TF) dan Inverse Document Frequency (IDF), yang masing-masing memiliki peran berbeda dalam menentukan bobot akhir sebuah kata (Maharani *et al.*, 2025).

2.5.1 Term Frequency (TF)

Term Frequency (TF) mengukur frekuensi kemunculan sebuah kata (term) di dalam suatu dokumen. Semakin sering sebuah kata muncul dalam dokumen, semakin besar pula nilai TF kata tersebut. Nilai TF dihitung menggunakan persamaan berikut (Fajri, Rizaldy and Abidin, 2026):

$$TF\ (t,d) = \frac{f(t,d)}{\sum f\ (t,d)}$$

Keterangan: f(t,d) adalah frekuensi kemunculan term t pada dokumen d, dan Σf(t',d) adalah total keseluruhan term yang terdapat pada dokumen d. Nilai TF dinormalisasi terhadap panjang dokumen untuk mencegah bias terhadap dokumen yang lebih panjang.

2.5.2 Inverse Document Frequency (IDF)

Inverse Document Frequency (IDF) mengukur seberapa jarang sebuah kata muncul di seluruh koleksi dokumen. Kata yang muncul di hampir semua dokumen (seperti kata umum) akan memiliki nilai IDF yang rendah, sedangkan kata yang hanya muncul pada sejumlah kecil dokumen akan memiliki nilai IDF yang tinggi karena dianggap lebih spesifik dan informatif. Nilai IDF dihitung menggunakan persamaan berikut (Fajri, Rizaldy and Abidin, 2026):

$$IDF\ (t,D) = \log\left(\frac{N}{df(t)}\right)$$

Keterangan: N adalah jumlah total dokumen dalam korpus, dan df(t) adalah jumlah dokumen yang mengandung term t. Untuk menghindari pembagian dengan nol pada kata yang tidak muncul di korpus pelatihan, umumnya diterapkan teknik smoothing dengan menambahkan nilai konstan pada df(t).

2.5.3 Pembobotan TF-IDF

Nilai akhir bobot TF-IDF suatu kata diperoleh dari hasil perkalian antara nilai TF dan IDF, sebagaimana dirumuskan pada persamaan berikut (Wong, Yoanita and et all, 2026):

$$W(t,d) = TF(t,d)X\ IDF(t,D)$$

Hasil perhitungan TF-IDF ini merepresentasikan setiap dokumen sebagai sebuah vektor numerik dalam ruang multidimensi, di mana setiap dimensi mewakili satu kata unik dalam korpus dan nilainya merupakan bobot TF-IDF kata tersebut pada dokumen yang bersangkutan. Representasi vektor inilah yang selanjutnya digunakan sebagai dasar perhitungan kemiripan antar dokumen menggunakan metode Cosine Similarity

2.6 Cosine Similarity

Cosine Similarity adalah metode pengukuran kemiripan antara dua vektor dengan cara menghitung nilai kosinus sudut yang terbentuk di antara keduanya dalam ruang vektor multidimensi. Metode ini banyak digunakan dalam bidang information retrieval dan text mining karena efektivitasnya dalam mengukur kemiripan dokumen tanpa terpengaruh oleh panjang dokumen (Hasibuan, Simangunsong and et all, 2023).

Nilai Cosine Similarity dihitung dengan membagi hasil perkalian titik (dot product) antara dua vektor dengan hasil perkalian magnitudo (panjang) dari masing-masing vektor tersebut, sebagaimana dirumuskan pada persamaan berikut:

$$cos(\theta) = (A \cdot B) / (||A|| \times ||B||)$$

Keterangan: A dan B adalah vektor representasi dari dua dokumen yang dibandingkan, A · B adalah hasil dot product antara vektor A dan B, sedangkan ||A|| dan ||B|| adalah magnitudo (Euclidean norm) dari masing-masing vektor. Nilai dot product dan magnitudo masing-masing dihitung menggunakan persamaan berikut:

$$A \cdot B = \Sigma(Ai \times Bi) \qquad ||A|| = \sqrt{(\Sigma Ai^2)}$$

Hasil perhitungan Cosine Similarity menghasilkan nilai dalam rentang 0 hingga 1, di mana nilai 0 menunjukkan bahwa kedua dokumen tidak memiliki kesamaan kata sama sekali (tidak mirip), sedangkan nilai 1 menunjukkan bahwa kedua dokumen identik sepenuhnya. Semakin mendekati nilai 1, semakin tinggi tingkat kemiripan antara kedua dokumen yang dibandingkan, dan semakin besar pula indikasi terjadinya plagiarisme.

