BAB 2
LANDASAN TEORI

2.1 Tinjauan Teori

2.1.1 *Expert System*

Teknologi *Artificial Intelligence* (AI) memiliki cakupan yang luas seperti rule-based *expert system*, *frame-based expert system*, *fuzzy logic, neural network, genetic algorithm*, dan lainnya. Salah satu dari teknologi AI yaitu *expert system* dibuat dengan meng-*extract* pengetahuan dari seorang atau sekelompok ahli yang kemudian diaplikasikan ke sebuah program komputer agar pengetahuan tersebut dapat diproses [10]. Pembuatan *expert system* dikategorikan berdasarkan bidang apa *expert system* tersebut dibuat karena *expert system* tersebut memiliki rule, kode, algoritma, bahkan cara iteraksi dengan user yang berbeda [10]. Fungsi utama dari *Expert system* adalah untuk melakukan diagnosa, prediksi, atau membantu pemilhan desisi agar kekurangan tenaga kerja dapat diatasi dengan mudah dan dapat membantu ahli dibidang dimana *expert system* tersebut dibuat.

Dalam *expert system* terdapat sebuah *inference engine* yang merupakan bagian dari *expert system* yang berisikan mekanisme untuk berpikir dan nalar yang digunakan oleh *expert system* [10]. Teknik *inference engine* dalam pembuatan *expert system* adalah *Forward Chaining* dan *Backward Chaining*. Teknik *forward chaining* memulai penalarannya dari data yang diberikan untuk menarik kesimpulan akhir, sedangkan *backward chaining* memulai penalarannya dari hipotesis yang akan ditarik menjadi kesimpulan yang mengandung hipotesis tersebut.

2.1.2 MCDM

Multi-Criteria Decision Making (MCDM) adalah sebuah metode pemilihan desisi yang digunakan ketika terdapat beberapa atribut yang perlu dipertimbangkan untuk menyusun urutan prioritas dari desisi-desisi yang akan dipilih yang terdiri atas berbagai alternatif [11]. Metode MCDM ini digunakan ketika seorang individu maupun organisasi kesulitan atau bingung untuk memilih suatu disisi dan alternatif-alternatifnya, sehingga dengan digunakannya metode MCDM proses pemilihan tersebut dapat dilakukan dengan lebih mudah dan menghemat waktu. Secara umum, MCDM dimaksudkan untuk mengurangi pembuatan desisi yang

didasarkan oleh firasat dan juga kegagalan pengambilan keputusan kelompok yang hampir pasti menimpa pendekatan intuitif. Dengan membuat bobot dan *trade-off* terkait antara kriteria secara eksplisit dengan cara yang terstruktur, MCDM menghasilkan pengambilan keputusan yang lebih baik. Sehingga dengan adanya MCDM pemilihan desisi lebih dimudahkan. Terdapat berbagai macam teknik MCDM yang telah digunakan dalam pembuatan *Decision Support System* (DSS) seperti AHP, ANP, TOPSIS, VIKOR, SAW, ELECTRE, PROMETHEE, MAUT dan MPE [12].

A Perbandingan MCDM

Berikut perbandingan antara beberapa metode MCDM2.1

Tabel 2.1. Perbandingan MCDM [1]

Metode	Keuntungan	Kerugian	Area Aplikasi
***Simple Additive Weighting* (SAW)**	Kemampuan untuk mengkompensasi di antara kriteria; Intuitif bagi para pengambil keputusan; Perhitungannya sederhana dan tidak memerlukan program komputer yang kompleks.	Estimasi tidak selalu mencerminkan situasi sebenarnya; Hasil yang diperoleh mungkin tidak logis.	Manajemen air, bisnis, dan manajemen keuangan.
Lanjut pada halaman berikutnya			

Tabel 2.1 Perbandingan Penelitian Serupa (lanjutan) [1]

Metode	Keuntungan	Kerugian	Area Aplikasi
ELECTRE	Mengambil ketidakpastian dan kekaburan menjadi pertimbangan.	Tidak menyediakan metode yang jelas untuk menetapkan bobot.	Lingkungan, hidrologi, manajemen air, bisnis dan keuangan, kimia, logistik dan transportasi, manufaktur dan perakitan, energi, pertanian.
TOPSIS	Memiliki proses yang sederhana; mudah digunakan dan diprogram; Jumlah langkah tetap sama terlepas dari jumlah atribut.	Penggunaan Jarak *Euclidean* tidak mempertimbangkan korelasi atribut	Manajemen rantai pasokan dan logistik, *engineering*, sistem manufaktur, bisnis dan pemasaran, lingkungan, sumber daya manusia, dan manajemen sumber daya air.