Nilai kemiripan yang diperoleh kemudian dikonversi ke dalam bentuk persentase dan dikelompokkan ke dalam beberapa kategori tingkat plagiarisme guna memudahkan interpretasi hasil oleh pengguna. Kategorisasi yang digunakan dalam penelitian ini disajikan pada Tabel 2.1 berikut.

Tabel 2.1 Kategori Tingkat Plagiarisme Berdasarkan Skor Cosine Similarity

Rentang Skor	Persentase	Kategori	Indikator Warna
≥ 0,80	≥ 80%	Plagiarisme Tinggi	Merah
0,60 – 0,79	60% – 79%	Plagiarisme Sedang	Oranye
0,40 – 0,59	40% – 59%	Plagiarisme Rendah	Kuning
< 0,40	< 40%	Tidak Plagiat	Hijau

Kategorisasi pada Tabel 2.1 disusun berdasarkan kombinasi referensi ambang batas yang digunakan oleh penelitian-penelitian terdahulu, seperti pengelompokan kemiripan ringan (<30%), sedang (30%–70%), dan berat (>70%),

dengan penyesuaian agar lebih sesuai dengan kebutuhan sistem yang dikembangkan pada penelitian ini.

2.7 Graphical User Interface (GUI) dan Python Tkinter

Graphical User Interface (GUI) adalah antarmuka visual yang memungkinkan pengguna berinteraksi dengan sistem komputer melalui elemen-elemen grafis seperti tombol, kotak teks, menu, dan jendela, tanpa harus mengetikkan perintah berbasis teks (command-line). Penggunaan GUI bertujuan untuk meningkatkan kemudahan penggunaan (usability) suatu aplikasi, khususnya bagi pengguna awam yang tidak terbiasa dengan antarmuka berbasis perintah.

Tkinter merupakan pustaka (library) standar bawaan Python yang digunakan untuk membangun aplikasi GUI berbasis desktop. Sebagai pustaka standar, Tkinter memiliki keunggulan utama berupa kemudahan proses instalasi dan distribusi aplikasi, karena tidak memerlukan instalasi dependensi tambahan di luar instalasi Python itu sendiri. Karakteristik inilah yang menjadikan Tkinter pilihan tepat untuk membangun aplikasi deteksi plagiarisme yang dirancang untuk beroperasi secara offline dan dapat didistribusikan dengan mudah ke berbagai perangkat (Novendra Sulu *et al.*, 2024).

Beberapa komponen (widget) utama Tkinter yang dimanfaatkan dalam pengembangan sistem pada penelitian ini meliputi Frame sebagai wadah penampung elemen-elemen antarmuka, Label untuk menampilkan teks maupun informasi statis, Button untuk menjalankan suatu aksi atau fungsi tertentu ketika diklik, Text dan ScrolledText untuk menampilkan maupun menerima input teks dalam jumlah besar, Treeview untuk menampilkan data dalam format tabel, serta

Notebook untuk mengelola tampilan antarmuka berbasis tab. Kombinasi widget-widget tersebut memungkinkan pembangunan antarmuka yang informatif dan intuitif bagi pengguna.

2.8 Model Waterfall

Model Waterfall merupakan salah satu model pengembangan perangkat lunak yang bersifat sekuensial dan linear, di mana setiap fase pengembangan harus diselesaikan terlebih dahulu sebelum dilanjutkan ke fase berikutnya. Model ini sangat cocok diterapkan pada proyek perangkat lunak yang kebutuhan sistemnya sudah terdefinisi dengan jelas dan tidak banyak mengalami perubahan di tengah proses pengembangan (Ayunita Pertiwi *et al.*, 2023).

Model Waterfall umumnya terdiri dari lima fase utama, yaitu: (1) Requirements Analysis, tahap identifikasi dan analisis kebutuhan sistem baik fungsional maupun non-fungsional; (2) System Design, tahap perancangan arsitektur sistem, antarmuka, dan struktur data; (3) Implementation, tahap penerjemahan hasil rancangan ke dalam bentuk kode program; (4) Testing, tahap pengujian sistem untuk memastikan seluruh fungsi berjalan sesuai dengan spesifikasi yang ditetapkan; dan (5) Maintenance, tahap pemeliharaan serta perbaikan sistem pasca implementasi. Kelima fase tersebut menjadi acuan dalam penyusunan kerangka penelitian.

2.9 Pengujian Black Box

Black Box Testing merupakan salah satu metode pengujian perangkat lunak yang berfokus pada pengujian fungsionalitas sistem berdasarkan spesifikasi kebutuhan, tanpa memperhatikan struktur kode atau alur logika program di

dalamnya. Pengujian ini dilakukan dengan memberikan sejumlah data masukan (input) pada sistem, kemudian membandingkan hasil keluaran (output) yang dihasilkan dengan hasil yang diharapkan (Fatimah, Paryatin and Nurhasanah, 2022).

Metode pengujian yang umum digunakan dalam Black Box Testing antara lain Equivalence Partitioning, yaitu pembagian data uji ke dalam kelompok-kelompok yang memiliki perlakuan serupa, dan Boundary Value Analysis, yaitu pengujian terhadap nilai-nilai pada batas kondisi (boundary) suatu variabel. Hasil pengujian Black Box dinyatakan dalam kategori PASS apabila output aktual sistem sesuai dengan output yang diharapkan, dan FAIL apabila terjadi ketidaksesuaian antara keduanya.

2.10 System Usability Scale (SUS)

System Usability Scale (SUS) adalah instrumen kuesioner standar yang dikembangkan oleh John Brooke pada tahun 1986 untuk mengukur tingkat kegunaan (usability) suatu sistem atau produk secara subjektif berdasarkan persepsi pengguna. SUS banyak digunakan dalam penelitian rekayasa perangkat lunak karena sifatnya yang sederhana, cepat diisi, serta telah teruji validitas dan reliabilitasnya secara luas (Taufik Agus Haryanto and Eka Rini Yulia, 2025).

Kuesioner SUS terdiri dari 10 butir pernyataan dengan skala Likert 1 sampai 5, di mana 1 berarti "Sangat Tidak Setuju" dan 5 berarti "Sangat Setuju". Sepuluh pernyataan tersebut terdiri dari pernyataan bernada positif pada nomor ganjil (1, 3, 5, 7, 9) dan pernyataan bernada negatif pada nomor genap (2, 4, 6, 8, 10), sebagaimana ditampilkan pada Tabel 2.2 berikut.

Tabel 2.2 Daftar Pernyataan Kuesioner System Usability Scale (SUS)

No	Pernyataan	Jenis
1	Saya pikir saya akan sering menggunakan sistem ini.	Positif
2	Saya merasa sistem ini terlalu kompleks, padahal tidak perlu.	Negatif
3	Saya pikir sistem ini mudah digunakan.	Positif
4	Saya membutuhkan bantuan orang teknis untuk menggunakan sistem ini.	Negatif
5	Saya merasa berbagai fungsi dalam sistem ini terintegrasi dengan baik.	Positif
6	Saya merasa terlalu banyak ketidakkonsistenan dalam sistem ini.	Negatif
7	Saya bayangkan kebanyakan orang belajar menggunakan sistem ini sangat cepat.	Positif
8	Saya merasa sistem ini sangat rumit untuk digunakan.	Negatif
9	Saya merasa sangat percaya diri menggunakan sistem ini.	Positif
10	Saya perlu belajar banyak hal sebelum bisa menggunakan sistem ini.	Negatif

Skor SUS dihitung melalui dua tahap. Tahap pertama adalah menghitung nilai kontribusi tiap butir pernyataan, dengan ketentuan: untuk pernyataan bernomor ganjil (positif), nilai kontribusi diperoleh dari skor jawaban dikurangi 1; sedangkan untuk pernyataan bernomor genap (negatif), nilai kontribusi diperoleh dari 5 dikurangi skor jawaban. Tahap kedua adalah menjumlahkan seluruh nilai kontribusi dari 10 pernyataan, kemudian mengalikannya dengan konstanta 2,5 untuk memperoleh skor akhir dalam skala 0 sampai 100, sebagaimana dirumuskan pada persamaan berikut:

$$Skor\ SUS = (\Sigma\ Kontribusi\ Pernyataan) \times 2{,}5$$

Skor akhir SUS yang diperoleh kemudian diinterpretasikan ke dalam beberapa kategori tingkat usability sebagaimana disajikan pada Tabel 2.3 berikut.

Tabel 2.3 Interpretasi Skor System Usability Scale (SUS)

Rentang Skor	Kategori	Deskripsi
≥ 85	Excellent	Sistem sangat mudah digunakan tanpa perlu pelatihan khusus
71 – 84	Good	Sistem mudah digunakan dengan sedikit kendala
52 – 70	OK / Acceptable	Sistem cukup dapat diterima namun masih perlu perbaikan
39 – 51	Poor	Sistem memiliki kendala usability yang cukup signifikan
< 39	Awful	Sistem sangat sulit digunakan dan perlu perancangan ulang

.