Berdasarkan tabel perbandingan tersebut, TOPSIS dipilih sebagai MCDM yang digunakan dalam perancangan DSS karena TOPSIS mudah untuk digunakan dan diproses dalam perancangan sistem, lalu meskipun tidak mempertimbangkan korelasi antar atribut. Pada DSS yang dirancang, pengguna dapat mengatur bobot kriteria sesuai dengan preferensi mereka. TOPSIS juga cocok untuk digunakan dalam lingkup bisnis dan pemasaran dikarenakan aspek-aspek yang dipertimbangkan dalam tempat penginapan dijadikan bahan pemasaran, baik itu dari harga, servis, dan lokasi tempat penginapan tersebut. Oleh karena itu, TOPSIS menjadi pilihan dalam perancangan DSS tempat penginapan.

2.1.3 TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution atau dikenal dengan TOPSIS adalah metode analisis keputusan multi-kriteria. TOPSIS

membandingkan serangkaian alternatif berdasarkan kriteria yang telah ditentukan sebelumnya [13]. TOPSIS menggunakan prinsip bahwa alternatif yang terpilih harus mempunyai jarak terdekat dari solusi ideal positif dan terjauh dari solusi ideal negatif dari sudut pandang geometris dengan menggunakan jarak *Euclidean* untuk menentukan kedekatan relatif dari suatu alternatif dengan solusi optimal [14]. Dengan mempertimbangkan baik jarak terhadap solusi ideal positif dan jarak terhadap solusi ideal negatif dengan mengambil kedekatan relatif terhadap solusi ideal positif, urutan ranking dari alternatif yang ada dapat dihasilkan menggunakan TOPSIS. Dalam TOPSIS data yang digunakan terdiri atas *m* alternatif, *A1, . . . , Am*, dan *n* kriteria, *C1, . . . , Cn* beserta apakah kriteria tersebut termasuk dalam *benefit* atau *cost*. Setiap alternatif kemudian dievaluasi sesuai dengan nilai dari kriteria mereka [15]. Evaluasi ini membentuk sebuah decision matrix $X = (x_{ij})_{(m*n)}$. Vektor dari bobot setiap kriteria juga dibentuk *W = (W1, . . . , Wn)*, dengan nilai dari setiap bobot ditentukan sesuai besar pengaruh kriteria tersebut. Tahapan dalam metode TOPSIS sebagai berikut:

1. Menyusun *Normalized Decision Matrix*

   Pada tahap ini, dilakukan transformasi dari *various attribute dimensions* menjadi *non-dimensional attributes*, sehingga perbandingan antar kriteria dapat dilakukan [15]. Dalam *normalized decision matrix*, nilai normalisasi dari setiap x_{ij} dihitung dengan rumus

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}, \quad i = 1,\ldots,m, \quad j = 1,\ldots,n \tag{2.1}$$

   Hasil dari setiap nilai normalisasi dibentuk kedalam matriks R seperti berikut

$$R = \begin{bmatrix} r_{11} & \ldots & r_{1j} \\ \ldots & \ldots & \ldots \\ r_{i1} & \ldots & r_{ij} \end{bmatrix}, i = 1,\ldots,m, j = 1,\ldots,n \tag{2.2}$$

2. Menyusun *weighted normalized decision matrix*

Pada tahap ini, setiap nilai r dikalikan dengan bobot dari kriteria masing-masing sehingga menghasilkan elemen untuk matriks baru [15]. Bobot yang digunakan adalah bobot yang sudah ditetapkan sebelumnya bersamaan dengan alternatif dan kriteria. Elemen tersebut didapat dengan rumus

$$v_{ij} = w_j r_{ij}, \quad i = 1, \cdots, m, \quad j = 1, \cdots, n \tag{2.3}$$

Hasil tersebut kemudian dibentuk kembali menjadi matriks V seperti berikut

$$V = \begin{bmatrix} v_{11} & \dots & v_{1j} \\ \dots & \dots & \dots \\ v_{i1} & \dots & v_{ij} \end{bmatrix}, i = 1, \dots, m, j = 1, \dots, n \tag{2.4}$$

3. Menentukan solusi ideal dan non-ideal

Pada tahap ini solusi ideal dan non-ideal perlu dihitung dan tidak ditetapkan secara konstan dari awal, hal ini agar DSS dapat menentukan jarak yang lebih kredibel antara kedua poin tersebut [15]. Rumus yang digunakan mencari solusi ideal adalah

$$A^* = \{v_1^*, v_2^*, \cdots, v_n^*\} = \left\{ \left(\max_j v_{ij} \mid i \in I'\right), \left(\min_j v_{ij} \mid i \in I''\right) \right\} \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n \tag{2.5}$$

Berdasarkan rumus solusi ideal, dapat dilihat bahwa solusi ideal dibentuk menggunakan seluruh performa terbaik normalized decision matrix [12]. Sedangkan rumus untuk solusi non-ideal adalah

$$A^- = \left\{v_1^-, v_2^-, \cdots, v_n^-\right\} = \left\{ \left(\min_j v_{ij} \mid i \in I'\right), \left(\max_j v_{ij} \mid i \in I''\right) \right\}, \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n \tag{2.6}$$

Berbanding terbalik dengan solusi ideal, elemen dari solusi non-ideal dibentuk menggunakan seluruh performa terburuk *normalized decision matrix*. *I'* diasosiasikan pada kriteria *benefit*, sedangkan *I"* diaosiasikan dengan kriteria *cost*.

4. Menghitung *separation measures* untuk setiap alternatif

Pada tahap ini dilakukan perhitungan jarak dari setiap alternatif terhadap baik solusi ideal maupun solusi non-ideal. Rumus untuk menghitung solusi ideal adalah

$$D_i^* = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^*\right)^2}, \quad i = 1,2,\cdots,m, \quad j = 1,2,\cdots,n \tag{2.7}$$

Sedangkan rumus untuk menghitung solusi non-ideal adalah

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}, \quad i = 1,2,\cdots,m, \quad j = 1,2,\cdots,n \tag{2.8}$$

Kedua rumus tersebut adalah rumus yang paling popular yaitu menggunakan *Eucladian distance*, terdapat beberapa cara yang lain yaitu *Hamming distance* dan *Chebyshev distance* [15]. Rumus yang digunakan dalam step ini adalah yang paling popular dan umum digunakan yaitu *Eucladian distance*.

5. Menghitung *relative closeness* terhadap solusi ideal

   Pada tahap ini dilakukan perhitungan relative closeness dengan hasil pasti diantara 0 dan 1, serta hasil terbaik adalah hasil yang paling mendekati 1 [15]. Perhitungan dilakukan menggunakan hasil solusi ideal dan non-ideal dari setiap alternatif dengan rumus

$$C_i^* = \frac{D_i^-}{D_i^* + D_i^-}, \quad i = 1,2,\cdots,m \tag{2.9}$$

6. Menyusun *ranking* dari *preference order*

   Pada tahap ini dilakukan pengurutan hasil *relative closeness*, dan alternatif yang memiliki *relative closeness* tertinggi menjadi alternatif terbaik.

2.1.4 Web Storage

Web Storage mengacu pada mekanisme penyimpanan data di sisi klien *HTML5*. *Web Storage* menyimpan data dalam bentuk pasangan *key-value*. Walaupun *web storage* dan *cookies* menyimpan data di sisi klien, namun keduanya bekerja dengan cara yang berbeda. *Cookies* dikirimkan antara *client* dan *server* dengan setiap permintaan dari *web*. Sedangkan *Web Storage* tidak pernah secara otomatis dikirimkan ke *server*. *Web Storage* juga tidak memiliki waktu kadaluarsa

seperti *cookies*. *Web Storage* harus dihapus secara koding atau dihapus oleh *user* melalui *browser* masing-masing. *Web Storage* dibagi menjadi dua, yaitu *session storage* dan *local storage*. Data *Session storage* hanya disimpan selama *tab* masih terbuka, dan ketika *browser* atau *tab* ditutup maka datanya akan dihapus. Sedangkan data *local storage* masih tersimpan ketika *browser* maupun *tab* ditutup sehingga data masih ada sampai data dihapus oleh pengguna maupun secara koding. *Web storage* tersedia untuk hampir seluruh *browser*, hanya sedikit *browser* yang tidak menyediakan fitur *web storage* seperti *browser* versi *lite*. [16].

2.1.5 Merge Sort

Sorting merujuk pada operasi penyusunan data dalam urutan baik itu dari kecil ke besar (*ascending*) ataupun dari besar ke kecil (*descending*)[17]. Algoritma *sorting* digunakan untuk mengurutkan data dengan posisi yang acak kebentuk yang lebih urut baik itu *ascending* atau *descending* agar lebih mudah untuk dilihat ataupun diolah. Sebagai contoh, terdapat sebuah daftar elemen A sebanyak n $(A_1, A_2, A_3, ..., A_n)$. Proses pengurutan daftar elemen A secara *ascending* akan membentuk daftar elemen A menjadi bentuk seperti berikut [17],

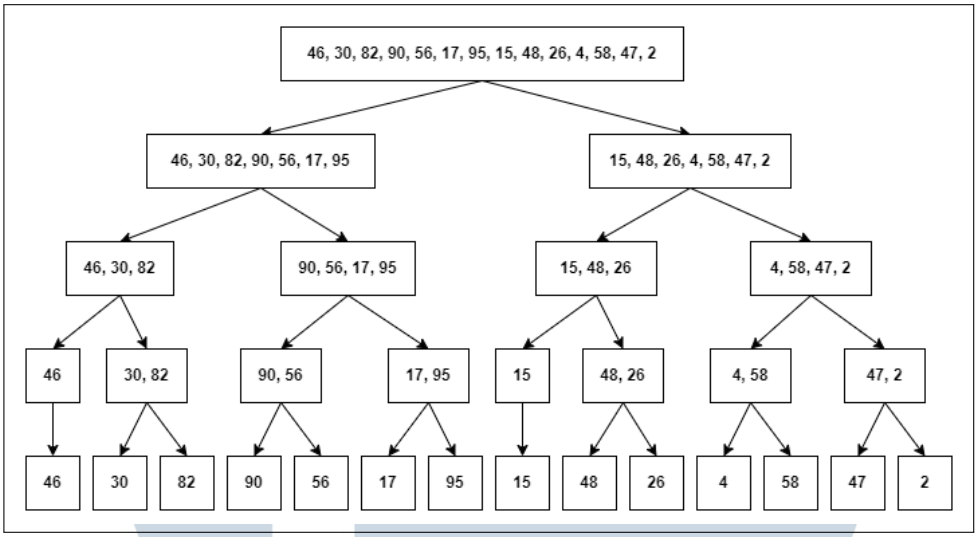
$$A_1 \leq A_2 \leq A_3 \leq ... \leq A_n \tag{2.10}$$

Merge sort merupakan salah satu algoritma *sorting* yang menggunakan teknik *divide and conquer*. *Merge sort* menggunakan teknik fungsi rekursif untuk mengurutkan data. *Merge sort* baik untuk skenario terbaik maupun terburuk memiliki kompleksitas algoritma $O(n \log_2 n)$, sehingga *merge sort* cocok baik untuk dataset kecil maupun besar. Daftar data akan dibagi secara rata dan disimpan secara sementara sebagai variabel tersendiri, proses membagi ini dilakukan terus menerus hingga tidak dapat dilakukan lagi. Data yang telah dibagi-bagi tersebut, kemudian masuk kedalam proses *merge* dengan membandingkan setiap setiap data yang telah dibagi baik secara *ascending* atau *descending* lalu digabungkan. Proses *merge* dilakukan hingga seluruh data yang telah terbagi kembali menjadi satu daftar yang telah terurut [17]. Sebagai contoh, terdapat sebuah daftar angka yang ingin diurutkan secara *ascending* seperti berikut,

$$46, 30, 82, 90, 56, 17, 95, 15, 48, 26, 4, 58, 47, 2 \tag{2.11}$$

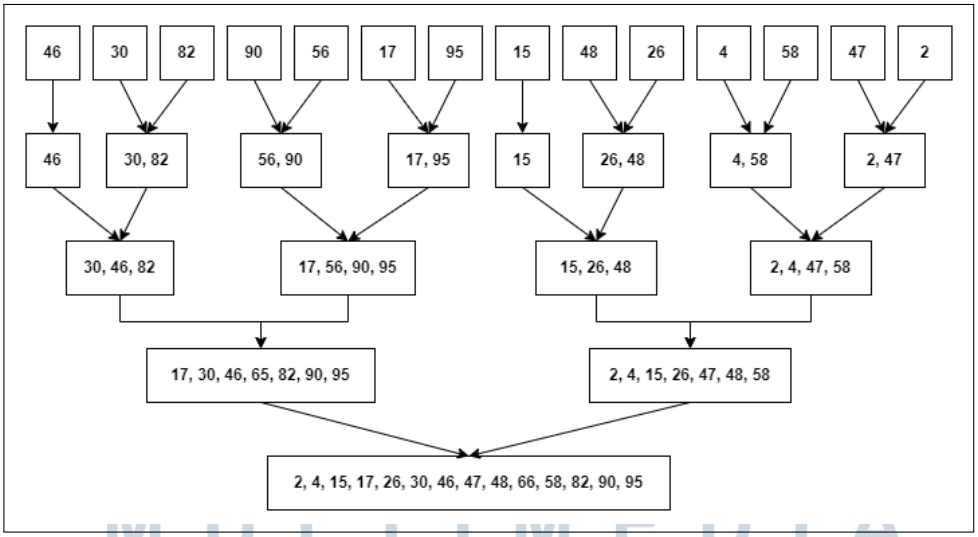
Daftar angka tersebut kemudian dipecah secara rata hingga bagian terkecil

seperti pada Gambar 2.1



Gambar 2.1. Proses *split* dalam *Merge sort*

Data yang telah dibagi-bagi tersebut kemudian masuk kedalam proses *merge*. Pada proses ini data dibandingkan untuk diurutkan secara *ascending* pada saat proses penggabungan. Data yang telah melalui proses *merge* kembali menjadi satu daftar yang telah terurut seperti pada Gambar 2.2



Gambar 2.2. Proses *merge* dalam *Merge sort*

2.1.6 Skala Likert

Skala likert merupakan skala pengukuran yang lahir pada tahun 1932 oleh Renis Likert. Pengukurang skala likert menggunakan empat atau lebih butir

pertanyaan yang mempresentasikan sebuah kondisi atau jawaban untuk membentuk skor atau nilai [18]. Skala likert dikembangkan menggunakan lima titik respon yaitu dari sangat tidak setuju, tidak setuju, netral, setuju, dan sangat setuju, dengan jumlah butir yang diatur sesuai peneliti. Skala likert tidak mengharuskan menggunakan nilai seperti ”setuju” melainkan dapat juga menggunakan nilai seperti ”baik” atau ”relevan”, tergantung kebutuhan peneliti. Setiap pertanyaan akan memiliki nilai jawaban dalam jarak satu hingga lima, tergantung dari pilihan yang dipilih oleh responden. Hasil dari penilaian menggunakan skala likert kemudian dapat diolah menggunakan formula sebagai berikut [18]:

1. Menghitung nilai masing-masing pertanyaan

$$Score = (P1 * 1) + (P2 * 2) + (P3 * 3) + (P4 * 4) + (P5 * 5) \tag{2.12}$$

   Total dari nilai didapat dari mengkalikan jumlah responden yang menjawab nilai dari yang paling rendah (P1) hingga yang paling tinggi (P5), kemudian masing-masing jumlah jawaban dikalikan dengan nilainya masing-masing dari jarak satu hingga lima.

2. Menghitung skor dari setiap pertanyaan

$$Skor(N) = \frac{TotalSkor}{SkorMaksimum} * 100 \tag{2.13}$$

   Skor sebuah atau sekelompok pertanyaan dihitung dengan membagian total skor yang diperoleh terhadap skor maksimum yang dapat diperoleh. Hasilnya kemudian dikalikan dengan 100 untuk mendapatkan persentasi skor.

3. Menghitung rata-rata dari skor

$$Rata - rata = \frac{N1 + N2 + N3 + ... + Nx}{x} * 100 \tag{2.14}$$

   Rata-rata dihitung dengan menjumlahkan total skor yang diperoleh lalu dibagi oleh jumlah pertanyaan (x). Hasilnya kemudian dikalikan dengan 100 untuk mendapatkan persentasi skor.

Hasil dari kalkulasi kemudian dapat dievaluasi menggunakan skala interval sebagai berikut [18],

- 0% - 19.99% = Sangat Tidak (setuju / baik / relevan)
- 20% - 39.99% = Tidak (setuju / baik / relevan)

- 40% - 59.99% = Cukup / netral
- 60% - 79.99% = Setuju / baik / relevan
- 80% - 100% = Sangat (setuju / baik / relevan)

2.1.7 EUCS Questionnaire

End User Computing Satisfaction (EUCS) adalah alat pengukur tingkat kepuasan pengguna terhadap suatu sistem atau aplikasi dan hasil dari pengukuran EUCS dihitung secara statistik untuk mendapatkan hasilnya [19]. EUCS dalam pengukuran *website* menggunakan lima dimensi penilaian, yaitu konten, akurasi, format, kemudahan penggunaan, dan ketepatan waktu. Penjelasan dari kelima dimensi tersebut adalah [20],

- Konten, berkaitan dengan relevansi dan kelengkapan pada website.
- Akurasi, berkaitan dengan ketepatan dan kebenaran informasi pada website.
- Format, berkaitan dengan tampilan dari website.
- Kemudahan penggunaan, berkaitan dengan kemudahan pengguna dalam mengakses fitur website.
- Ketepatan waktu, berkaitan dengan performa, kecepatan waktu sistem, dan apakah mempercepat kebutuhan pengguna.

2.1.8 Uji Validitas

Validitas berasal dari kata *validity* yang berarti sejauh mana ketepatan suatu alat ukut dalam melakukan pengukurannya. Validitas juga merupakan ukuran yang menunjukkan bahwa variabel yang diukur merupakan variabel yang hendak diteliti oleh peneliti [21]. Uji validitas instrumen dilakukan terhadap hasil kuesioner terhadap penelitian untuk menunjukkan bahwa instrumen yang digunakan sudah tepat dan valid hasil yang diberikan [22]. Teknik pengujian untuk uji validitas instrumen yang sering digunakan adalah menggunakan teknik *Bivariate Pearson* (Produk Momen Pearson) [21]. Pengujian dilakukan dengan mencari korelasi masing-masing pertanyaan. Nilai total adalah penjumlahan dari nilai seluruh pertanyaan. Pertanyaan yang memiliki korelasi signifikan dengan nilai total

dikatakan mampu mengungkapkan apa yang ingin diungkapkan secara valid [21]. Jika $r_{hitung} \geq r_{tabel}$ dengan menggunakan uji 2 sisi dengan sig. 0.05, maka instrumen dikatakan valid. Rumus untuk menghitung korelasi dari pertanyaan adalah sebagai berikut,

$$r_{xy} = \frac{n\sum XY - (\sum X)(\sum Y)}{\sqrt{\left\{n\sum X^2 - (\sum X)^2\right\}\left\{n\sum Y^2 - (\sum Y)^2\right\}}} \tag{2.15}$$

Dengan keterangan:

- r_{xy} = koefisien korelasi
- n = jumlah responden
- X = nilai setiap pertanyaan
- Y = nilai seluruh pertanyaan responden

Bila menggunakan *spreadsheet* atau *excel*, maka korelasi dapat dihitung menggunakan *formula CORREL()* atau *PEARSON()* [22]. Setelah menghitung r_{hitung} dari seluruh pertanyaan, selanjutnya dibandingkan dengan r_{tabel}. R tabel (r_{tabel}) adalah acuan dalam ilmu statistik yang digunakan untuk melakukan uji validitas terhadap data penelitian [22]. R tabel adalah tabel berisi angka yang digunakan untuk menguji validitas data penelitian. Secara umum dalam pengujian dan buku statistika, kriteria pengujian yang digunakan adalah pengujian dua taraf dengan *degree of freedom* bernilai $(n-2)$ dan menggunakan sig. 0.05. Nilai dari R tabel seperti pada Gambar 2.3

df = (N-2)	Tingkat signifikansi untuk uji satu arah				
	0.05	0.025	0.01	0.005	0.0005
	Tingkat signifikansi untuk uji dua arah				
	0.1	0.05	0.02	0.01	0.001
1	0.9877	0.9969	0.9995	0.9999	1.0000
2	0.9000	0.9500	0.9800	0.9900	0.9990
3	0.8054	0.8783	0.9343	0.9587	0.9911
4	0.7293	0.8114	0.8822	0.9172	0.9741
5	0.6694	0.7545	0.8329	0.8745	0.9509
6	0.6215	0.7067	0.7887	0.8343	0.9249
7	0.5822	0.6664	0.7498	0.7977	0.8983
8	0.5494	0.6319	0.7155	0.7646	0.8721
9	0.5214	0.6021	0.6851	0.7348	0.8470
10	0.4973	0.5760	0.6581	0.7079	0.8233
11	0.4762	0.5529	0.6339	0.6835	0.8010
12	0.4575	0.5324	0.6120	0.6614	0.7800
13	0.4409	0.5140	0.5923	0.6411	0.7604
14	0.4259	0.4973	0.5742	0.6226	0.7419
15	0.4124	0.4821	0.5577	0.6055	0.7247
16	0.4000	0.4683	0.5425	0.5897	0.7084
17	0.3887	0.4555	0.5285	0.5751	0.6932
18	0.3783	0.4438	0.5155	0.5614	0.6788
19	0.3687	0.4329	0.5034	0.5487	0.6652
20	0.3598	0.4227	0.4921	0.5368	0.6524
21	0.3515	0.4132	0.4815	0.5256	0.6402
22	0.3438	0.4044	0.4716	0.5151	0.6287
23	0.3365	0.3961	0.4622	0.5052	0.6178
24	0.3297	0.3882	0.4534	0.4958	0.6074
25	0.3233	0.3809	0.4451	0.4869	0.5974
26	0.3172	0.3739	0.4372	0.4785	0.5880
27	0.3115	0.3673	0.4297	0.4705	0.5790
28	0.3061	0.3610	0.4226	0.4629	0.5703
29	0.3009	0.3550	0.4158	0.4556	0.5620
30	0.2960	0.3494	0.4093	0.4487	0.5541

Gambar 2.3. R tabel

2.1.9 TypeScript

JavaScript adalah bahasa pemrograman yang digunakan untuk pengembangan aplikasi *website* untuk membuat aplikasi *website* yang dinamis dan responsif. *TypeScript* merupakan ekstensi dari *JavaScript* yang lebih mendukung dalam pengembangan aplikasi skala besar. *TypeScript* memperkaya JavaScript dengan sistem modul, kelas, antarmuka, dan sistem tipe statis karena *TypeScript* bertujuan untuk memberikan bantuan kepada *programmers* dengan menyediakan sistem modul serta fleksibel dan mudah digunakan [23].

2.1.10 Angular

Angular adalah sebuah platform dan *framework open-source* yang dikembangkan oleh tim Angular di Google. Angular merupakan *framework JavaScript* untuk membuat aplikasi dan menyediakan berbagai fitur seperti *component*, *directives*, *data binding*, *form*, *service*, dan *dependency injection*. Aplikasi Angular ditulis menggunakan *TypeScript* yang merupakan perluasan dari JavaScript menyediakan fitur *object oriented programming*, seperti *class, interface,* dan *data type*. Kode *TypeScript* Angular kemudian dikompilasi menjadi *JavaScript* murni sehingga dapat digunakan disemua *browser*. Aplikasi Angular bersifat *single page application*, sehingga ketika *website* dibuka seluruh komponen *website* dimuat dan dapat mengganti konten halaman secara dinamis tanpa perlu memuat ulang *website* [24]. Beberapa fitur kunci angular adalah,

- *Framework* berbasis komponen
  Angular menggunakan pendekatan berbasis komponen, yang berarti aplikasi dibangun dengan komponen-komponen mandiri, sehingga mempermudah pengelolaan, organisir, dan pemeliharaan aplikasi.
- *Data binding*
  Angular menyediakan fitur *two-way binding*, sehingga perubahan pada sistem dapat langsung memberikan output atau perubahan pada tampilan layar.
- *Dependency injection*
  Angular memiliki sistem *dependency injection* yang memungkinkan komponen-komponen dalam aplikasi untuk bergantung pada layanan-layanan lain dengan mudah. Hal ini membantu dalam membuat aplikasi yang terstruktur, mudah diuji, dan mudah diubah.

- Dukungan untuk *TypeScript*
  Angular dikembangkan menggunakan *TypeScript*, sehingga memberikan banyak keuntungan seperti tipe statis, pemahaman kode yang lebih baik, dan alat bantu pengembangan yang lebih baik.

2.1.11 NoSQL

Database NoSQL merupakan *Database Management Software* yang berisfat *open-source* yang menudukung penyimpanan data besar di server. NoSQL mendukung SQL atau kueri mirip SQL, sehingga NoSQL bukan berarti tidak dapat menggunakan SQL, melainkan NoSQL berarti "*Not Only SQL*", karena NoSQL menggabungkan banyak fitur tambahan selain fitur basis data konvensional. *Database* NoSQL dirancang untuk mengatasi kebutuhan kinerja dan skalabilitas aplikasi web. *Rational database* perlu mengikuti peraturan sesuai struktur datanya ketika menyimpan data, sedangkan NoSQL bersifat *schema-less*, sehingga dapat menyimpan data tanpa terhalangi oleh struktur data dan memilih skalabilitas yang tinggi. [25]. Motivasi umum dalam desain *database* NoSQL adalah,

- *Deployment* yang mudah
- Data dapat dalam skala besar
- Memenuhi skalabilitas

2.1.12 Firebase

Firebase adalah platform *development* yang dikembangkan oleh Google dari tahun 2016 dengan tujuan untuk memberikan alat dan infrastruktur untuk membuat aplikasi. Firebase menyediakan berbagai macam servis, diantaranya adalah *database* [26]. *Cloud Firestore* merupakan salah satu servis yang diberikan oleh Firebase Google. *Cloud Firestore* adalah *database* NoSQL milik Firebase Google yang fleksibel dan memiliki skalabilitas. *Database* NoSQL *Cloud Firestore* memungkinkan fitur *query*, pembaruan secara *real-time*, dukungan aplikasi web maupun mobile, serta melalui skalabilitasnya dapat meng-*handle* aplikasi yang terus berkembang dengan data yang banyak tanpa perlu dijaga secara manual [26]. Keuntungan dari penggunaan *Cloud Firestore* milik Firebase adalah sebagai berikut [26],

- *Real-time Synchronization*
  Meng-*update* data secara *real time* sehingga cocok untuk aplikasi yang memerlukan perubahan data secara langsung.
- Skalabilitas
  Dapat menangani jumlah data besar dan *traffic* data tanpa perlu *scaling* secara manual atau melakukan manajemen *server*.
- Fleksibel
  Memiliki struktur data yang fleksibel, sehingga lebih mudah untuk mengelola hubungan data yang kompleks dan penyesuaian cepat terhadap skema data sesuai dengan kebutuhan aplikasi.
- *Serverless database*
  Tidak perlu mengelola *server* dan infrastruktur secara manual karena telah diakomodasi oleh Firebase. Sehingga *developer*, dapat fokus pada pengembangan aplikasi dan tidak perlu menangani pemeliharaan dan operasi *database*.

2.1.13 Tinjauan Jurnal

A Jurnal 1, "Sistem Pendukung Keputusan Untuk Pemilihan Hotel di Kota Medan Dengan Menggunakan Metode *Simple Additive Weighting*", Tjokro, Stanley (2020)

Meningkatnya jumlah tempat wisata di Medan menyebabkan bertambahnya wisatawan yang datang ke Medan, sehingga bisnis perhotelan meningkat. Meskipun bisnis perhotelan di Medan meningkat, akan tetapi jumlah hotel yang banyak membuat wisatawan kesulitan untuk memilih hotel karena pengunjung perlu mempertimbangkan harga, pelayanan, lokasi, bintang dan juga kebersihan hotel. Pada penelitian ini, digunakan metode *Simple Additive Weighting* untuk merekomendasikan hotel di Medan yang akan menghasilkan rekomendasi berdasarkan bobot yang telah diatur oleh pengguna. Kriteria harga masuk kedalam kriteria *COST* karena semakin kecil nilai semakin baik, sedangkan kriteria lainnya masuk kedalam kriteria *BENEFIT* karena semakin besar nilai semakin baik. Nilai dari setiap kriteria didapat berdasarkan tabel rubrik penilaian, kemudian dibentuk matriks normalisasi berdasarkan dari nilai kriteria tersebut. Hasilnya kemudian dikalikan dengan bobot kriteria, lalu sistem menampilkan lima tempat penginapan dengan nilai tertinggi. Penelitian ini menyimpulkan calon pengunjung

hotel dapat menggunakan aplikasi ini untuk mencari rekomendasi hotel di Medan menggunakan metode SAW yang berbasis *website* dan memberikan hasil dalam bentuk peringkat.

B Jurnal 2, ”Sistem Penentuan Penginapan dengan Metode Promethee”, Muntiari, Novita Ranti, dkk (2023)

Setiap orang yang melakukan perjalanan ke suatu tempat untuk beberapa hari membutuhkan penginapan dengan kriteria tertentu dan berbagai kemungkinan untuk setiap orang. Permasalahannya terletak pada penentuan rekomendasi penginapan yang diinginkan karena terlalu banyak pilihan dan kriteria yang beragam. Studi ini melakukan pencarian penginapan di Yogyakarta dengan kriteria jarak, harga, kelas penginapan, fasilitas kamar, dan fasilitas pendukung. Metode Promethee digunakan untuk mendapatkan peringkat setiap alternatif dengan menghitung bobot untuk setiap kriteria. Nilai alternatif yang diinginkan oleh pengunjung diproses untuk menemukan nilai kriteria dan preferensi kriteria kemudian dihitung sisa aliran dan aliran masuk untuk mendapatkan aliran bersih. Penelitian ini menghasilkan peringkat data yang telah diuji menggunakan sistem dengan nilai aliran bersih tertinggi adalah 0,5, yaitu alternatif penginapan 1 yang diprioritaskan sesuai dengan kriteria tersebut menggunakan metode *Promethee* [9].

C Perbandingan Penelitian

Tabel 2.2. Perbandingan Penelitian Serupa

Keterangan	Jurnal 1	Jurnal 2
Metode	*Simple Additive Weighting* (SAW)	Promethee
Cakupan Wilayah	Medan	Yogyakarta
Kriteria	Harga, Bintang, Servis, Lokasi, Kebersihan	Jarak, Harga, Kelas Penginapan, Fasilitas Kamar, Fasilitas Pendukung
Lanjut pada halaman berikutnya		

Tabel 2.2 Perbandingan Penelitian Serupa (lanjutan)

Keterangan	Jurnal 1	Jurnal 2
Kelebihan	Dapat mengatur level bobot dari setiap kriteria	- Dapat mengatur level bobot dari setia kriteria - Nilai kriteria tidak dibatasi
Kekurangan	- Tampilan kurang menarik - Nilai kriteria dari setiap alternatif dibatasi	- Pengguna harus menginput detail dari setiap alternatif dengan sendirinya - Tampilan terlalu teknis

Berdasarkan perbandingan terhadap jurnal-jurnal dengan penelitian serupa, pada penelitian ini perancangan DSS dilakukan menggunakan metode MCDM TOPSIS dengan bobot kriteria yang dapat diatur oleh pengguna agar dapat dengan mudah disesuaikan dengan preferensi pengguna. DSS memanfaatkan *database* untuk menarik data tempat penginapan, sehingga pengguna hanya perlu melakukan sedikit *user activity* untuk mendapatkan rekomendasinya, Aplikasi yang dibuat juga berbasis *website* dan memanfaatkan fitur *localstorage* untuk menyimpan bobot yang telah diatur dan kondisi yang diinput pengguna untuk mencari rekomendasi.